MEASURES FOR THE SIMILARITY OF PATHS IN COMPLEX NETWORKS

MEASURES FOR THE SIMILARITY OF PATHS IN COMPLEX NETWORKS

MAREIKE BOCKHOLT

Prof. Dr. Katharina A. Zweig

Prof. Dr. Paul Lukowicz

Department of Computer Science
Technische Universität Kaiserslautern

October 12, 2015

Mareike Bockholt
Measures for the Similarity of Paths in Complex Networks
Master Thesis

MATRICULATION NUMBER: 388426
CONTACT: mareike.bockholt@cs.uni-kl.de

SUPERVISORS:
Prof. Dr. Katharina A. Zweig
Prof. Dr. Paul Lukowicz

LOCATION:
Graph Theory and Complex Network Analysis Group
Department of Computer Science
Technische Universität Kaiserslautern

SUBMITTED:
October 12, 2015

ABSTRACT

In various research fields, methods from complex network analysis have been applied in order to analyze complex systems. There are network representations for which the idea of traversing the network arises naturally, consider for example travels in a transportation or road network, packets being routed through the Internet, or humans navigating through a problem's state space. Although the concept of paths in networks is intuitive, to our knowledge, there has not been any approach to compare paths by similarity in order to cluster more similar paths in groups. Therefore, this work presents an analysis of path similarity and distance measures and a first attempt of clustering real path data by similarity. Aiming at deriving appropriate path similarity and distance measures, we identify characteristic features of paths a similarity measure can be based on as well as general properties a similarity measure for paths should fulfill. Guided by the identified features, we propose several similarity and distance measures for paths in networks and analyze the proposed measures for their properties. Furthermore, we use path data collected on a web-based learning tool where the paths represent the navigation of humans through the problem state of a game while solving the game. We compute the similarity value for each of the proposed measures for each pair of paths and cluster the paths by their similarity using an hierarchical clustering approach.

ZUSAMMENFASSUNG

Methoden der Netzwerkanalyse kommen in den unterschiedlichsten Forschungsbereichen zur Anwendung um komplexe Systeme zu analysieren. In einigen Modellierungen von Systemen als Netzwerke erscheint das Konzept, das Netzwerk als Transportmedium zu verwenden und einen Weg durch das Netzwerk zu finden, natürlich. Als Beispiele können Reisen in Transport- oder Straßennetzwerken, Navigation von Benutzern auf Webseiten oder andere menschliche Navigation in Netzwerken genannt werden. Obwohl das Konzept der Wege durch Netzwerke ein intuitives ist, hat es unseres Wissens nach bislang keinen Ansatz gegeben, Pfade in Netzwerken zu vergleichen und sie nach Ähnlichkeit zu gruppieren. Die vorliegende Arbeit beschäftigt sich deshalb mit möglichen Ähnlichkeitsmaßen für Pfade in Netzwerken, indem zunächst Eigenschaften von Pfaden identifiziert werden, auf deren Grundlage wir Ähnlichkeitsmaße entwickeln. Weiterhin postulieren wir allgemeine Eigenschaften, die Ähnlichkeitsmaße für Pfade erfüllen sollten und prüfen die vorgeschlagenen Maße auf diese Eigenschaften. Im zweiten Teil der Arbeit verwenden wir die vorgestellten Maße, um Pfade mit hierarchischem Clustering in Gruppen von ähnlichen Pfaden zu partitionieren und die Maße zu evaluieren.

CONTENTS

1

INTRODUCTION

1.1 MOTIVATION

In recent years, the analysis of complex networks has been applied in various disciplines: in almost every area of research, there are concepts or systems which can be modeled as complex networks in order to use methods from complex network analysis for investigating properties of the system. There are classical examples of the application of network analysis in the area of biology, chemistry, or sociology. Not in all, but in some of these systems modeled as networks, the idea of traversing the network arises naturally. Consider for example travels in transportation networks, packets being routed through the Internet, or a user navigating on a web page by following the links. All the mentioned entities use the underlying network as environment which they can traverse. By traversing the network, they create paths from one "place" in the network to another. Although the idea of paths in networks is intuitive and well-known, there has not been any research aiming at comparing paths in networks: given two different paths in the same network, what are the necessary conditions that need to be fulfilled such that the paths can be considered as similar or very different? How can we measure the similarity or distance of two paths in a network?

Apart from academic interest, being able to quantify the similarity of paths in a network might be interesting for several reasons: having a similarity measure for paths at hand, it might be possible to apply clustering algorithms in order to partition paths in groups of similar paths. This is especially of importance when a large amount of paths in a network is considered and it is necessary to choose a representative subset of paths. A desired property is then to have one or several typical paths for each group of paths.

Another scenario in which the similarity of paths can be used, might be recommender systems when the order of purchased or clicked items is of relevance. An example for this are e-learning systems in which the order in which the learning materials are viewed by the student is as important as the materials themselves. A student browsing on a e-learning platform creates a path of viewed documents and a recommender system might suggest documents to the student by comparing the student's learning path with the paths of other students and evaluating the students' learning success.

The above mentioned scenarios are only two examples in which the concept of a path similarity could of use. Therefore, the present thesis proposes an approach how to measure the similarity of paths in networks.

1.2 SCOPE OF THIS WORK

The present work aims at presenting a first approach for comparing paths in graphs and cluster them by similarity. Therefore, this thesis consists of two main parts, one is concerned with the development of appropriate similarity and distance measures for paths in networks, the second describes an approach of clustering real path data based on their similarity measures and comparing the clustering results from each similarity measure. More detailed, the present work is structured as follows: section 1.3 provides the definitions and notions which are needed in the further chapters, before chapter 2 will give an overview of existing work about similarity measures in general and especially of sequences. Chapter 3 then deals with possible similarity measures for paths in networks and their properties. Thus, section 3.1 proposes general properties which should be satisfied by a path similarity measure, before section 3.2 describes possible features of paths on which the development of a similarity measure could be based on. Section 3.3 uses the proposed path features to describe several different path similarity and distance measures which are analyzed for their properties in section 3.4. This concludes the first main chapter of the present work.

The second main chapter is chapter 4 which describes the clustering of real path data based on their similarity measure values. Section 4.1 gives an overview of existing clustering methods in order to justify the chosen clustering method before section 4.2 describes the source and the structure of the available path data. The values for each of the proposed similarity and distance measures are computed for each pair of paths and an hierarchical clustering approach is used to group similar paths in clusters. The methods, the results of this approach as well as the evaluation of the similarity measures is provided in section 4.4 before chapter 5.1 gives a summary of the work and outlines possible future work.

1.3 DEFINITIONS

1.3.1 *Basic definitions*

Most of the given definitions and notations are following Krumke and Noltemeier [25].

GRAPH A graph G is a tuple $G = (V, E)$ with a set of nodes V and a set of edges $E \subseteq V \times V$. For simplicity and without loss of generality, we assume $V \subseteq \mathbb{N}$. It can be then assumed to have an ordering on the nodes by taking the natural ordering on $\mathbb{N}$. A graph is *finite* if V and E are both finite sets. If E contains only unordered pairs of nodes $\{v, w\}$ with $v, w \in V$, the graph is called *undirected*. If E contains ordered pairs (v, w), the graph is called *directed*.

In a directed graph, for an edge $e = (v, w)$, we call v the source node of e and w the target node of e. We define the functions $\alpha : E \to V$ and $\omega : E \to V$ which yield the source respectively the target node of an edge ($\alpha((v, w)) = v$ and $\omega((v, w)) = w$). In an undirected graph, for an edge $e = \{v, w\}$ with $v \leq w$, we define $\alpha(e) := v$ and $\omega(w) := w$.

An edge e with identical source and target node, i.e. $\alpha(e) = \omega(e)$, is called *self-loop*. If E is a multiset, i.e. there are edges $e, e' \in E$ with $e \neq e'$ and $\alpha(e) = \alpha(e')$ and $\omega(e) = \omega(e')$, G is called a *multiple* graph.

If G is not multiple and does not contain any self-loops, G is called *simple*.

In the following, only finite, simple and undirected graphs are considered.

INCIDENCE, ADJACENCY, DEGREE An edge $e \in E$ is called *incident* to a node $v \in V$ if $\alpha(e) = v$ or $\omega(e) = v$. If there is an edge $e = (v, w) \in E$, the nodes v and w are said to be *adjacent* to each other or to be *neighbors*. The number of neighbors of a node v is called *degree* of v, i.e. in undirected graphs $deg(v) := |\{w \in V | \{v, w\} \in E\}|$. In directed graphs, we distinguish between the *in-degree* and the *out-degree*, i.e. the number of ingoing edges and the number of outgoing edges of a node v: $deg^-(v) := |\{w \in V | (w, v) \in E\}|$ and $deg^+(v) := |\{w \in V | (v, w) \in E\}|$ and $deg(v) := deg^+(v) + deg^-(v)$.

PATH A *path* p in a graph G is a finite sequence $p = (v_1 e_2 v_2 \dots v_{k-1} e_k v_k)$ with $k \in \mathbb{N}$, $v_1, \dots, v_k \in V$ and $e_i = \{v_{i-1}, v_i\} \in E$ for all $i \in \{2, \dots, k\}$. Because only simple graphs are considered in the following, a path is uniquely determined by its node sequence, and therefore, the notation of a path can be simplified to $p = (v_1 v_2 \dots v_k)$ with the same requirements as above. Let $V(p) = \{v_1, \dots, v_k\}$ and $E(p) = \{e_2, \dots, e_k\}$ denote the set of nodes and the set of edges which are contained in p, respectively. If a node v or an edge e is contained in a path p, we write $v \in V(p)$ and $e \in E(p)$, or also short $v \in p$ and $e \in p$.

For a path p, the functions α and ω can be defined accordingly: $\alpha(p) := v_1$ and $\omega(p) := v_k$ yield the start and end node of p. For the sake of better readability, we sometimes write α_p and ω_p instead of $\alpha(p)$ and $\omega(p)$.

Any subsequence $(v_i e_{i+1} \dots e_{i+j} v_{i+j})$ of p with $1 \leq i \leq k$ and $j \geq 0$ and $i + j \leq k$ is called a *subpath* of p. We denote this subpath by $p^{i,j}$. As special case, we define $p^{i,i-1} := ()$ as the empty path.

We say that p *induces an ordering* $\preceq_p \subseteq V(p) \times V(q)$ *on a subset of the nodes* by $v_{p_i} \preceq_p v_{p_j}$ iff $p_i \leq p_j$.

PATH PROPERTIES The *length* $|p| = k - 1$ of a path p is the number of edges in p. It holds $|p| \geq |E(p)|$. A path is called *simple* if none of the edges is used more than once, i.e. $e_i \neq e_j$ for all $i, j \in \{2, \dots, k\}$ with $i \neq j$. For a simple path p, $|p| = |E(p|)$ holds. A path is called *elementary* if it is simple and $v_i \neq v_j$ for all $i, j \in \{1, \dots, k\}$, hence if no edge and no node is contained more than once in p; only the same start and end node are allowed, i.e. $\alpha(p) = \omega(p)$. In the case that $\alpha(p) = \omega(p)$, p is called a *cycle*. Figure 1 shows an example in which the path $p = (1, 2, 3, 1, 4)$ is simple, but not elementary.

Let $\mathcal{P}_{v_1 \to v_k}$ be the set of all paths p with $\alpha(p) = v_1$ and $\omega(p) = v_k$. Let $\mathcal{P}_V = \bigcup_{v_i, v_j \in V} \mathcal{P}_{v_i \to v_j}$ the set of all paths in graph G. Let $\mathcal{P}_V^{\leq l}$ and $\mathcal{P}_V^{=l}$ the set of paths p in G with $|p| \leq l$ and $|p| = l$, respectively.

CONNECTEDNESS, DIAMETER For two nodes $v, w \in V$, let their *distance* $d(v, w)$ be the length of the shortest path between v and w:

$$d(v, w) = \min \{|p| \mid p \in \mathcal{P}_{v \to w}\}$$

In the case that there is no path from v to w, hence $\mathcal{P}_{v \to w} = \emptyset$, it is set $d(v, w) := \infty$. If $d(v, w) = 1$, v and w are *adjacent*.

An undirected graph is called *connected* if there is a path between any two nodes $v, w \in V$, i.e. $\forall v, w \in V$, it holds $d(v, w) < \infty$. A directed graph is called *strongly connected* if, for any two nodes $v, w \in V$, there is a path from v to w and a path from w to v. A directed graph is called *(weakly) connected* if, for any two nodes $v, w \in V$, there is a path from v to w or there is a path from w to v.

The *diameter* of a graph G is defined as $diam(G) := \max_{v,w \in V}\{d(v, w)\}$, hence the longest shortest path between any two nodes in the graph.

PATH OPERATIONS In the following, several operations on paths are defined. The operation of *path concatenation* of paths in a graph G is defined as $\oplus : \mathcal{P}_V \times \mathcal{P}_V \to \mathcal{P}_V$ with

$$\begin{aligned}
((v_{p_1} \dots v_{p_l}), (v_{q_1} \dots v_{q_m})) &\mapsto (v_{p_1} \dots v_{p_l} v_{q_1} \dots v_{q_m}) \text{ if } (v_{p_l}, v_{q_1}) \in E \\
(v, (v_1 \dots v_l)) &\mapsto (v v_1 \dots v_l) \text{ if } (v, v_1) \in E \\
((v_1 \dots v_l), v) &\mapsto (v_1 \dots v_l v) \text{ if } (v_l, v) \in E \\
(v, w) &\mapsto (vw) \text{ if } (v, w) \in E
\end{aligned}$$

For the definition of *path inversion*, the notion of an *edge inversion* is required. An edge $e = (v, w) \in E$ or $e' = \{v, w\} \in E$ is inverted by $inv(e) := (w, v)$ if $(w, v) \in E$ or $inv(e') = e'$ in an undirected graph. Path inversion is then defined as $inv : \mathcal{P}_V \to \mathcal{P}_V$ with $inv(p) := (v_k inv(e_k) \dots v_2 inv(e_2) v_1)$ for a path $p = (v_1 e_2 v_2 \dots v_{k-1} e_k v_k)$. Note that in a directed graph, $inv(p)$ does not need to exist if there is an edge $e_i \in p$ for which the inverted edge does not exist in G, i.e. $inv(e_i) \notin E$. In undirected graphs, the inverted path always exists.

SIMILARITY MEASURE A similarity measure over a set of objects X is a real-valued function $\sigma : X \times X \to \mathbb{R}$ which indicates how similar two objects of X are. The more similar two objects are, the higher the value of the similarity function should be. It is differentiated between unnormalized and normalized similarity measures: while σ can take arbitrary high (positive)

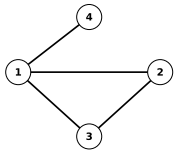


Figure 1: Example of a simple, but not elementary path.

values, the normalized similarity measure is constrained to the interval $[0,1]$, i.e. $\sigma_N : X \times X \to [0,1]$. For any similarity measure σ, a transformation to the interval $[0,1]$ can be achieved by

$$\sigma_N(x,x') = \frac{\sigma(x,x') - \min_{y,y' \in X} \sigma(y,y')}{\max_{y,y' \in X} \sigma(y,y') - \min_{y,y' \in X} \sigma(y,y')}$$

for $x, x' \in X$. The present work will consider similarity measures over $\mathcal{P}_V$.

DISTANCE MEASURE AND METRIC A *distance measure* over a set of objects X is a real-valued function $\delta : X \times X \to \mathbb{R}$ which indicates the dissimilarity or distance of two objects of X. The more dissimilar or distant two objects are, the higher the value of δ. Also for distance measures, it is differentiated between normalized and unnormalized distance measures: $\delta(x,y) \in \mathbb{R}$ and $\delta_N(x,y) \in [0,1]$ for all $x, y \in X$. Any distance measure can be transformed to a normalized distance measure by the transformation described above:

$$\delta_N(x,x') = \frac{\delta(x,x') - \min_{y,y' \in X} \delta(y,y')}{\max_{y,y' \in X} \delta(y,y') - \min_{y,y' \in X} \delta(y,y')}$$

for $x, x' \in X$. A distance function δ is called a *distance metric* if for any $x, y, z \in X$ the following conditions are satisfied:

(i) $\delta(x,y) \geq 0$ (*non-negativity*)

(ii) $\delta(x,y) = 0 \Leftrightarrow x = y$ (*coincidence*)

(iii) $\delta(x,y) = \delta(y,x)$ (*symmetry*)

(iv) $\delta(x,z) \leq \delta(x,y) + \delta(y,z)$ (*triangle inequality*)

For each normalized distance measure δ_N, there is an associated normalized similarity measure and vice versa. It is obtained by any transformation function $f : [0,1] \to [0,1]$ which is monotonically decreasing. For example, f might be defined as

$$\sigma_N(x,y) = f(\delta_N(x,y)) := 1 - \delta_N(x,y)$$

$$\text{or as} \qquad \sigma_N(x,y) = f(\delta_N(x,y)) := \frac{1}{\delta_N(x,y) + 1}$$

$$\text{or as} \qquad \sigma_N(x,y) = f(\delta_N(x,y)) := \mathrm{e}^{-\delta_N(x,y)}.$$

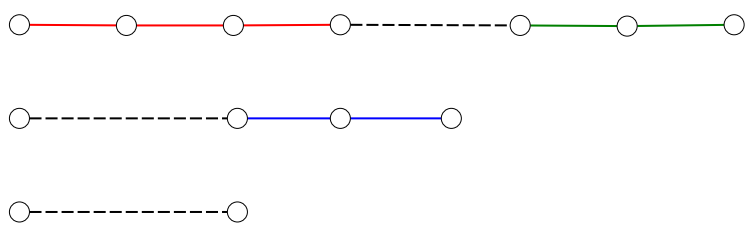


Figure 2: For illustration of path concatenation: In the top example, the red and the green path are concatenated by the black dashed edge, in the second example, a single node is concatenated to the blue path, in the third example, two single nodes are concatenated by the dashed edge, building then a path of length 1.

The present work consider distance measures and metrics over paths, hence over $\mathcal{P}_V$.

CLUSTERING Given a finite set of objects X and a distance measure δ or a similarity measure σ over X, a function f is called a clustering function if it returns, given X and δ or σ respectively, a partition of X. Therefore, the task of a clustering function is to group the given objects based on the similarity between them. Different clustering functions and algorithms are discussed in 4.1.

1.3.2 *Notation*

In the upcoming chapters, we use a notational convention. If not defined otherwise at some points, by default, the following symbols are meant as defined here:

- Let $G = (V, E)$ be the graph in which the considered paths are.
- Let $i, j, k, l, m, n \in \mathbb{N}$.
- Let v, w, x with any index be nodes of graph G and e, e' with any index be edges.
- Let $p, p', q, q', r, r' \in \mathcal{P}_V$ be paths: $p = (v_{p_1} e_{p_2} v_{p_2} \dots e_{p_k} v_{p_k})$ and $q = (v_{q_1} e_{q_2} v_{q_2} \dots e_{q_m} v_{q_m})$ and $r = (v_{r_1} e_{r_2} v_{r_2} \dots e_{r_l} v_{r_l})$.

2

RELATED WORK

To our knowledge, there is no existing work which analyzes the similarity of paths in graphs, but there is a broad variety of research about similarity measures in general and measuring the similarity of sequences in several application domains. In the area of graphs and networks, there are multiple approaches to measure the similarity or equivalence of nodes, but none which is concerned with the similarity of paths in graphs. In the following paragraphs, existing research approaches about similarity measures in general and in particular for sequences are presented as well as other work related to the research described in this thesis.

2.1 WORK ON SIMILARITIES AND CLUSTERING

The concept of similarity or distances is an essential one in many disciplines and fields. Therefore, there are countless works about similarity and distance measures in general, and special similarity measures for particular application areas, as well as analyses of the measures' properties, their applicability and performance. Especially in some tasks in machine learning and data analysis, where the goal is to find meaningful groups of objects, it is a crucial task to find an appropriate measure of similarity. In order to be able to group objects in meaningful cluster, it is necessary to define a similarity or distance on the set of pairs of objects. The challenge is that the objects do not need to be numerical, but can come from any context and have any structure. In machine learning, there are several approaches to deal with this problem.

In the following, a small selection of existing work is presented. The selection of works aims at showing a broad spectrum of how a similarity measure can be derived in different areas of applications. The methods of measure development is rather in the focus of the following section than the concrete derived measures. Therefore, the discussion starts with two axiomatic approaches – one for clustering and one for a similarity measure – in order to show how a measure or a clustering function can be found by stating assumptions in the very beginning of the development from which the measure is then derived. We then present a totally different approach which delegates the development of a distance measure to a learning algorithm. As an example of the development of a similarity measure in a specific application domain, we present the work about a similarity measure for two-dimensional shapes in the area of image processing.

For an overview of common existing similarity measures and their possible properties, see for example Gower [11]. An introduction to similarity and distance measures in data mining is given by Pang [43].

2.1.1 *Axiomatic approaches to similarity and clustering*

An axiomatic approach for deriving a similarity measure is used in the work of Lin [32] who presents an approach which does not start in the context of an application domain and derives a similarity measure for the application of interest, but rather proceeds the other way around: he defines the concept of similarity from an abstract point of view by stating general intuitions and assumptions about how a similarity measure should behave, independently of the domain and without using any assumptions or constraints of the domain for stating the measure. He names two main reasons for this top-down approach instead of an application-driven approach: *universality* and *theoretical justification*. The first postulates that the definition of a similarity should be applicable to objects from any domain, the only prerequisite in this work is that the domain has a probabilistic model, i.e. the probability distribution of the objects in the domain is known. The latter requires – in order to justify the choice of a similarity measure – that the development of a measure should happen in a top-down approach in the sense that it does not start from a formula, but rather a set of assumptions is stated how the desired measure should behave. In the ideal case, the similarity measure follows from the assumptions directly. Lin states three intuitions about how a similarity for any kind of objects should behave and derives six assumptions from them. Having formulated the assumptions, an information-theoretic similarity which satisfies the required assumptions can be derived. The three intuitions are the following: (i) the similarity between two objects should be larger if the two objects share more commonality while (ii) the similarity between two objects should be smaller if they have more differences, (iii) the similarity of two objects should reach its maximum if the objects are identical – no matter how much commonality they share. From this basic intuitions which are met by a number of existing similarity measures, Lin formulates six information-theoretic assumptions. From these assumptions he can then prove that the function which satisfies the assumptions is given by

$$sim(A, B) = \frac{I(common(A, B))}{I(description(A, B))},$$

i.e. the ratio of the amount of information contained in the commonality of the objects A and B, and of the amount of information contained in the description of A and B, where the amount of information of a statement is measured by the negative logarithm of the probability of the statement.

In the experimental section of the paper, Lin can show that the derived similarity measure satisfies universality which was stated as goal, by demonstrating its applications in different domains, for example string similarity or semantic similarity. The theoretical justification which is stated as second goal is satisfied by the used methodology.

A methodically similar approach with a different result and from a different topic was proposed by Jon Kleinberg in his impossibility theorem of clustering functions [23]. In his article from 2003, he introduces an axiomatic framework for algorithms for clustering objects in which he proposes three properties one could require from a clustering function. But while in the work of Lin [32], the made assumptions lead to a similarity measure, Kleinberg can show that there does not exist any clustering function which satisfies all three formulated properties. Though, interestingly, functions which obey two of the three properties or relaxations of the properties are well-known algorithms which are widely used in several application domains.

As already defined in section 1.3, the idea of clustering is to partition a given set of objects into subsets such that these subsets satisfy certain properties. The set of objects to be partitioned is given as $S = \{1, 2, \cdots, n\}$, a *distance function* is defined on any pair of the n objects: let $d : S \times S \to \mathbb{R}$ be the reflexive and symmetric distance function such that for any $i, j \in S$, it holds $d(i, j) \geq 0$ and $d(i, j) = 0 \Leftrightarrow i = j$ and $d(i, j) = d(j, i)$. A *clustering function* is then defined as function f which gets a set S and a distance function d on S and returns a partition Γ of S which are the *clusters* of S.

The properties that Kleinberg suggests for a clustering function are as follows:

Scale-Invariance A clustering function should be robust against any scaling of the distance function by a constant factor: For any distance function d and any $\alpha > 0$, it is required $f(d) = f(\alpha \cdot d)$, where $\alpha \cdot d := \alpha d(i, j)$ for any $i, j \in S$.

Richness A clustering function should be rich in the sense that every possible partition of S is a possible output of f – if the distance function can be chosen accordingly: Let $\text{Range}(f)$ denote the set of all partitions that f can produce for any distance function d, then $\text{Range}(f)$ should be equal to the set of all partitions of S.

Consistency If f produces a partition Γ of S with the distance function d, the clustering function should be insensitive against the following modification of the distance function leading to d': the distance between any pair of elements which are in the same cluster in Γ is decreased, the distance between elements which are in different clusters in Γ is increased. Then, f should produce the same partition with d as well as with d'. Formally, d' is defined as a Γ-transformation of d, if $\forall i, j \in S$ belonging to the same cluster of Γ, it holds $d'(i, j) \leq d(i, j)$, and $\forall i, j \in S$ belonging to different clusters of Γ, it holds $d'(i, j) \geq d(i, j)$. The requirement for f is then: Let d and d' be two distance functions. If $f(d) = \Gamma$, and d' is a Γ-transformation of d, then $f(d') = \Gamma$.

Kleinberg can then prove that for each $n \geq 2$, there is no clustering function which satisfies Scale-Invariance, Richness and Consistency. Furthermore, he considers some relaxations of the three properties and describes how well-known clustering methods, for example centroid-based clustering methods or single-linkage clustering relate to the relaxed properties.

Coming back to the challenge of finding an appropriate similarity or distance measure, two further works which deal with this topic are presented. Xing et al. [46] from the field of machine learning and statistics describe that having a

good distance metric at hand is a crucial point when objects need to be clustered according to this metric. The problem is that there are almost always several plausible ways to cluster objects, but not all clustering results are meaningful for the user in the end. Therefore, in order to get meaningful clustering results, the similarity measure is often manually changed to capture the most important aspects which should be reflected by the measure. For this reason, they propose a systematic way to identifya good similarity measure, by giving an algorithm which can learn a similarity measure for objects of a given domain. The input data which is used to learn the measure is a set of pairs of objects which are considered as similar by the user. Hence, the user specifies by examples which objects the measure should rate as similar. The algorithm then learns a distance metric which respects the given similar objects and can, after the learning phase, compute the distance value for any two objects from this domain. Xing et al. propose the development of their learning algorithm as the formulation of a convex optimization problem which can be solved by standard efficient, local-optima-free methods. Their results on artificial and real-world data look quite promising.

As an example from a specific application domain in which a good similarity measure was needed, the work of Latecki and Lakämper about the similarity of shapes is presented [29]. This work from the area of image processing describes the development of a similarity measure for two-dimensional shapes, i.e. silhouettes of objects. The main requirement for the similarity measure is its consistency with the principles of human visual perception: the similarity measure should rate shapes as similar which are rated as similar by humans even if the objects are mathematically different. Further requirements for the desired similarity measure are

- it should be robust against noise in the representation of the shapes, which might for example come from digitization errors,
- it should respect significant visual parts of the objects,
- it should be independent of orientation, scale, and position of the objects,
- it should not be restricted to a subset of possible shapes, i.e. it should be applicable to all shapes.

Latecki and Lakämper base their similarity measure on the idea of finding a function which establishes a correspondence between the visually significant parts of the two shapes of interest. The similarity measure then compares the values of an associated function (the so called tangent function) of the corresponding parts of the shapes. They can prove that their proposed similarity measure satisfies the requirements they stated and illustrate in examples that their similarity measure gives results which are quite consistent with the principles of human visual perception of objects.

2.1.2 *Similarities in graphs and networks*

In the context of graphs and networks, we are only aware of several works about similarity of nodes, for example [4, 31, 19]. For an overview of similarity and distance measures in networks, see Akcora and Ferrari [1].

In their work of 2002, Jeh and Widom propose the *SimRank measure* as a structural-context similarity for measuring the similarity of objects from any

domain – as long the objects have relationships between each other, i.e. can be considered as nodes in a graph. The main idea of their similarity measure is that objects should be considered as similar if they occur in similar contexts which is a self-referential concept.

Leicht et al. [31] propose anotherapproach, motivated by the concept of social networks: two vertices in a social network can be considered as similar if their neighbors in the network are similar. The most basic idea to capture this idea is to count the number of common neighbors for two nodes and take this value as similarity score. Formally, for a node $v \in V$, let $\Gamma(v)$ be the set of neighbor nodes, then a simple similarity of the two nodes $v_i, v_j \in V$ could be $|\Gamma(v_i) \cap \Gamma(v_j)|$. Though, this measure is not quite fair, because it is more likely that nodes with a higher degree will have an absolutely higher score than nodes with a smaller degree. However, for example, two nodes with ten neighbors each and ten common neighbors should get a higher similarity score than two nodes with 1000 neighbors each and 20 common neighbors. For this reason, it might be reasonable to normalize the measure for which there are several possibilities:

- normalize by the total numbers of neighbors of the two nodes which yields

  $$\frac{|\Gamma(v_i) \cap \Gamma(v_j)|}{|\Gamma(v_i) \cup \Gamma(v_j)|}$$

  and is known as *Jaccard index* [15].
- normalize by the the square root of the product of the cardinalities of the two neighbor sets which yields

  $$\frac{|\Gamma(v_i) \cap \Gamma(v_j)|}{\sqrt{|\Gamma(v_i)| \cdot |\Gamma(v_j)|}}$$

  and is known as *cosine similarity* [41].
- normalize by the minimum of the cardinalities of the two neighbor sets:

  $$\frac{|\Gamma(v_i) \cap \Gamma(v_j)|}{\min\{|\Gamma(v_i)|, |\Gamma(v_j)|\}}.$$

All of these measures are well-known and commonly used, however, the authors propose that two nodes might be considered as similar even if they do not share any common neighbors. As an example they mention the social networks of two companies in which the CEOs of both companies should be considered as similar to each other because of their position in the network – although their nodes might not share any neighbors. Based on this intuition, Leicht et al. use the idea of regular equivalence of nodes which is a similar concept to the *SimRank measure* of Jeh and Widom [19]: two nodes are said to be similar if they are connected to nodes which are themselves similar, and – as termination criterion for the recursion – a node is similar to itself. The measure resulting from these two assumptions can be formulated in matrix form which is why the respective algorithm can make use of standard linear algebra methods.

The basic intuition is the same as in the approach of Jeh and Widom, but in the derivation of the formulas, there is a significant difference: both measure formulas include the paths between the respective nodes, though, the proposed formula

of Jeh and Widom only considers paths of even length between the nodes which yields a similarity that is o for nodes connected only by paths of odd length. This counterintuitive behavior of the measure is avoided in the measure of Leicht et al.

Leicht et al. evaluate their proposed measure on one artificial network and two networks from real-world data which gives intuitively meaningful results.

2.2 WORK ON SEQUENCE SIMILARITY

As mentioned above, we are not aware of any work dealing with measuring the similarity or distance of paths in graphs or networks, however, there has been proposed many methods for comparing sequences of objects, two- and three-dimensionaltrajectories, click streams,or paths in documents from a broad range of fields and manifold types of approaches. We will present a few of them in the following section.

2.2.1 *Comparing object trajectories*

Thereis a wealth of literature oncomparing and classifying trajectories of objects moving in two- or three-dimensional space, for example [44, 20, 5, 48, 35]. The basic setting for all these works is the following: the video recordings from a surveillance camera observing a fixed outdoor scene is processed by extracting the trajectories which are made by the objects moving through the scene. For each observed object, a trajectory – a sequence of the object's positions at different time points – is extracted from the video recordings. The trajectories might be two- or three-dimensional. The goal is to build a system which is able to automatically extract, compare and classify the trajectories in order to distinguish between regular and irregular paths. For example, the camera is observing an area where some part must not be entered.If it happens that an object enters this area, the system should give an alert. For the comparison and classification of the trajectories, a similarity measure for trajectories is needed. However, all the approaches need to take into account that the available data contains a lot of noise and errors due to the extraction of the paths from the image data.

The goal of Vlachos et al. [44] in their work is to achieve an automatic classification of trajectories by a nearest neighbor classification for which a distance function for trajectories and an efficient indexing scheme is needed. More concretely, they assume to be given a database of trajectories, each given as a sequence of consecutive locations in a multidimensional space, and a query which is not already in the database. Goal is to find the trajectory in the database which is closest to the query trajectory. They formulate the following requirements for a desired similarity or distance function:

- it should be robust against noise in the data. It is probable that satisfying this requirement will prevent the measure to fulfill the triangle inequality because being robust against noise means that the measure will ignore some of the most dissimilar parts of the trajectories which might violate the triangle inequality.
- it should be robust against variations in time: since the trajectories are given as coordinates with time information, the distance function should

be robust against varying time intervals, different sampling intervals or different speeds of the objects.
- the distance function should be invariant against translation in space, i.e. recognize similar movements even if they are in different space regions, for example shifted by a constant which might be due to different centerings of the camera.
- the distance function should be able to handle trajectories of different lengths.
- it should be computationally feasible to compute the distance measure for two trajectories.

Following these requirements, Vlachos et al. propose a similarity based on the longest common subsequence of two trajectories which is defined as follows: For two trajectories A and B of moving objects with $A = ((x_{a_1}, y_{a_1}), \ldots, (x_{a_n}, y_{a_n}))$ and $B = ((x_{b_1}, y_{b_1}), \ldots, (x_{b_m}, y_{b_m}))$, where (x_i, y_i) are the coordinates of the object, the *head* of a sequence of length n is defined as the first $n-1$ elements of the sequence, hence $Head(A) := ((x_{a_1}, y_{a_1}), \ldots, (x_{a_{n-1}}, y_{a_{n-1}}))$. Furthermore, let $Last(A) := (x_{a_n}, y_{a_n})$ and $\pi_x((x_{a_i}, y_{a_i})) := x_{a_i}$ and $\pi_y((x_{a_i}, y_{a_i})) := y_{a_i}$ as well as $length(A) = n$.

With these notations, the *LCSS-distance* of two trajectories A and B can be defined with two parameters δ and ϵ as follows: With an integer δ and a real number $0 < \epsilon < 1$,

$$LCSS_{\delta,\epsilon}(A,B) := \begin{cases} 0 & \text{if } A \text{ or } B \text{ is empty} \\ 1 + LCSS_{\delta,\epsilon}(Head(A), Head(B)) & \\ \qquad \text{if } |\pi_x(Last(A)) - \pi_x(Last(B))| < \epsilon & \\ \qquad\quad \text{and } |\pi_y(Last(A)) - \pi_y(Last(B))| < \epsilon & \\ \qquad\quad \text{and } |length(A) - length(B)| \leq \delta & \\ max\{LCSS_{\delta,\epsilon}(Head(A), B), LCSS_{\delta,\epsilon}(A, Head(B))\} & \\ \qquad \text{otherwise} & \end{cases}$$

The LCSS tries to match the two sequences by starting at the end of the sequences. If the second case is satisfied, two elements of the sequences canbe matched, the LCSS is increased by one and the matched elements are not considered anymore. Here, the parameter ϵ controls how close in space the two elements must be to each other such that they are considered as close and will be matched.The parameter δ allows the two matched elements to be shifted in time. If the last two elements of the sequences cannot be matched, one of them is left unmatched and the next but last element is considered. Hence, LCSS counts the number of matched elements in the sequences.

A first similarity measure based on the LCSS model is

$$S1(\delta, \epsilon, A, B) = \frac{LCSS_{\delta,\epsilon}(A,B)}{min\{n, m\}}$$

which indicates the ratio of number of matched elements to number of elements that could have been matched in the ideal case.

In order to recognize parallel movements in different space regions, they extend their measure by allowing a linear shift of the trajectories in all space dimensions and choose the shift which maximizes similarity. This yields the measure $S2$ which allows the trajectories to be linearly shifted in the space. The family $\mathcal{F}$ of translations is defined as all functions $f_{c,d}$ with $f_{c,d}(A) = ((a_{x,1} + c, a_{y,1} + d), \ldots, (a_{x,n} + c, a_{y,n} + d))$. With this family of translations, the refined similarity measure

$$S2(\delta, \epsilon, A, B) = max_{f_{c,d} \in \mathcal{F}} S1(\delta, \epsilon, A, f_{c,d}(B))$$

is obtained. They can show that they only need to consider a finite number of translations in order to find the tranlsation which maximizes the similarity and this finite set of translations can be efficiently enumerated. This result is based on the observation that each shift with a translation function leads to a certain LCSS value from which there is only a finite number of different possible values. Vlachos et al. compare the clustering performance with their method with the clustering results with the euclidean distance and the dynamic time warping distance function on two different data sets and find that their method outperforms the previous methods in terms of accuracy efficiency, especially when the used data contain noise.

Buzan et al. [5] extend the approach from Vlachos et al. [44] and they are able to cluster the trajectories into groups of similar ones using an hierarchical clustering approach. Their method seems to, at least by visual inspection, yield meaningful groups of trajectories.

While the above presented approaches both use the LCSS as similarity measure and only consider spatial features of the trajectories, Junejo et al. [20] are the first ones which also take other features of the trajectories into account in order to group them by similarity. They present an algorithm which is able to distinguish trajectories which are spatially dissimlar and also trajectories which are spatially close, but different in their spatio-temporal features (as speed of the object or curvature information, including discontinuities in velocity, acceleration, and position of the trajectory).

In the training phase of their algorithm, the algorithm is given a sufficiently great amount of object trajectories $\mathbb{T} = \{t_1, \ldots, t_m\}$, extracted from video recordings of a stationary camera. If an object i was tracked through n frames, the object's trajectory is given as $t_i = \{(x_{i_1}, y_{i_1}), \ldots, (x_{i_n}, y_{i_n})\}$, whereas the tuples contain the two-dimensional image coordinates of the object in the corresponding frame. Each $t_i \in \mathbb{T}$ is smoothed by a moving average filter to remove outliers and reduce the noise.

As distance measure for two trajectories t_j and t_i, the Hausdorff distance $d_{Hausdorff}$ is used which is defined as

$$d_{Hausdorff}(t_i, t_j) = max\{d(t_i, t_j), d(t_j, t_i)\}$$

with

$$d(t_i, t_j) = max_{a \in t_i} min_{b \in t_j} \|a - b\|$$

with an appropriate norm $\|\cdot\|$. Therefore, the Hausdorff distance takes the largest (smallest) distance of any two points of the trajectories as measure how similar respectively distant two trajectories are.

One advantage is its capability to compare trajectories of different lengths, a significant disadvantage is that outliers or noise will considerably influence the measure, even if the trajectories are close in all other points.

After clustering the training trajectories into groups of similar trajectories, a spatial envelope (or corridor) is calculated for each cluster which encloses all trajectories of the group and represents the spatial extent of all trajectories in the cluster. An average path for each group is also computed.

For assigning new trajectories to the existing clusters, two conditions are considered: first, 90 % of the points of the new trajectory must lie within the envelope of the cluster, second, the Hausdorff distance between the new trajectory and the average path must not be larger than the largest distance between the envelope boundaries. If these conditions are satisfied, the trajectory is checked for similarity of velocity and curvature features which is not of interest here. If the trajectory cannot be assigned to any of the existing cluster, it is marked as anomalous.

In 2006, Zhang et al. [48] present an overview and evaluation of often used similarity measures for objects' trajectories from video surveillance scenes. Among other, they present and test the following measures for trajectories:

- *Euclidean distance* as proposed by Fu et al. [10]
- *Euclidean distance with principal components analysis* as proposed by Bashi et al. [3]
- *Hausdorff distance* as proposed by Lou et al. [34] and Junejo et al. [20]
- *LCSS similarity* as proposed by Buzan et al. [5] and Vlachos et al. [44]
- *dynamic time warping* as proposed by Keogh and Pazzani [22]

For the evaluation of the listed measures, Zhang et al. use 130 trajectories, extracted from a recorded surveillance scene of three hours. There is a ground truth for the trajectories generated, by manually labeling the trajectories (with one out of 13 labels). Each similarity for the trajectories is computed and the trajectories are clustered into 13 groups, using spectral clustering, for each similarity measure. The clustering results for each measure are evaluated by correct clustering rate according to the ground truth and computational efficiency. Zhang et al. find that the correct clustering rate is almost the same for the euclidean distance, the euclidean distance with principal components analysis, dynamic time warping and the LCSS similarity, however, only the LCSS similarity is able to correctly classify trajectories with different speeds. The euclidean distance with or without principal components analysis is computationally the least expensive, dynamic time warping and the LCSS measure are more costly, but still computationally cheaper than the Hausdorff distance. Though, the LCSS similarity needs adjustment for the two parameter which adds further costs. In the presence of noise in the data, the euclidean distance with principal components analysis and the dynamic warping function perform best.

2.2.2 *Comparing sequences of events*

Besides the existing work about the similarity of trajectories, there are approaches which analyze sequences of events. They might be sequences of events

in telecommunication data, ordered lists of courses a student has taken during his or her studies, or sequences of stock prices from financial data.

In their article from 1997, Mannila and Ronkainen [37] describe a model for measuring the similarity of event sequences which can be efficiently computed by a dynamic programming approach. They assume to be given an ordered sequence of events where each event is a tuple of the description of event type and the timestamp of the event. They postulate that an appropriate distance measure for event sequences should satisfy non-negativity, coincidence, symmetry and the triangle inequality, i.e. the measure should be a metric. They therefore propose a distance measure which is based on the idea of an edit distance between event sequences. Given two sequences, the following edit operations are allowed to transform the one sequence into another:

- $insertion(e, t)$: inserting an event e to the sequence at a certain time point t
- $delete(e, t)$: delete an event e from the sequence at a time point t
- $move(e, t_1, t_2)$: move an event e from time point t_1 to a different time point t_2

where each of the operations has certain costs

- $cost(insertion(e, t)) := w(e)$, with $w(e)$ is a constant proportional to the inverse of the number of occurrences of e in a long reference sequence, i.e. it is more expensive to insert a rare event than a common one
- $cost(delete(e, t)) := w(e)$ with the same $w(e)$ as above
- $cost(move(e, t_1, t_2)) := c \cdot |t_2 - t_1|$ with c a constant, i.e. it is more expensive to move an event further in time than closer. This cost measure assumes that the occurrence times in different sequences are approximately in the same scale, otherwise the difference of time points of different scales would yield unexpected results

The distance of two event sequences is then the minimal cost to transform one sequence into the other. Mannila and Ronkainen can show that this distance measure is indeed a metric and can be computed efficiently by using a dynamic programming approach. A more detailed description of the method and the results can be found in [38].

In a work of Mannila and Moen [36] which also deals with event sequences, there can be found interesting ideas about the similarity of sequences. This article is actually concerned with the development of a useful notion of similarity between *types of events* occurring in sequences. They suggest the approach of considering event types as similar if they occur in similar contexts in the sequences. The context of an occurrence can be understood as the set of event types occurring within a certain time limit before the occurrence of the event type of interest. For this goal, several possibilities to compare two different contexts in sequences are developed: given m event types, the context of an event in a sequence can be modeled as m-dimensional vector with 0 and 1 as entries, where the i-th entry indicates if the i-th event type is element of the corresponding context or not. Then, there are several possibilities to compare two contexts, for example,

- Hamming distance of the two binary vectors, i.e. the number of differing vector entries,

- the two vectors can be considered as samples from two probability distributions, the distance of the two vectors are then computed by computing the difference of the corresponding distributions,
- each of the context vectors is associated with a centroid vector of the same dimension, where the i-th entry of the centroid vector is the mean value of the i-th entry in the context vector. The distance between the context vectors is then the L_1-distance between the associated centroid vectors.

It should be noted, though, that all three proposed methods to compare contexts in sequences are set-based measures in which the order of events occurring in the context is not considered anymore.

In an experimental evaluation, they use the centroid vector measure and enrollment data from the computer science department of the university of Helsinki. One sequence in the data is associated with one student and is an ordered sequence of courses in which (out of 18) the student was enrolled during his or her studies. The assumption is that two courses should be similar if they occur in the same stage of the curriculum. For each of the 18 courses, the centroid vector distance is computed from the course sequences of about 5000 students and compared with ground truth which was generated as follows: each of the courses has a recommended term in which the course should be taken, therefore, for the similarity of two courses, the difference between the ordinal number of the recommended terms of the courses was taken as ground truth for the distance of the courses. Mannila and Moen find a correlation between the course similarity based on their context method and the described ground truth.

2.2.3 *Further approaches*

Furthermore, there are approaches from different research areas which consider the similarity of sequences or paths, for example [30, 47, 26, 7, 28, 27, 12, 45, 39] which are described briefly in the following paragraph.

Lee et al. [30] consider the problem of finding an efficient indexing scheme for large data bases of XML documents. The computation of the index is based on the contained paths in the tree-structured document and they therefore propose a similarity of paths in XML documents. Yang and Wang [47] propose a model which uses significant statistical properties of sequences (for example from biological data, as genome sequences) to compare and efficiently cluster them. Kumar [26] who is in the field of knowledge discovery in databases investigates in his PhD thesis the clustering of sequential data and proposes as similarity measure for sequences the linear combination of a set based and an order based measure where the two parameters can be chosen according to the application scenario.

Das et al. [7] introduce a model for measuring the similarity of time series as they occur in financial or scientific applications. Their measure is based on the assumption that two series should be similar if they show similar behavior for a large part of their length. Laasonen presents an approach to compare ordered sequences of GSM cells from users with mobile phones who move around, and each time the mobile phone is registered in a cell, the cell identifier is added to the sequence. For the task of predicting the next cell a particular user will move next,

he introduces a similarity measure and route merging, clustering and prediction methods [27, 28].

In the context of user navigation in web pages, particularly in e-learning environments, the work of Gündüz and Öszu [12], of Wang and Zaïane [45] and of Mor and Minguillón [39] area of interest.

3

MEASURING THE SIMILARITY OF PATHS

3.1 PROPERTIES OF SIMILARITY AND DISTANCE MEASURES FOR PATHS

In section 1.3, a definition of similarity and distance measures for paths is given. The definition given there is rather general and allows a broad variety of functions to be similarity or distance measures for paths. Though, not every function which fulfills the definition of a similarity or distance measure, makes sense to be used for measuring the distance or similarity of paths. Therefore, before functions are proposed as similarity and distance measures in section 3.3, we formulate properties that one could require from a similarity or distance measure. There is not a single similarity or distance measure in section 3.3 which satisfies all properties formulated in the following paragraphs. It is an open question whether there exists such a distance or similarity measure at all.

(i) PREFIX AND SUFFIX CONSISTENCY If two paths share a common prefix or suffix, a similarity measure should judge them more similar than if the paths did not share this common prefix or suffix.

A similarity measure σ respectively a distance measure δ satisfies *suffix consistency*, if for any two paths $p \in \mathcal{P}_{V \to v_i}$ with end node v_i and $q \in \mathcal{P}_{V \to v_j}$ with end node v_j and a path $r \in \mathcal{P}_{v_k \to V}$ with $(v_i, v_k), (v_j, v_k) \in E$,

$$\sigma(p, q) \leq \sigma(p \oplus r, q \oplus r)$$

respectively

$$\delta(p, q) \geq \delta(p \oplus r, q \oplus r)$$

holds.

Furthermore, σ respectively δ satisfies *prefix consistency*, if for any two paths $p \in \mathcal{P}_{v_i \to V}$, $q \in \mathcal{P}_{v_j \to V}$ and $r \in \mathcal{P}_{V \to v_k}$ with $(v_k, v_i), (v_k, v_j) \in E$,

$$\sigma(p, q) \leq \sigma(r \oplus p, r \oplus q)$$

respectively

$$\delta(p, q) \geq \delta(r \oplus p, r \oplus q)$$

holds.

Informally, this means that two paths will become more similar if a common prefix or suffix is appended to them (cf. figure 3).

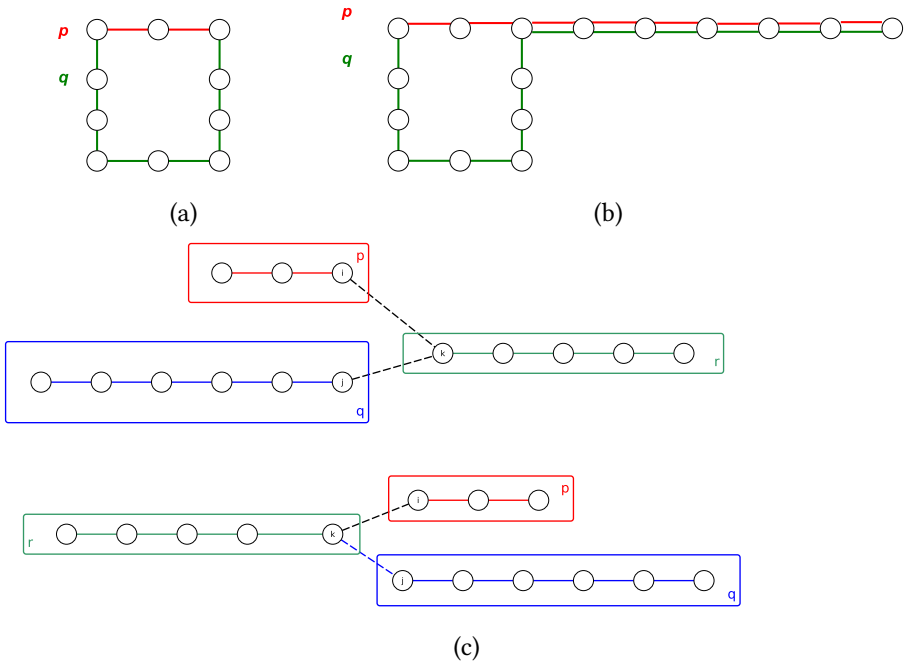


Figure 3: Illustration why the prefix and suffix consistency might be a desired property for a similarity or distance measure for paths: Although the paths in figure 3a and 3b contain the same substructure, it is intuitively clear that the paths in 3b should be rated more similar than the paths in figure 3a. For this reason, the prefix and suffix consistency is defined which is depicted in figure 3c: the paths $p \oplus r$ and $q \oplus r$ should be rated as more similar than the paths p and q. Similar for prefix consistency: the paths $r \oplus p$ and $r \oplus q$ should get an higher similarity score than the paths p and q.

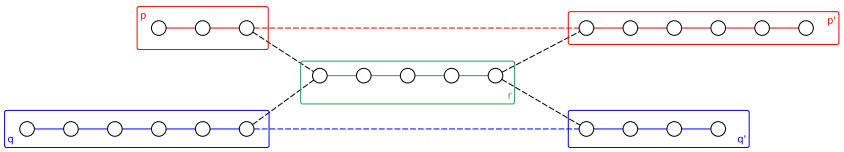


Figure 4: An illustration for insertion consistency: Paths with a common subpaths should get an higher similarity value than the same paths without the common subpath. In this example, the red and the blue path, i.e. $p \oplus p'$ and $q \oplus q'$ should be less similar than the paths $p \oplus r \oplus p'$ and $q \oplus r \oplus q'$. The common subpath does not need to be inserted at the same position in both paths, the only requirement is that that the edges for concatenation exist in the graph.

(ii) INSERTION CONSISTENCY More general than the prefix and suffix consistency is the insertion consistency. While prefix and suffix consistency deals with common subpaths at the end or beginning of the paths, insertion consistency states the desired behavior of similarity and distance measures when two paths share a common subpath at any position. Intuitively, the similarity of two paths should be larger if they share a common subpath than if they do not.

A similarity measure σ respectively a distance measure δ satisfies *insertion consistency*, if for any paths $p, q, p', q', r \in \mathcal{P}_V$ with $\{(\omega_p, \alpha_r), (\omega_q, \alpha_r), (\omega_r, \alpha_{p'}), (\omega_r, \alpha_{q'})\} \subseteq E$,

$$\sigma(p \oplus p', q \oplus q') \leq \sigma(p \oplus r \oplus p', q \oplus r \oplus q')$$

respectively

$$\delta(p \oplus p', q \oplus q') \geq \delta(p \oplus r \oplus p', q \oplus r \oplus q')$$

holds. An illustration can be found in figure 4.

(iii) CONCATENATION CONSISTENCY When paths are concatenated, the similarity of the concatenated paths should not be smaller than the similarity of the single paths. Therefore, a similarity measure σ and a distance measure δ, respectively, satisfy *concatenation consistency*, if for any paths $p, p', q, q' \in \mathcal{P}_V$,

$$\sigma(p \oplus p', q \oplus q') \geq \min\{\sigma(p, q), \sigma(p', q')\}$$

and

$$\delta(p \oplus p', q \oplus q') \leq \max\{\delta(p, q), \delta(p', q')\}$$

holds, respectively.

(iv) EDGE IMBALANCE CONSISTENCY A larger number of common edges should increase the similarity of two paths. The property capturing this intuition is called edge imbalance consistency, since it considers paths of equal length with different number of common edges which is why they are considered to be in an edge imbalance.

A similarity measure σ and a distance measure δ, respectively, satisfy *edge imbalance consistency* if, for any paths $p, q, r \in \mathcal{P}_V$ with $|p| = |q| = |r|$, it holds that

$$|E(p) \cap E(q)| > |E(p) \cap E(r)| \Rightarrow \sigma(p,q) > \sigma(p,r)$$

respectively

$$|E(p) \cap E(q)| > |E(p) \cap E(r)| \Rightarrow \delta(p,q) < \delta(p,r).$$

(v) BOUNDEDNESS It might be required that the similarity or distance measure for two paths is bounded from above by a constant. Therefore, a similarity measure σ or a distance measure δ satisfies *C-boundedness*, if for any paths $p, q \in \mathcal{P}_V$,

$$\sigma(p,q) \leq C \qquad \text{or} \qquad \delta(p,q) \leq C$$

holds for a constant $C \in \mathbb{N}$. The constant C can be chosen according to the underlying graph G. Normalized similarity or distance measures are by definition 1-bounded. If the bound is sharp, i.e. for C-bounded similarity measure σ or distance measure δ, there are paths p, q such that $\sigma(p,q) = C$ or $\delta(p,q) = C$, we say that σ or δ satisfies *strong C-boundedness*. For C-bounded similarity measures, the associated distance function can be obtained by $\delta(p,q) = C - \sigma(p,q)$. For C-bounded distance measures, the associated similarity measure can be obtained by $\sigma(p,q) = C - \delta(p,q)$.

(vi) NON-NEGATIVITY While the property of boundedness makes sure that the measure is bounded from above, it might be necessary to bound the measure from below. We say, a distance measure δ or a similarity measure σ satisfies *non-negativity*, if for any paths $p, q \in \mathcal{P}_V$,

$$\delta(p,q) \geq 0 \qquad \text{or} \qquad \sigma(p,q) \geq 0$$

holds. Normalized distance and similarity measures are by definition non-negative.

(vii) COINCIDENCE Coincidence is one of the four properties that are necessary for a distance measure to be called a distance metric. A distance measure δ satisfies *coincidence*, if for any two paths $p, q \in \mathcal{P}_V$, it holds

$$\delta(p,q) = 0 \Leftrightarrow p = q.$$

For a similarity measure, it is only appropriate to require coincidence, if it is strongly C-bounded. Hence, a strongly C-bounded similarity measure σ satisfies *coincidence* if for any two paths $p, q \in \mathcal{P}_V$, it holds

$$\sigma(p,q) = C \Leftrightarrow p = q.$$

(viii) SYMMETRY A further property that one could require from a similarity or distance measure is symmetry: it should not make a difference if path p is compared to q or q is compared to p. Therefore, a distance measure or a similarity measure satisfies *symmetry* if for any two paths $p, q \in \mathcal{P}_V$,

$$\sigma(p,q) = \sigma(q,p) \qquad \text{or} \qquad \delta(p,q) = \delta(q,p)$$

holds, respectively.

(ix) TRIANGLE INEQUALITY The triangle inequality is also one of the four axioms which make, if satisfied, a distance metric from a distance measure and can also be appropriate to be required from a distance measure for paths. A distance measure δ satisfies the *triangle inequality* if, for any paths $p, q, r \in \mathcal{P}_V$,

$$\delta(p,r) \leq \delta(p,q) + \delta(q,r)$$

holds.

For a C-bounded similarity measure for paths, an equivalent triangle inequality can be formulated by using its associated distance measure:

$$\begin{aligned}\sigma(p,r) &= C - \delta(p,r)\\ &\geq C - (\delta(p,q) + \delta(q,r))\\ &= C - (C - \sigma(p,q)) - (C - \sigma(q,r))\\ &= \sigma(p,q) + \sigma(q,r) - C\end{aligned}$$

3.2 ABOUT SIMILARITY OF PATHS IN GENERAL

In the next section, we propose several distance and similarity measures for paths which use different approaches to capture the similarity or distance of paths in graphs. For a subset of the proposed measures, section 3.4 discusses which properties stated in section 3.1 are satisfied by the measures. But before, we discuss in the following section general approaches how to measure the similarity or distance of paths.

3.2.1 *Development of similarity measures*

There are in general several types of approaches how to measure the similarity or distance of paths in a graph, depending on which characteristics of paths are considered to be important such that they are integrated in the computation of a similarity. The decision of which properties are the most relevant features of a path is highly dependent on the meaning of a path – whether the path of interest is a route through a road network or whether it represents a part of a food chain in a food web. The same principle holds for the development of similarity measures in other domains: for example the computation of string similarity is dependent on the meaning of the strings – whether strings are considered as words with a semantic meaning, and where the similarity of the *meaning* is of interest, or whether the single letters of the string carry information, and where the similarity of the actual *word* is wanted, for example strings as sequences of genetic information. In the latter case, a similarity measure such as the edit distance might be appropriate while this does not make any sense when the strings as *words* are compared. Comparing the strings *pineapple* and *strawberry* with an edit based measure is not reasonable if the meaning of the two words is important. But even then, it is not obvious how to compare the two words: if it is of interest that both words stand for items which can both be classified as

fruits, the two words would get a high similarity value. Though, if the color of the items represented by the words is relevant, they should get a low similarity value.

The main point of this paragraph is that there is not *the* similarity measure to use when comparing objects, it always depends on the context and the purpose of comparing objects: why are certain objects being compared and which are the characteristic features of the object which are of interest? These questions need to be answered in order to develop – or to choose – an appropriate measure of similarity for objects.

For this reason, the next section discusses some properties of paths which could be of interest in certain application domains, before similarity and distance measures based on these properties are proposed in section 3.3.

3.2.2 *Features of paths*

The following paragraph lists properties of paths which could be relevant in certain contexts and domains and can be a starting point for the development of similarity or distance measures for paths. Surely, this list can not cover all possible aspects of paths, but aims at stating an exemplary subset.

POSITION IN THE GRAPH Thinking of route planning in a road network, for two given paths, it might be an important feature where they are located in the network. A route might be considered as similar to a second route with the same starting and end point if it always uses roads which are parallel and close to the roads the second path uses. On the other hand, a route which takes a long detour might be considered as dissimilar to the other one.

Therefore, the position of paths in the graph might be an important factor to consider for the development of similarity or distance measures for paths. The challenge for the development of similarity or distance measures will be to find an appropriate way to measure the distance of two paths in a graph.

CONTAINED ELEMENTS It might be that for comparing paths the contained elements are the essential feature and two paths should be considered as very similar if they share a large amount of elements. If the Wikipedia network is taken as example which contains articles as nodes and links between articles as directed edges, a path from one article to another is a sequence of articles in which each article is reached by following one of the links in the preceding article. If two of such paths have to be compared, it could be reasonable only to consider the contained articles in the paths as a set, and rate them as dissimilar if they do not share any common articles, while they are rated as similar if they have large number of common articles.

ORDER OF CONTAINED ELEMENTS In other application domains, the order of the nodes which is induced by the path is of essential importance. An example can be found in the area of learning analytics (for an overview of the

field of learning analytics see [40]). Let there be a student aiming at understanding a particular concept. He or she has several documents at hand in which certain aspects of the concept are explained in different degrees of detail, difficulty, or in different styles. By reading the different documents, the student creates a path through the network of documents. In this case, the order in which the documents are read by the student is a key property of the path: the order in which the documents are read play an important role whether and how fast the student will understand the required materials. Starting with the most advanced material will certainly have another effect than starting with a document giving a rough overview of the topic or explaining the most basic aspects and reading the advanced material afterwards.

Therefore, the order of elements which is imposed by a path can be a feature which might be taken into account in the development of a similarity or distance measure for paths.

STRUCTURE A further aspect of paths that should be integrated in the computation of similarity in some application areas is the shape or structure of the path which is probably most difficult to quantify. If again the road network is taken as an example in which road junctions are nodes and edges represent roads connecting the junctions. A path is then a route from some starting point to some end point, using roads of the network. If such paths are for example viewed from a athlete's perspective who is running or biking these routes, two of such paths can be considered as quite similar even if they in totally different places and do not share any common roads. Their similarity is only based on their shape or structure, for example there are two routes which are both round trips with a lot of turns to the right, there are two points on the route where it crosses a different part of the route, and the home stretch is a long straight road. It might be that at some other place in the world, there is another route with these properties and which could be regarded as similar to this one. This seems to be an artificial example, and the further work will not concentrate on this kind of similarity of paths, but it is meant to illustrate the wide variety of possible properties on which a similarity measure can be built on.

3.3 PROPOSING SIMILARITY AND DISTANCE MEASURES FOR PATHS

This section proposes several similarity and distance measures for paths, based on the path features presented in 3.2.2. There will be found more possible similarity and distance measures than they can be analysed and discussed in this work. Therefore, there are measures which are introduced and defined, but will not analysed and used in the following chapters. Their analysis and evaluation is left for future work. The measures in bold characters are part of the next chapters, the remaining measures are introduced in order to show that there are more measures possible and which need to be evaluated in future work. An overview of the introduced measures can be found in table 1.

3.3.1 *Position based distance measures*

The following distance measures for two paths are based on the assumption that the position of the paths in the graph is the essential feature for similarity computation, and that two paths are more similar to each other if they are *closer* to each other. Hence, the following three measures aim at compute some kind of spatial distance between paths.

HAUSDORFF DISTANCE Junejo et al. [20] propose in their work about trajectory analysis in video surveillance systems to use the Hausdorff distance as distance measure for trajectories. The Hausdorff distance was developed to measure how close two subsets in a metric space are and can be adapted to trajectories and paths. For two paths, the Hausdorff distance is defined as

$$\delta_{Hausdorff}(p,q) := max\{max_{v\in p} min_{w\in q} d(v,w), max_{w\in q} min_{v\in p} d(w,v)\}$$

and takes therefore the longest shortest path between any nodes of the two paths. It therefore considers very specifically one particular property of the two paths, namely their largest distance to each other, and leaves other properties of the paths aside. There might be cases where this approach can be appropriate, but it is necessary to be aware of the strong effect the maximization of the distance between the two paths has.

Since the maximal value the Hausdorff distance can take is the diameter of the graph, a possible normalization for the Hausdorff distance could be by the diameter:

$$\delta_{Hausdorff,N}(p,q) := \frac{\delta_{Hausdorff}(p,q)}{diam(G)}$$

The Hausdorff distance is very sensitive to outliers, in the sense that the Hausdorff distance will remain unchanged as long as the maximal distance between the two paths does not change – even if the rest of the paths are the same. This might be an unwanted effect in certain scenarios, that is why we propose an *average distance* of two paths as distance measure.

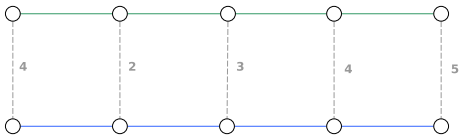
SIMPLE AVERAGE DISTANCE Idea of the simple average distance is to calculate the distance in the graph from each node in p to its counterpart in q and to calculate the average of these node distances. The main problem in the calculation of the average distance is to find the appropriate counterpart of each node. As a first idea, we propose to constraint the distance measure on paths with equal length and compare the i-th node of the paths with each other. For two paths $p, q \in \mathcal{P}_V$ with $|p| = |q| = k - 1$, the *simple average distance* is defined as

$$\delta_{sad}(p,q) := \frac{1}{k}\sum_{i=1}^{k} d(v_{p_i}, v_{q_i}).$$

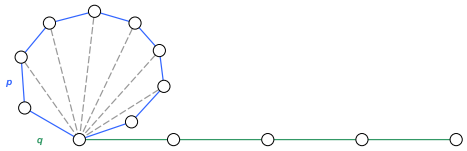
An illustration can be found in figure 5a. Since the maximal value which δ_{sad} can take for two paths is the diameter of the graph in which the paths lie, the *normalized simple average distance* is defined as

$$\delta_{sad,N} := \frac{\delta_{sad}(p,q)}{diam(G)}.$$

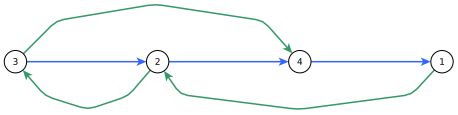
The normalized measure is only needed if a value in $[0,1]$ is required or the distance of paths in two different graphs are compared to each other. The simple average distance has two main deficiencies: it is only applicable to paths of equal length, and the matching which node of the one path is compared with which node of the second path, is very naive and might not be a good choice in many cases. For these reasons, we propose the *matched average distance*.



(a) An example for the simple average distance. The dashed lines represent which node is compared with which node where the numbers indicate the distances of the nodes in the graph. The two shown paths would get a distance value of $\frac{1}{5} \cdot (4+2+3+4+5) = \frac{18}{5}$.



(b) Mapping example for matched average distance. The dashed lines represent the mapping of g.



(c) Two example paths for the tuple similarity. The edges are drawn directed for a better readability although we consider undirected graphs.

Figure 5: Examples for the similarity and distance measures.

MATCHED AVERAGE DISTANCE The matched average distance is defined as follows: for two paths $p,q \in \mathcal{P}_V$ with $k-1 = |p| \geq |q|$, let $g : V(p) \to V(q)$ a function which maps each node of p onto a node of q. Given a mapping g and two paths $p,q \in \mathcal{P}_V$ with $k-1 = |p| \geq |q|$, the *matched average distance* is then defined as

$$\delta^g_{mad}(p,q) := \frac{1}{k}\sum_{i=1}^{k} d(v_{p_i}, g(v_{p_i}))$$

Furthermore, we define

$$\delta^g_{mad}(q,p) := \delta^g_{mad}(p,q)$$

and in case $|p| = |q| = k - 1$, we define

$$\delta^g_{mad}(p,q) = \delta^{g'}_{mad}(q,p) = \min\left\{\frac{1}{k}\sum_{i=1}^{k} d(v_{p_i}, g(v_{p_i})), \frac{1}{k}\sum_{i=1}^{k} d(v_{q_i}, g'(v_{q_i}))\right\}$$

with $g' : V(q) \to V(p)$. The mapping g can be chosen appropriately for the given application scenario, as intuitive mapping, we propose g to map each node of p on the by distance closest node of q, i.e.

$$g(v_{p_i}) \in \{v_{q_j} \in V(q) | \forall v_{q_{j'}} \in V(q) : d(v_{p_i}, v_{q_{j'}}) \geq d(v_{p_i}, v_{q_j})\}.$$

Hereinafter, this mapping is used and δ_{mad} is used without the superscript g. Furthermore, when using this mapping g, the formula for the matched average distance can be simplified to

$$\delta_{mad}(p,q) = \frac{1}{k}\sum_{i=1}^{k} \min_{w \in q} d(v_{p_i}, w).$$

Note that with this mapping, it might happen that there are nodes in path q which are not matched at all, although it is the shorter path of the two (consider for example figure 5b, where all nodes of p are mapped onto one node of q).

Finding a mapping which captures the intuition of similarity in a given scenario will be the main challenge when applying the matched average distance.

Since also this distance measure takes its maximal value when the distance of each node of p to its mapped node of q is the diameter of the graph, the *normalized matched average distance* results from the normalization by the diameter of the graph:

$$\delta^g_{mad,N}(p,q) := \frac{\delta^g_{mad}(p,q)}{diam(G)}$$

3.3.2 *Element based similarities*

The following two similarity measures are based on the assumption that the most characteristic feature of a path is the contained elements. Two paths are considered as more similar if they share an (absolutely or relatively) higher number of elements.

NODE SET SIMILARITY The most simple approach to capture this idea is to count the number of elements the two paths share. Therefore, we define the *node set similarity* for two given paths p, q as

$$\sigma_{nss}(p,q) := |V(p) \cap V(q)|$$

which takes the absolute number of nodes the two paths have in common. As this measure does not take into account the length of the two paths at all, and therefore, longer paths with a larger set of nodes will be more likely

have a higher value of similarity than shorter paths. As consequence, this measure is normalized by the number of nodes the two paths *could* have in common and define

$$\sigma_{nss,N}(p,q) := \frac{\sigma_{nss}(p,q)}{|V(p) \cup V(q)|} = \frac{|V(p) \cap V(q)|}{|V(p) \cup V(q)|}$$

which is also known as Jaccard measure [15]. Note that this normalization is qualitatively different from the normalizations done for the previous distance measures. While the simple and the matched average distance are normalized by value which is the same for all paths in the same graph, here, the normalization is achieved by a value which is specific for the particular pair of paths.

EDGE SET SIMILARITY The same approach as the previous similarity measure can be done by counting the common number of edges instead of nodes. We therefore get the *edge set similarity* for two paths p and q by

$$\sigma_{ess}(p,q) := |E(p) \cap E(q)|$$

and the normalized similarity measure by

$$\sigma_{ess,N}(p,q) := \frac{\sigma_{ess}(p,q)}{|E(p) \cup E(q)|} = \frac{|E(p) \cap E(q)|}{|E(p) \cup E(q)|}$$

3.3.3 *Order based similarities*

The similarity measures described in the paragraph before consider paths as unordered sets of elements, though, in many cases, the *order* that the path imposes on the nodes is an essential information of the path. Hence, the order of the nodes or edges in the path should be the base for similarity measures for paths. Figure 8b shows the weakness of set based similarity measures. The two paths will have a very high node set similarity (the normalized node set similarity will even be 1). Though, the order of the nodes in the two paths is totally different. This path feature is not used at all in computing the similarity. For this reason, the following similarity measures will take the order of the nodes as most important feature to consider for the computation of path similarity.

LCSS SIMILARITY One measure that takes the order of the nodes in the paths into account is an adaption of the inclusion similarity for two strings, proposed by Laasonen [28, 27]. Laasonen introduces the inclusion similarity for two strings $s, t \in \Sigma^*$, Σ alphabet, with $|s| \geq |t|$ as $I(s,t) = \frac{T}{|t|}$ with T the number of elements of t which occur in the same order in s. However, this measure yields unexpected results because it is normalized by the length of the *shorter* string and not by the length of the longer one. If the two strings `abcdefgh` and `ah` are taken as an example, the inclusion similarity yields a similarity of 1 which does not fit to our understanding of similar strings, but does make sense in the context of the work of Laasonen. Therefore, we propose the following as a similarity measure which

considers the order of the nodes: Let $p, q \in \mathcal{P}_V$ with $|p| \geq |q|$. Then, the *LCSS similarity* (the name is explained below) is defined as

$$\sigma_{lcss}(p,q) := P(p,q)$$

with

$$\begin{aligned} P(p,q) := \max \Big\{ & |(v_{q_{i_1}} \dots v_{q_{i_n}})| \;\big|\; v_{q_{i_1}}, \dots, v_{q_{i_n}} \in V(q) \\ & \text{and } v_{q_{i_1}} \preceq_q \cdots \preceq_q v_{q_{i_n}} \\ & \text{and } \exists (v_{p_{j_1}} \dots v_{p_{j_n}}) \text{ with } v_{p_{j_1}}, \dots, v_{p_{j_n}} \in V(p) \\ & \text{and } v_{p_{j_1}} \preceq_p \cdots \preceq_p v_{p_{j_n}} \\ & \text{and } v_{q_{i_1}} = v_{p_{j_1}}, \dots, v_{q_{i_n}} = v_{p_{j_n}} \Big\} \end{aligned}$$

The similarity measure is normalized by the length of the longer path:

$$\sigma_{lcss,N}(p,q) := \frac{\sigma_{lcss}(p,q)}{\max\{|p|+1, |q|+1\}}$$

The two paths p and q in figure 8b would get similarity values of $\sigma_{lcss}(p,q) = 4$ and $\sigma_{lcss,N}(p,q) = \frac{2}{3}$.

This similarity measure is actually a well-known concept: If the two paths p and q are considered as strings in which the i-th letter represents the i-th node in the path, $P(p,q)$ is actually the length of the longest common subsequence of the two path strings. The longest common subsequence of two strings $a = a_1 a_2 \dots a_n \in \Sigma^*$ and $b = b_1 b_2 \dots b_l \in \Sigma^*$ for some alphabet Σ can be defined in the following way [13]: if there is a list of indices $i_1 < i_2 < \cdots < i_k$ with $k < n$, the subsequence specified by this indices list is $a_{i_1} a_{i_2} \dots a_{i_k}$. In particular, this means that the letters from a which occur in the subsequence need to be in the same order than in a, but do not need occur consecutively in a. `ac` for example is a subsequence of `abc`. For two strings a and b, a common subsequence is a subsequence that occurs in both a and b, the longest common subsequence (LCSS) is then the longest of all common subsequences. The length of the longest common subsequence of the paths p and q considered as strings is exactly the definition of $P(p,q)$. This is why we call this similarity measure LCSS similarity.

LONGEST COMMON SUBSTRING SIMILARITY The LCSS similarity looks for nodes in the paths which occur in the same order in both paths, but does not require that these nodes occur directly consecutively in the paths. Informally, the LCSS similarity allows gaps in the common node sequences. For cases in which this flexibility is not desired, but consecutive common sequences are the desired path feature which should be used for comput-

ing the similarity of paths, the longest common substring similarity is designed. The *longest common substring similarity* is defined as

$$\sigma_{str}(p,q) := \max\{\, j \in \mathbb{N} \mid v_{p_i} = v_{q_{i'}}, v_{p_{i+1}} = v_{q_{i'+1}}, \ldots, v_{p_{i+j}} = v_{q_{i'+j}}, \\ i \in \{1,\ldots,k\}, i' \in \{1,\ldots,m\}\} - 1$$

The normalization is the same as for the LCSS measure:

$$\sigma_{str,N}(p,q) := \frac{\sigma_{str}(p,q)}{\max\{|p|+1, |q|+1\}}$$

TUPLE SIMILARITY As an even more special measure that considers the order of the nodes of the paths, we propose the tuple similarity. It only makes sense in cases in which the two parts share a large proportion of their nodes (i.e. the normalized node set similarity of the paths is high). The tuple similarity counts how many node tuples, triples, quadruples, etc. the two paths have in common in the sense that the nodes in the tuples, triples, etc. occur in this order in both paths. Formally, let

$$\begin{aligned} T_1(p) &= \{v \in V(p)\} \\ T_2(p) &= \{(v,w) \in V(p) \times V(p) | v \preceq_p w\} \\ T_3(p) &= \{(v,w,x) \in V(p) \times V(p) \times V(p) | v \preceq_p w \preceq_p x\} \\ &\ldots \\ T_{|p|+1}(p) &= \{(v_1,\ldots v_k) \in V(p)^{|p|+1} | v_1 \preceq_p v_2 \preceq_p \cdots \preceq_p v_k\} \end{aligned}$$

be the sets of ordered tuples of nodes with a certain length that occur in this order in a considered path.

The similarity measure is then for two paths $p, q \in \mathcal{P}_V$ with $k-1 = |p| \geq |q|$

$$\sigma_{ts}(p,q) = \frac{1}{k} \sum_{i=1}^{k} |T_i(p) \cap T_i(q)|$$

The similarity sums up how many tuples of each length the two paths have in common.

The normalization for this measure needs to be different than for the other measures since it is strongly dependent on i how large the i-th summand can be. For all summands $|T_i(p) \cap T_i(q)|$, it holds that it takes its maximal value if the two paths are identical. The largest value $|T_i(p) \cap T_i(q)|$ can take is $\binom{k}{i}$, the number of possibilities of choosing i elements from a set with k objects, i.e. the number of subsets of size i of a set with size k. Hence, we propose to normalize each summand separately and sum up the normalized summands:

$$\sigma_{ts,N}(p,q) := \frac{1}{k} \sum_{i=1}^{k} \frac{|T_i(p) \cap T_i(q)|}{\binom{k}{i}}$$

For an example, consider the paths shown in figure 5c. Let $p = (v_1v_2v_3v_4)$ be the green path and $q = (v_3v_2v_4v_1)$ be the blue path. It is $|p| = |q| = 3$. The tuple sets are then

$$\begin{aligned}
T_1(p) &= \{v_1, v_2, v_3, v_4\} \\
T_1(q) &= \{v_1, v_2, v_3, v_4\} \\
T_2(p) &= \{(v_1, v_2), (v_1, v_3), (v_1, v_4), (v_2, v_3), (v_2, v_4), (v_3, v_4)\} \\
T_2(q) &= \{(v_3, v_2), (v_3, v_4), (v_3, v_1), (v_2, v_4), (v_2, v_1), (v_4, v_1)\} \\
T_3(p) &= \{(v_1, v_2, v_3), (v_1, v_2, v_4), (v_1, v_3, v_4), (v_2, v_3, v_4)\} \\
T_3(q) &= \{(v_3, v_2, v_4), (v_3, v_2, v_1), (v_3, v_4, v_1), (v_2, v_4, v_1)\} \\
T_4(p) &= \{(v_1, v_2, v_3, v_4)\} \\
T_4(q) &= \{(v_3, v_2, v_4, v_1)\}
\end{aligned}$$

and therefore

$$\begin{aligned}
\sigma_{ts}(p,q) &= \frac{1}{4} \cdot (|T_1(p) \cap T_1(q)| + |T_2(p) \cap T_2(q)| \\
&\quad + |T_3(p) \cap T_3(q)| + |T_4(p) \cap T_4(q)|) \\
&= \frac{1}{4} \cdot (4 + 2 + 0 + 0) = \frac{3}{2}
\end{aligned}$$

and

$$\begin{aligned}
\sigma_{ts,N}(p,q) &= \frac{1}{4} \cdot \left(\frac{|T_1(p) \cap T_1(q)|}{\binom{4}{1}} + \frac{|T_2(p) \cap T_2(q)|}{\binom{4}{2}} \right. \\
&\quad \left. + \frac{|T_3(p) \cap T_3(q)|}{\binom{4}{3}} + \frac{|T_4(p) \cap T_4(q)|}{\binom{4}{4}} \right) \\
&= \frac{1}{4} \cdot \left(\frac{4}{4} + \frac{2}{6} + \frac{0}{4} + \frac{0}{1} \right) = \frac{1}{3}
\end{aligned}$$

It also might be appropriate to choose a subset of tuple sets which are used to compute the tuple similarity. Depending on the application scenario, it might be more important that many short subsequences are common for both paths or a few long ones or any value between. Choosing a subset will also reduce the computation time.

EDIT DISTANCE A widely used distance measure for objects, strings and even sequences (for example see [36]) is the edit distance which measures the minimal costs of operations to transform one object into another, given a set of allowed transformation operations and their respective costs. The usually allowed operations are inserting, deleting and substituting or moving parts of the object. For paths, this approach is also applicable, but the allowed operations should be carefully chosen as well as the conditions which should hold during the transformation process. One possibility is to allow the classic operations of inserting, deleting and substituting nodes in the paths and to only require that the resulting node sequence is a valid path in the given path, i.e. to allow that node sequences which occur during the transformation process, are no real paths in the given graph. Then,

for a given node sequence $p = (v_{p_1} \dots v_{p_k})$ and a node $v \in V$, we define the edit operations

$$\begin{aligned} insert(p, v, i) &= (v_{p_1} v_{p_2} \dots v_{p_{i-1}} v v_i v_{p_{i+1}} \dots v_{p_k}) \\ delete(p, v_{p_j}) &= (v_{p_1} \dots v_{p_{j-1}} v_{p_{j+1}} \dots v_{p_k}) \\ substitute(p, v_{p_j}, v) &= (v_{p_1} \dots v_{p_{j-1}} v v_{p_{j+1}} \dots v_{p_k}) \end{aligned}$$

A node sequence p can then be transformed into another node sequence q by a sequence of edit operations $(e_1 \dots e_l)$ on p if $e_l(e_{l-1}(\dots e_1(p, \cdot, \cdot), \cdot, \cdot) \dots), \cdot, \cdot) = q$ yields q, whereas $e_1, \dots, e_l \in \{insert, delete, substitute\}$. For a better readability, the edit operations in the given transformation process all have placeholders for two arguments. If the respective edit operations is a *delete*-operation, the edit operation evidently only has one argument.

A cost function c assigns each edit operations a value which represents the effort that is needed to perform the modification. The distance measure for two paths is then defined as

$$\delta_{ed}(p, q) := \min \left\{ \sum_{i=1}^{l} c(e_i) \middle| e_l \left(e_{l-1} \left(\dots \left(e_1 \left(p, \cdot, \cdot \right), \cdot, \cdot \right) \dots \right), \cdot, \cdot \right) = q \right\}$$

A possible normalization could be done by the maximal value δ_{ed} can take for the two paths which is dependent on the design of the cost function. In simple cases, a trivial upper bound is the costs of deleting all nodes from p and inserting all nodes from q or – if substituting a node is cheaper than deleting and inserting one – the costs of substituting all nodes from p by nodes from q (and delete delete remaining nodes or insert missing nodes). Then, for the first case,

$$\delta_{ed,N}(p, q) := \frac{\delta_{ed}(p, q)}{\sum_{i=1}^{k} c(delete(p_i, v_{p_i})) + \sum_{i=1}^{m} c(insert(p_i, v_{q_i}, i))}$$

whereas p_i is the intermediate node sequence during the transformation process.

Since the cost function can also take the parameters of the edit operation as arguments and the costs for insertion, deletion and substitution can be arbitrarily different, the proposed normalization might be inappropriate in certain cases. It should be chosen according to the characteristics of the selected cost function and application domain.

For the context of paths in graphs, it might be reasonable to require that all intermediate node sequences during the transformation process are valid paths in the given graph, as well. This requirement will potentially increase the edit costs for paths. Though if the edit distance is thought of being a model for an actual transformation process, it is natural to impose the constraint that only valid intermediate stages can be used.

MEASURE	NAME	NORMALIZATION	RANGE
$\delta_{Hausdorff}$	Hausdorff distance		$\{0,\dots,diam(G)\} \subseteq \mathbb{N}$
$\delta_{Hausdorff,N}$	normalized Hausdorff distance	$diam(G)$	$[0,1] \subseteq \mathbb{R}$
δ_{sad}	simple average distance		$[0,diam(G)] \subseteq \mathbb{R}$
$\delta_{sad,N}$	normalized simple average distance	$diam(G)$	$[0,1] \subseteq \mathbb{R}$
δ_{mad}	matched average distance		$[0,diam(G)] \subseteq \mathbb{R}$
$\delta_{mad,N}$	normalized matched average distance	$diam(G)$	$[0,1] \subseteq \mathbb{R}$
σ_{nss}	node set similarity		$\{0,\dots,\lvert V\rvert\} \subseteq \mathbb{N}$
$\sigma_{nss,N}$	normalized node set similarity	$\lvert V(p) \cup V(q)\rvert$	$[0,1] \subseteq \mathbb{R}$
σ_{ess}	edge set similarity		$\{0,\dots,\lvert E\rvert\} \subseteq \mathbb{N}$
$\sigma_{ess,N}$	normalized edge set similarity	$\lvert E(p) \cup E(q)\rvert$	$[0,1] \subseteq \mathbb{R}$
σ_{lcss}	longest common subsequence similarity		$\{0,\dots,\min\{\lvert p\rvert+1,\lvert q\rvert+1\}\} \subseteq \mathbb{N}$
$\sigma_{lcss,N}$	normalized LCSS	$\max\{\lvert p\rvert+1,\lvert q\rvert+1\}$	$[0,1] \subseteq \mathbb{R}$
σ_{str}	longest common substring similarity		$\{0,\dots,\min\{\lvert p\rvert+1,\lvert q\rvert+1\}\} \subseteq \mathbb{N}$
$\sigma_{str,N}$	normalized longest common substring similarity	$\max\{\lvert p\rvert+1,\lvert q\rvert+1\}$	$[0,1] \subseteq \mathbb{R}$
σ_{ts}	tuple similarity		$[0,\sum_{i=1}^{k} \frac{(k-1)!}{i!(k-i)!}] \subseteq \mathbb{R}$ for $k := \max\{\lvert p\rvert+1,\lvert q\rvert+1\}$
$\sigma_{ts,N}$	normalized tuple similarity	each summand by $\binom{k}{i}$	$[0,1] \subseteq \mathbb{R}$
δ_{ed}	edit distance		$\mathbb{R}$ (depends on cost function)
$\delta_{ed,N}$	normalized edit distance	depends on cost function	$[0,1]$

Table 1: An overview of the introduced similarity and distance measures.

3.3.4 *Further ideas*

PARAMETRIZED MEASURES In their work about the similarity of trajectories of objects moving in a two- or three-dimensional space, Vlachos et al. [44] propose the longest common subsequence measure to compare to trajectories. Though, they introduce their similarity measure as parametrized measure with two parameters δ and ϵ. The parameter ϵ indicates how close in space two points of the trajectories need to be in order that they are considered as the same and can contribute to a common subsequence. The parameter δ controls how close in time the two points need to be such that they can be considered as a match.

The further details of their approach are not of interest here, but the idea of parametrized similarity or distance measures is also an interesting one for paths in graphs. For example, there could be a parameter which determines which distance two nodes in graph are allowed to have such that they are considered as the same and can contribute to a similarity or distance measure. This would make the measures more flexible in the sense that two nodes do not necessarily need to be exactly the same in order to be a match, for example to increase the node set similarity. If two nodes are neighbors, i.e. have a distance of 1, it might be reasonable to count them as match or as common node, at least more than two distant nodes.

For almost each of the introduced similarity and distance measures, there is a parametrization which might make sense in certain application areas, but this approach needs to evaluated further in future work.

3.4 PROPERTIES OF THE PROPOSED MEASURES

This section reviews the similarity and distance measures proposed in section 3.3 for their properties: for all proposed measures, it is checked whether they satisfy the properties introduced in section 3.4. An overview of the results can be found in table 2. The proofs and counterexamples for the entries in the table are given in the following paragraphs. Some of the entries are marked with a question mark in order to make clear that we did not find a proof or counterexample for this pair of measure and property.

3.4.1 *Simple Average Distance*

PREFIX AND SUFFIX CONSISTENCY δ_{sad} and $\delta_{sad,N}$ satisfy prefix and suffix consistency.

	(i)	(ii)	(iii)	(iv)	(v)	(vi)	(vii)	(viii)	(ix)
Simple average distance									
unnormalized	✓	✗	✓	✗	✓	✓	✓	✓	✓
normalized	✓	✗	✓	✗	✓	✓	✓	✓	✓
Matched average distance									
unnormalized	✓	✓	?	✗	✓	✓	✗	✓	✗
normalized	✓	✓	?	✗	✓	✓	✗	✓	✗
Node set similarity									
unnormalized	✓	✓	✓	✗	✓	✓	✗	✓	✓
normalized	?	?	?	✗	✓	✓	✗	✓	?
Edge set similarity									
unnormalized	✓	✗	✓	✓	✓	✓	✗	✓	✓
normalized	?	✗	?	?	✓	✓	✗	✓	?
LCSS similarity									
unnormalized	✓	✓	✓	✗	✗	✓	✗	✓	✗
normalized	✓	✓	?	✗	✓	✓	✓	✓	?

Table 2: The properties have the enumeration as in section 3.1:

(i) prefix and suffix consistency
(ii) insertion consistency
(iii) concatenation consistency
(iv) edge imbalance consistency
(v) boundedness
(vi) non-negativity
(vii) coincidence
(viii) symmetry
(ix) triangle inequality

Proof. We only show the suffix consistency, the proof for prefix consistency is similar. For given paths p, q, r with $|p| = |q| = k - 1$ and $|r| = l - 1$, it holds

$$\begin{aligned}
\delta_{sad}(p,q) &= \frac{1}{k}\sum_{i=1}^{k} d(v_{p_i}, v_{q_i}) \\
&= \frac{1}{k}\left(\sum_{i=1}^{k} d(v_{p_i}, v_{q_i}) + \sum_{i=1}^{l} 0\right) \\
&= \frac{1}{k}\left(\sum_{i=1}^{k} d(v_{p_i}, v_{q_i}) + \sum_{i=1}^{l} d(v_{r_i}, v_{r_i})\right) \text{ since } d(v,v) = 0 \; \forall v \in V \\
&\geq \frac{1}{k+l}\left(\sum_{i=1}^{k} d(v_{p_i}, v_{q_i}) + \sum_{i=1}^{l} d(v_{r_i}, v_{r_i})\right) \\
&= \delta_{sad}(p \oplus r, q \oplus r)
\end{aligned}$$

For the normalized simple average distance, it also holds

$$\begin{aligned}
\delta_{sad,N}(p,q) &= \frac{1}{diam(G)} \cdot \frac{1}{k}\sum_{i=1}^{k} d(v_{p_i}, v_{q_i}) \\
&\geq \frac{1}{diam(G)} \cdot \delta_{sad}(p \oplus r, q \oplus r) \\
&= \delta_{sad,N}(p \oplus r, q \oplus r)
\end{aligned}$$

□

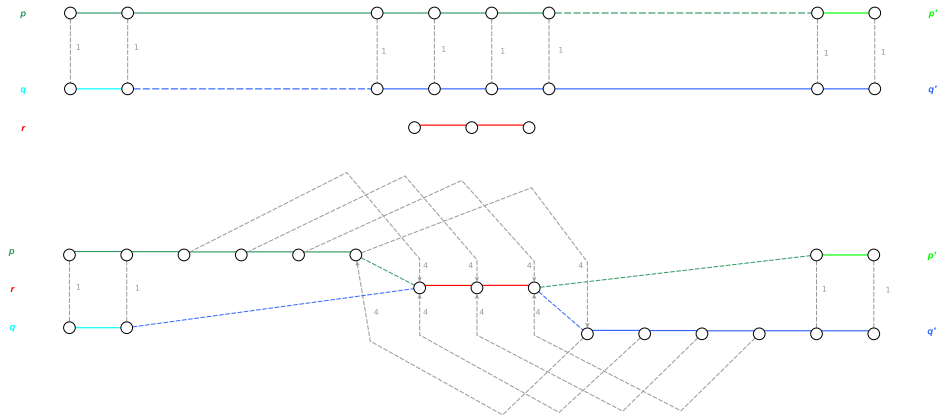
INSERTION CONSISTENCY δ_{sad} and $\delta_{sad,N}$ do not satisfy insertion consistency.

Proof. The reason why insertion consistency does not hold for the simple average distance is that insertion consistency does not require that the common subpath of the two paths is at the same position for both paths. Consider the example shown in figure 6a. The paths p, p', q, q' are drawn in (light) blue and green. The path r which is inserted in p and q is drawn in red. The gray edges with label depict the distance of the nodes in the graph and how the nodes of p and q are mapped onto each other. Therefore, the simple average distance for the paths without the common subpath gives

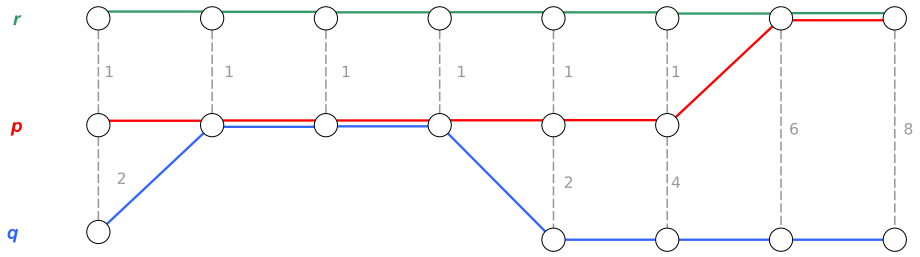
$$\delta_{sad}(p \oplus p', q \oplus q') = \frac{1}{8}(1+1+1+1+1+1+1+1) = 1.$$

If then r is inserted between p and p' and between q and q', respectively, the nodes are matched in a way that considerably increases the simple average distance although the same subpath is inserted in both paths, as it happens in figure 6a:

$$\begin{aligned}
&\delta_{sad}(p \oplus r \oplus p', q \oplus r \oplus q') \\
&= \frac{1}{11} \cdot (1+1+4+4+4+4+4+4+4+4+1+1) \\
&= \frac{32}{11} > 1 = \delta_{sad}(p \oplus p', q \oplus q')
\end{aligned}$$



(a) An example in which the simple average distance does not satisfy insertion consistency.



(b) An example in which the simple average distance does not satisfy edge imbalance consistency.

Figure 6: Examples for properties of the simple average distance

The same example works for the normalized simple average distance:

$$\delta_{sad,N}(p \oplus p', q \oplus q') = \frac{1}{diam(G)} < \frac{32}{11 \cdot diam(G)} = \delta_{sad,N}(p \oplus r \oplus p', q \oplus r \oplus q')$$

for any $diam(G) > 0$. □

CONCATENATION CONSISTENCY δ_{sad} and $\delta_{sad,N}$ satisfy concatenation consistency.

Proof. For paths p, p', q, q' with $|p| = |q| = k-1$ and $|p'| = |q'| = k'-1$, it holds

$$\begin{aligned}
&\delta_{sad}(p \oplus p', q \oplus q') \\
&= \frac{1}{k+k'} \cdot \left(\sum_{i=1}^{k} d(v_{p_i}, v_{q_i}) + \sum_{j=1}^{k'} d(v'_{p_j}, v'_{q_j}) \right) \\
&= \frac{k \cdot \delta_{sad}(p,q) + k' \cdot \delta_{sad}(p',q')}{k+k'} \\
&\leq \frac{k \cdot \max\{\delta_{sad}(p,q), \delta_{sad}(p',q')\} + k' \cdot \max\{\delta_{sad}(p,q), \delta_{sad}(p',q')\}}{k+k'} \\
&= \frac{1}{k+k'} \cdot (k+k') \cdot \max\{\delta_{sad}(p,q), \delta_{sad}(p',q')\} \\
&= \max\{\delta_{sad}(p,q), \delta_{sad}(p',q')\}.
\end{aligned}$$

This result can be transferred to the normalized simple average distance

$$\begin{aligned}
\delta_{sad,N}(p \oplus p', q \oplus q') &= \frac{\delta_{sad}(p \oplus p', q \oplus q')}{diam(G)} \\
&\leq \frac{\max\{\delta_{sad}(p,q), \delta_{sad}(p',q')\}}{diam(G)} \\
&= \max\left\{ \frac{\delta_{sad}(p,q)}{diam(G)}, \frac{\delta_{sad}(p',q')}{diam(G)} \right\} \\
&= \max\{\delta_{sad,N}(p,q), \delta_{sad,N}(p',q')\}.
\end{aligned}$$

□

EDGE IMBALANCE CONSISTENCY δ_{sad} and $\delta_{sad,N}$ do not satisfy edge imbalance consistency.

Proof. Consider the paths p, q, r in figure 6b. The three paths have all equal length, therefore, the simple average distance can be computed. It holds

$$|E(p) \cap E(q)| = 2$$

and

$$|E(p) \cap E(r)| = 1,$$

therefore,

$$|E(p) \cap E(q)| > |E(p) \cap E(r)|.$$

But

$$\delta_{sad}(p,r) = \frac{1}{8}(1+1+1+1+1+1+0+0) = \frac{3}{4}$$

and

$$\delta_{sad}(p,q) = \frac{1}{8}(2+0+0+0+2+4+6+8) = \frac{11}{4}$$

and therefore

$$\delta_{sad}(p,q) > \delta_{sad}(p,r).$$

The same counterexample also holds for $\delta_{sad,N}$. □

BOUNDEDNESS δ_{sad} and $\delta_{sad,N}$ satisfy $diam(G)$-boundedness respectively 1-boundedness.

Proof. By definition of the diameter of a graph G, for any two nodes $v, w \in V$, it holds $d(v,w) \leq diam(G)$. Then, for any paths $p, q \in \mathcal{P}_V$ in a graph G with $|p| = |q| = k-1$, it holds

$$\begin{aligned}\delta_{sad}(p,q) &= \frac{1}{k}\sum_{i=1}^{k} d(v_{p_i}, v_{q_i}) \leq \frac{1}{k}\sum_{i=1}^{k} \max\{d(v,w) | v,w \in V\} \\ &= \frac{1}{k}\sum_{i=1}^{k} diam(G) = \frac{k \cdot diam(G)}{k} = diam(G)\end{aligned}$$

Since

$$\delta_{sad,N}(p,q) = \frac{\delta_{sad}(p,q)}{diam(G)} \leq \frac{diam(G)}{diam(G)} = 1,$$

$\delta_{sad,N}$ is 1-bounded. δ_{sad} and $\delta_{sad,N}$ also satisfy strong $diam(G)$-boundedness and 1-boundedness, respectively: let v and w the nodes for which $d(v,w) = diam(G)$, then for $p = (v)$ and $q = (w)$,

$$\delta_{sad}(p,q) = diam(G)$$

and

$$\delta_{sad,N}(p,q) = 1.$$

□

NON-NEGATIVITY δ_{sad} and $\delta_{sad,N}$ satisfy non-negativity.

Proof. For any two nodes $v, w \in V$ in an unweighted graph G, $d(v,w) \geq 0$ holds. Hence, for any paths p, q, $\delta_{sad}(p,q) \geq 0$ and $\delta_{sad,N}(p,q) \geq 0$ is satisfied. If δ_{sad} is applied on graphs with possibly negative edge weights, this property is not necessarily satisfied anymore. □

COINCIDENCE δ_{sad} and $\delta_{sad,N}$ satisfy coincidence.

Proof. Let for any paths $p, q \in \mathcal{P}_V$

$$\begin{aligned}&\delta_{sad}(p,q) = 0 \\ &\Leftrightarrow \delta_{sad}(p,q) = \frac{1}{k}\sum_{i=1}^{k} d(v_{p_i}, v_{q_i}) = 0 \\ &\Leftrightarrow d(v_{p_i}, v_{q_i}) = 0 \; \forall i \in \{1,\ldots,k\} \text{ since } d(v,w) \geq 0 \; \forall v,w \in V \\ &\Leftrightarrow v_{p_i} = v_{q_i} \forall \; i \in \{1,\ldots,k\} \\ &\Leftrightarrow p = q.\end{aligned}$$

Note that we consider the paths p and $inv(p)$ as not equal which is consistent with the definitions of δ_{sad} and the coincidence property. The proof for $\delta_{sad,N}$ is similar. □

SYMMETRY δ_{sad} and $\delta_{sad,N}$ satisfy symmetry.

Proof. In undirected graphs, the distance of two nodes is symmetric, i.e. for any two nodes $v, w \in V$, it holds $d(v,w) = d(w,v)$. Therefore,

$$\begin{aligned}\delta_{sad}(p,q) &= \frac{1}{k}\sum_{i=1}^{k} d(v_{p_i}, v_{q_i}) \\ &= \frac{1}{k}\sum_{i=1}^{k} d(v_{q_i}, v_{p_i}) \\ &= \delta_{sad}(q,p)\end{aligned}$$

and similarly for $\delta_{sad,N}$. If this measure is applied on directed graphs, symmetry is not necessarily given anymore. □

TRIANGLE INEQUALITY δ_{sad} and $\delta_{sad,N}$ satisfy the triangle inequality.

Proof. We use that the distance of two nodes in a graph satisfies the triangle inequality: in a connected, simple, undirected graph $G = (V,E)$, for any nodes $v, w, x \in V$, it holds $d(v,x) \leq d(v,w) + d(w,x)$. Therefore, it holds

$$\begin{aligned}& d(v_{p_i}, v_{q_i}) \leq d(v_{p_i}, v_{r_i}) + d(v_{r_i}, v_{q_i}) \quad \forall i \in \{1,\dots,k\} \\ \Leftrightarrow\ & 0 \leq d(v_{p_i}, v_{r_i}) + d(v_{r_i}, v_{q_i}) - d(v_{p_i}, v_{q_i}) \quad \forall i \in \{1,\dots,k\} \\ \Leftrightarrow\ & 0 \leq \sum_{i=1}^{k} \left(d(v_{p_i}, v_{r_i}) + d(v_{r_i}, v_{q_i}) - d(v_{p_i}, v_{q_i})\right) \\ \Leftrightarrow\ & 0 \leq \sum_{i=1}^{k} d(v_{p_i}, v_{r_i}) + \sum_{i=1}^{k} d(v_{r_i}, v_{q_i}) - \sum_{i=1}^{k} d(v_{p_i}, v_{q_i}) \\ \Leftrightarrow\ & \sum_{i=1}^{k} d(v_{p_i}, v_{q_i}) \leq \sum_{i=1}^{k} d(v_{p_i}, v_{r_i}) + \sum_{i=1}^{k} d(v_{r_i}, v_{q_i}) \\ \Leftrightarrow\ & \frac{1}{k} \cdot \sum_{i=1}^{k} d(v_{p_i}, v_{q_i}) \leq \frac{1}{k} \cdot \sum_{i=1}^{k} d(v_{p_i}, v_{r_i}) + \frac{1}{k} \cdot \sum_{i=1}^{k} d(v_{r_i}, v_{q_i}) \\ \Leftrightarrow\ & \delta_{sad}(p,q) \leq \delta_{sad}(p,r) + \delta_{sad}(r,q)\end{aligned}$$

Since

$$\begin{aligned}& \delta_{sad}(p,q) \leq \delta_{sad}(p,r) + \delta_{sad}(r,q) \\ \Leftrightarrow\ & \frac{1}{diam(G)} \cdot \delta_{sad}(p,q) \leq \frac{1}{diam(G)} \cdot \delta_{sad}(p,r) + \frac{1}{diam(G)} \cdot \delta_{sad}(r,q) \\ \Leftrightarrow\ & \delta_{sad,N}(p,q) \leq \delta_{sad,N}(p,r) + \delta_{sad,N}(r,q)\end{aligned}$$

holds, the triangle inequality also holds for the normalized simple average distance. □

FURTHER PROPERTIES The idea of the simple average distance is a rather naive one since it measures how distant the single nodes of the two paths are from each other on average. The rule which node of the one path is compared to which node of the other path, is very simple and inflexible. Because the i-th

node of the first path is matched onto the i-th node of the other path, the applicability of the simple average distance is constrained to paths of equal length and its computation can result in an unintuitively high value because the strict mapping rule for the nodes might be inappropriate for certain paths. For example, it might be unexpected that the average distance of a path p and its inverse $inv(p)$ is never 0, but rather quite high. Furthermore, as it was shown in the counterexample for insertion consistency, the simple average distance is very sensitive to changes in the paths. Even very small changes in the paths (insertions or deletions of nodes) can considerably change the value of the simple average distance because the mapping of the nodes can be a completely different one.

On the other hand, this distance measure is very easy to compute, there is no need to compute a computationally costly mapping of path nodes, only $|p|+1 = |q|+1$ node distances need to be computed. In addition to that, the simple average distance is a distance metric and satisfies even some more of the checked properties.

3.4.2 *Matched Average Distance*

PREFIX AND SUFFIX CONSISTENCY δ_{mad} and $\delta_{mad,N}$ satisfy prefix and suffix consistency.

Proof. We only prove that δ_{mad} satisfies suffix consistency, the other cases are similar. Let $|p| = k-1 \geq |q| = m-1$ and $|r| = l-1$. Furthermore, let the mappings $g : V(p) \to V(q)$ and $g' : V(p) \cup V(r) \to V(q) \cup V(r)$ be as proposed in section 3.3, then,

$$\begin{aligned}
\delta_{mad}(p \oplus r, q \oplus r) &= \frac{1}{k+l}\left(\sum_{i=1}^{k} d\left(v_{p_i}, g'\left(v_{p_i}\right)\right) + \underbrace{\sum_{i=1}^{l} d\left(v_{r_i}, g'\left(v_{r_i}\right)\right)}_{\substack{=0 \\ \text{since } d\left(v_{r_i}, g'\left(v_{r_i}\right)\right)=0 \\ \forall\, i \in \{1,\ldots,k\}}}\right) \\
&= \frac{1}{k+l}\sum_{i=1}^{k} d\left(v_{p_i}, g\left(v_{p_i}\right)\right) \\
&\leq \frac{1}{k}\sum_{i=1}^{k} d\left(v_{p_i}, g\left(v_{p_i}\right)\right) \\
&= \delta_{mad}(p, q).
\end{aligned}$$

For the normalized matched average distance, it holds

$$\begin{aligned}
\delta_{mad,N}(p \oplus r, q \oplus r) &= \frac{\delta_{mad}(p \oplus r, q \oplus r)}{diam(G)} \\
&\leq \frac{\delta_{mad}(p, q)}{diam(G)} \\
&= \delta_{mad,N}(p, q).
\end{aligned}$$

□

INSERTION CONSISTENCY δ_{mad} and $\delta_{mad,N}$ satisfy insertion consistency.

Proof. Let $p, q, p', q', r \in \mathcal{P}_V$ be paths which can be concatenated to the paths $p \oplus p', q \oplus q', p \oplus r \oplus p'$ and $q \oplus r \oplus q'$, i.e. $\{(\omega_p, \alpha_{p'}), (\omega_q, \alpha_{q'}), (\omega_p, \alpha_r), (\omega_q, \alpha_r), (\omega_r, \alpha_{p'}), (\omega_r, \alpha_{q'})\} \subseteq E$ and $|p| = k-1$, $|p'| = k'-1$, $|q| = m-1$, $|q'| = m'-1$ and $|r| = l-1$. Let *wlog* $|p| + |p'| \geq |q| + |q'|$ and $g : V(p) \cup V(r) \cup V(p') \to V(q) \cup V(r) \cup V(q')$ and $g' : V(p) \cup V(p') \to V(q) \cup V(q')$ as proposed in section 3.3. Then

$$
\begin{aligned}
&\delta_{mad}(p \oplus r \oplus p', q \oplus r \oplus q') \\
&= \frac{1}{k+l+k'} \left(\sum_{i=1}^{k} d(v_{p_i}, g(v_{p_i})) + \underbrace{\sum_{i=1}^{l} d(v_{r_i}, g(v_{r_i}))}_{=0} + \sum_{i=1}^{k'} d\left(v_{p'_i}, g\left(v_{p'_i}\right)\right) \right) \\
&= \frac{1}{k+l+k'} \left(\sum_{i=1}^{k} d(v_{p_i}, g(v_{p_i})) + \sum_{i=1}^{k'} d\left(v_{p'_i}, g\left(v_{p'_i}\right)\right) \right) \\
&\leq \frac{1}{k+l+k'} \left(\sum_{i=1}^{k} d(v_{p_i}, g'(v_{p_i})) + \sum_{i=1}^{k'} d\left(v_{p'_i}, g'\left(v_{p'_i}\right)\right) \right) \\
&\leq \frac{1}{k+k'} \left(\sum_{i=1}^{k} d(v_{p_i}, g'(v_{p_i})) + \sum_{i=1}^{k'} d\left(v_{p'_i}, g'\left(v_{p'_i}\right)\right) \right) \\
&= \delta_{mad}(p \oplus p', q \oplus q')
\end{aligned}
$$

For the normalized matched average distance, it holds

$$
\begin{aligned}
\delta_{mad,N}(p \oplus r \oplus p', q \oplus r \oplus q') &= \frac{\delta_{mad,N}(p \oplus r \oplus p', q \oplus r \oplus q')}{diam(G)} \\
&\leq \frac{\delta_{mad}(p \oplus p', q \oplus q')}{diam(G)} \\
&= \delta_{mad,N}(p \oplus p', q \oplus q')
\end{aligned}
$$

□

EDGE IMBALANCE CONSISTENCY δ_{mad} and $\delta_{mad,N}$ do not satisfy edge imbalance consistency.

Proof. Consider the example in figure 7a. It holds

$$|E(p) \cap E(q)| = 3 > 1 = |E(p) \cap E(r)|$$

and

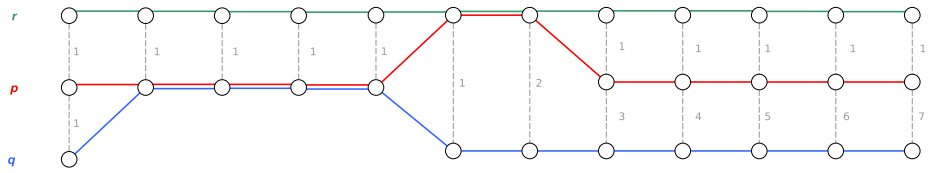
$$|p| = |q| = |r|,$$

but

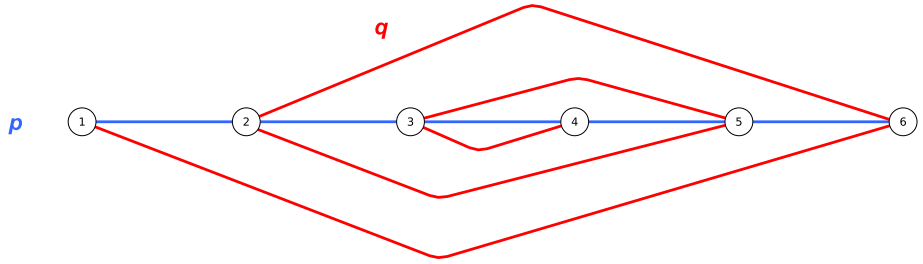
$$\delta_{mad}(p, q) = \frac{1}{12}(1+0+0+0+0+1+2+3+4+5+6+7) = \frac{29}{12}$$

and

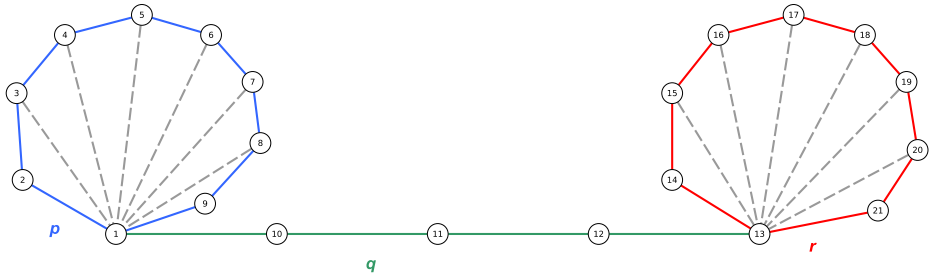
$$\delta_{mad}(p, r) = \frac{1}{12}(1+1+1+1+1+0+0+1+1+1+1+1) = \frac{5}{6}$$



(a) An example in which the matched average distance does not satisfy the edge imbalance.



(b) An example in which the matched average distance does not satisfy the coincidence.



(c) An example in which the matched average distance does not satisfy the triangle inequality.

Figure 7: Examples for properties of the matched average distance.

and therefore

$$\delta_{mad}(p,q) > \delta_{mad}(p,r).$$

The same counterexample holds for $\delta_{mad,N}$. □

BOUNDEDNESS δ_{mad} and $\delta_{mad,N}$ satisfy $diam(G)$-boundedness and 1-boundedness, respectively.

Proof. By definition of the diameter of a graph G, for any two nodes $v, w \in V$, it holds $d(v,w) \leq diam(G)$. Then, for any paths $p, q \in \mathcal{P}_V$ in a graph G, it holds

$$\begin{aligned}\delta_{mad}(p,q) &= \frac{1}{k}\sum_{i=1}^{k} d\left(v_{p_i}, g\left(v_{q_i}\right)\right) \leq \frac{1}{k}\sum_{i=1}^{k} \max\left\{d(v,w) | v,w \in V\right\} \\ &= \frac{1}{k}\sum_{i=1}^{k} diam(G) = \frac{k \cdot diam(G)}{k} = diam(G)\end{aligned}$$

Since

$$\delta_{mad,N}(p,q) = \frac{\delta_{mad}(p,q)}{diam(G)} \leq \frac{diam(G)}{diam(G)} = 1,$$

$\delta_{mad,N}$ is 1-bounded.

With the same example as for the simple average distance, δ_{mad} and $\delta_{mad,N}$ satisfy strong $diam(G)$-boundedness and 1-boundedness, respectively: Let v and w be the nodes with $d(v,w) = diam(G)$, then for $p = (v)$ and $q = (w)$,

$$\delta_{mad}(p,q) = diam(G)$$

and

$$\delta_{mad,N}(p,q) = 1.$$

□

NON-NEGATIVITY δ_{mad} and $\delta_{mad,N}$ satisfy non-negativity.

Proof. Since for any two nodes $v, w \in V$ in an unweighted graph G, $d(v,w) \geq 0$, for any paths p, q, it holds $\delta_{mad}(p,q) \geq 0$ and $\delta_{mad,N}(p,q) \geq 0$. □

COINCIDENCE δ_{mad} and δ_{mad} do not satisfy coincidence.

Proof. Consider the paths in figure 7b. Let $p = (v_1, v_2, v_3, v_4, v_5, v_6)$ and $q = (v_4, v_3, v_5, v_2, v_6, v_1)$ and therefore $p \neq q$. Let $g : V(p) \to V(q)$ and $g' : V(q) \to V(p)$ as proposed in section 3.3. Then

$$\begin{aligned}
\delta_{mad}(p,q) = & \min\left\{\frac{1}{|p|+1}\sum_{i=1}^{|p|+1} d\left(v_{p_i}, g\left(v_{p_i}\right)\right), \quad \frac{1}{|q|+1}\sum_{i=1}^{|q|+1} d\left(v_{q_i}, g\left(v_{q_i}\right)\right)\right\} \\
= & \frac{1}{6}\cdot \min\{d(v_1, g(v_1)) + d(v_2, g(v_2)) + d(v_3, g(v_3)) + \\
& d(v_4, g(v_4)) + d(v_5, g(v_5)) + d(v_6, g(v_6)), \\
& d\left(v_4, g'(v_4)\right) + d\left(v_3, g'(v_3)\right) + d\left(v_5, g'(v_5)\right) + \\
& d\left(v_2, g'(v_2)\right) + d\left(v_6, g'(v_6)\right) + d\left(v_1, g'(v_1)\right)\} \\
= & \frac{1}{6}\cdot \min\{d(v_1, v_1) + d(v_2, v_2) + d(v_3, v_3) + \\
& d(v_4, v_4) + d(v_5, v_5) + d(v_6, v_6), \\
& d(v_4, v_4) + d(v_3, v_3) + d(v_5, v_5) + \\
& d(v_2, v_2) + d(v_6, v_6) + d(v_1, v_1)\} \\
= & \frac{1}{6}\cdot \min\{0 + \cdots + 0, 0 + \cdots + 0\} = 0
\end{aligned}$$

although $p \neq q$.

The same counterexample works for the normalized measure $\delta_{mad,N}$. □

SYMMETRY δ_{mad} and $\delta_{mad,N}$ satisfy symmetry.

Proof. This directly follows from the definition of δ_{mad} and $\delta_{mad,N}$. □

TRIANGLE INEQUALITY δ_{mad} and $\delta_{mad,N}$ do not satisfy the triangle inequality.

Proof. Consider the paths in figure 7c, i.e. the paths $p = (v_1, v_2, v_3, v_4, v_5, v_6, v_7, v_8, v_9, v_1)$, $q = (v_1, v_{10}, v_{11}, v_{12}, v_{13})$ and $r = (v_{13}, v_{14}, v_{15}, v_{16}, v_{17}, v_{18}, v_{19}, v_{20}, v_{21}, v_{13})$. The dashed gray edges represent further edges in the underlying graph. Furthermore, let $g : V(p) \to V(q)$ and $g' : V(p) \to V(r)$. It holds

$$\begin{aligned}
\delta_{mad}(p,q) = & \frac{1}{9}\cdot (d(v_1, g(v_1)) + d(v_2, g(v_2)) + d(v_3, g(v_3)) + \\
& d(v_4, g(v_4)) + d(v_5, g(v_5)) + d(v_6, g(v_6)) + d(v_7, g(v_7)) + \\
& d(v_8, g(v_8)) + d(v_9, g(v_9)) + d(v_1, g(v_1))) \\
= & \frac{1}{9}\cdot (d(v_1, v_1) + d(v_2, v_1) + d(v_3, v_1) + d(v_4, v_1) + d(v_5, v_1) + \\
& d(v_6, v_1) + d(v_7, v_1) + d(v_8, v_1) + d(v_9, v_1) + d(v_1, v_1)) \\
= & \frac{1}{9}\cdot (0 + 1 + 1 + 1 + 1 + 1 + 1 + 1 + 1 + 0) \\
= & \frac{8}{9}
\end{aligned}$$

and

$$\delta_{mad}(q,r) = \delta_{mad}(r,q) = \frac{8}{9},$$

but

$$\begin{aligned}
\delta_{mad}(p,r) = \min\Bigg\{ & \frac{1}{9}\cdot\Big(d(v_1,g(v_1)) + d(v_2,g(v_2)) + d(v_3,g(v_3)) \\
& + d(v_4,g(v_4)) + d(v_5,g(v_5)) + d(v_6,g(v_6)) + d(v_7,g(v_7)) \\
& + d(v_8,g(v_8)) + d(v_9,g(v_9)) + d(v_1,g(v_1))\Big), \\
& \frac{1}{9}\cdot\Big(d(v_{13},g'(v_{13})) + d(v_{14},g'(v_{14})) + d(v_{15},g'(v_{15})) \\
& + d(v_{16},g'(v_{16})) + d(v_{17},g'(v_{17})) + d(v_{18},g'(v_{18})) + d(v_{19},g'(v_{19})) \\
& + d(v_{20},g'(v_{20})) + d(v_{21},g'(v_{21})) + d(v_{13},g'(v_{13}))\Big)\Bigg\} \\
= \min\Bigg\{ & \frac{1}{9}\cdot\Big(d(v_1,v_{13}) + d(v_2,v_{13}) + d(v_3,v_{13}) + d(v_4,v_{13}) + d(v_5,v_{13}) \\
& + d(v_6,v_{13}) + d(v_7,v_{13}) + d(v_8,v_{13}) + d(v_9,v_{13}) + d(v_1,v_{13})\Big), \\
& \frac{1}{9}\cdot\Big(d(v_{13},v_1) + d(v_{14},v_1) + d(v_{15},v_1) + d(v_{16},v_1) + d(v_{17},v_1) \\
& + d(v_{18},v_1) + d(v_{19},v_1) + d(v_{20},v_1) + d(v_{21},v_1) + d(v_{13},v_1)\Big)\Bigg\} \\
= & \frac{1}{9}\cdot(4+5+5+5+5+5+5+5+5+4) \\
= & \frac{48}{9} \\
\nleq & \frac{8}{9} + \frac{8}{9} = \delta_{mad}(p,q) + \delta_{mad}(q,r).
\end{aligned}$$

The same counterexample works for the normalized matched average distance. □

FURTHER PROPERTIES The basic idea of the matched average distance is the same as the idea of the simple average distance – to measure how far the nodes of the two paths are from each other on average. Though, the matched average distance includes a mapping which matches each node of the longer path to the node of the shorter path which is its closest by distance. This modification causes some advantages and some disadvantages. The costs to compute the matched average distance for two paths increases in comparison to the simple average distance, because for each node of the longer node, it needs to be computed to which other node it can be matched. Additionally, it is not obvious that the choice of the matching function g as it was made in section 3.3 is the optimal one. This mapping, for example, allows that all nodes of one path are matched onto only one node of the other path which might be an unwanted effect. Furthermore, because the nodes of the longer path are matched onto the nodes of the shorter path, the measure is not stable in the sense that small changes on the

paths might yield big changes in the measure value: if – by deletion or insertion of nodes – the formerly shorter path becomes the longer one, the mapping function might completely change which might give considerably different values for the distance measure of the paths. Additionally, integrating the matching function g into the average distance measure has the effect that two metric properties which were satisfied by the simple average distance, do not hold anymore for the matched average distance, namely coincidence and the triangle inequality which is why the matched average distance is not a metric.

3.4.3 *Node set similarity*

PREFIX AND SUFFIX CONSISTENCY σ_{nss} satisfies prefix and suffix consistency.

Proof. We only show suffix consistency, the proofs for prefix consistency are similar.

It is needed to prove that for paths $p, q \in \mathcal{P}_V$ and $r \in \mathcal{P}_V$ with $(\omega_p, \alpha_r), (\omega_q, \alpha_r) \in E$, it holds

$$\sigma_{nss}(p, q) \leq \sigma_{nss}(p \oplus r, q \oplus r).$$

In the following, let $V_p := V(p)$, $V_q := V(q)$, $V_r := V(r)$.
It holds

$$\begin{aligned}
\sigma_{nss}(p \oplus r, q \oplus r) &= |V(p \oplus r) \cap V(q \oplus r)| \\
&= |(V_p \cup V_r) \cap (V_q \cup V_r)| \\
&= |(V_p \cap V_q) \cup V_r| \\
&= |V_r| + \underbrace{|V_p \cap V_q|}_{=\sigma_{nss}(p,q)} - |V_p \cap V_q \cap V_r| \\
&= |V_r| + \sigma_{nss}(p, q) - |V_p \cap V_q \cap V_r|
\end{aligned}$$

Because $V_r \supseteq V_r \cap V_p \cap V_q$ and therefore $|V_r| \geq |V_r \cap V_p \cap V_q|$, it holds

$$\begin{aligned}
\sigma_{nss}(p \oplus r, q \oplus r) &= \sigma_{nss}(p, q) + \underbrace{|V_r| - |V_p \cap V_q \cap V_r|}_{\geq 0} \\
&\geq \sigma_{nss}(p, q)
\end{aligned}$$

which was to show. □

INSERTION CONSISTENCY σ_{nss} satisfies insertion consistency.

Proof. It needs to be shown that for any paths $p, q, p', q', r \in \mathcal{P}_V$ which can be concatenated to the paths $p \oplus r \oplus p'$ and $q \oplus r \oplus q'$ and $p \oplus p'$ and $q \oplus q'$, i.e. $\{(\omega_p, \alpha_r), (\omega_q, \alpha_r), (\omega_r, \alpha_{p'}), (\omega_r, \alpha_{q'}), (\omega_p, \alpha_{p'}), (\omega_q, \alpha_{q'})\} \subseteq E$, it holds that the paths with the common subpath are more similar to each other than the ones without the common subpath, i.e.

$$\sigma_{nss}(p \oplus p', q \oplus q') \leq \sigma_{nss}(p \oplus r \oplus p', q \oplus r \oplus q').$$

Since σ_{nss} is a set based measure, the proof from suffix consistency can be adapted: It holds

$$\begin{aligned}
& \sigma_{nss}(p \oplus r \oplus p', q \oplus r \oplus q') \\
= \quad & \left|V(p \oplus r \oplus p') \cap V(q \oplus r \oplus q')\right| \\
= \quad & \left|(V_p \cup V_r \cup V_{p'}) \cap (V_q \cup V_r \cup V_{q'})\right| \\
= \quad & \left|\left((V_p \cup V_{p'}) \cap (V_q \cup V_{q'})\right) \cup V_r\right| \\
= \quad & |V_r| + \underbrace{\left|(V_p \cup V_{p'}) \cap (V_q \cup V_{q'})\right|}_{=\sigma_{nss}(p \oplus p', q \oplus q')} \\
& - \left|(V_p \cup V_{p'}) \cap (V_q \cup V_{q'}) \cap V_r\right| \\
= \quad & |V_r| + \sigma_{nss}(p \oplus p', q \oplus q') \\
& - \left|(V_p \cup V_{p'}) \cap (V_q \cup V_{q'}) \cap V_r\right|
\end{aligned}$$

Since $(V_p \cup V_{p'}) \cap (V_q \cup V_{q'}) \cap V_r \subseteq V_r$ and therefore $|V_r| \geq \left|(V_p \cup V_{p'}) \cap (V_q \cup V_{q'}) \cap V_r\right|$, it holds

$$\begin{aligned}
& \sigma_{nss}(p \oplus r \oplus p', q \oplus r \oplus q') \\
= \quad & \sigma_{nss}(p \oplus p', q \oplus q') + \underbrace{|V_r| - \left|(V_p \cup V_{p'}) \cap (V_q \cup V_{q'}) \cap V_r\right|}_{\geq 0} \\
\geq \quad & \sigma_{nss}(p \oplus p', q \oplus q')
\end{aligned}$$

□

CONCATENATION CONSISTENCY σ_{nss} satisfies concatenation consistency.

Proof. Let

$$\begin{aligned}
& \sigma_{nss}(p \oplus p', q \oplus q') \\
= \quad & \left|V(p \oplus p') \cap V(q \oplus q')\right| \\
= \quad & \left|(V_p \cup V_{p'}) \cap (V_q \cup V_{q'})\right| \\
\geq \quad & \min\left\{\left|V_p \cap V_q\right|, \left|V_{p'} \cap V_{q'}\right|\right\},
\end{aligned}$$

since $V_p \cap V_q \subseteq (V_p \cup V_{p'}) \cap (V_q \cup V_{q'})$ and $V_{p'} \cap V_{q'} \subseteq (V_p \cup V_{p'}) \cap (V_q \cup V_{q'})$, and therefore

$$\sigma_{nss}(p \oplus p', q \oplus q') \geq \min\left\{\sigma_{nss}(p, q), \sigma_{nss}(p', q')\right\}.$$

□

EDGE IMBALANCE CONSISTENCY σ_{nss} and $\sigma_{nss,N}$ do not satisfy edge imbalance consistency.

Proof. Consider the paths p, q, r in figure 8a. It holds $|p| = |q| = |r|$ and

$$|E(p) \cup E(q)| = 1 > 0 = |E(p) \cup E(r)|,$$

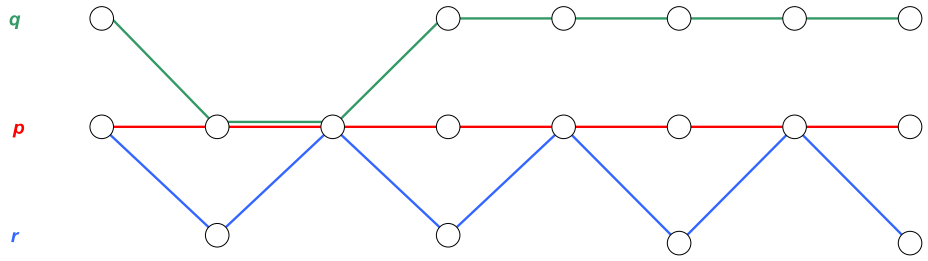
but

$$\sigma_{nss}(p, q) = 2 < 4 = \sigma_{nss}(p, r)$$

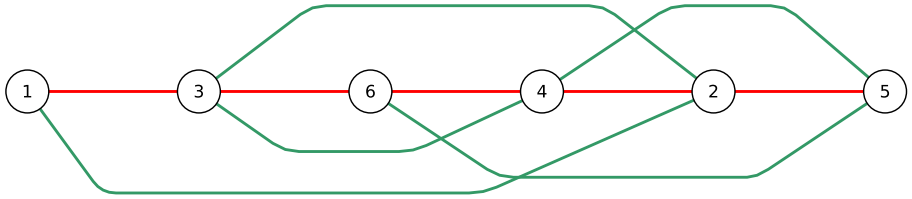
and

$$\sigma_{nss,N}(p, q) = \frac{2}{14} = \frac{1}{7} < \frac{1}{3} = \frac{4}{12} = \sigma_{nss,N}(p, r).$$

□



(a) An example in which the node set similarity does not satisfy edge imbalance consistency.



(b) An example in which the node set similarity does not satisfy coincidence.

Figure 8: Examples for properties of the node set similarity.

BOUNDEDNESS σ_{nss} and $\sigma_{nss,N}$ satisfy $|V|$-boundedness and 1-boundedness, respectively.

Proof. Since for any two paths $p, q \in \mathcal{P}_V$, $V_p \cap V_q \subseteq V$ holds, $\sigma_{nss}(p,q) \leq |V|$ follows directly. Similarly, and with $|V_p \cap V_q| \leq |V_p \cup V_q|$, $\sigma_{nss,N}(p,q) \leq 1$ follows. In any connected graphs, σ_{nss} satisfies strong $|V|$-boundedness, since there is always a path p for which $V(p) = V$ holds, then it follows

$$\sigma_{nss}(p,p) = |V(p)| = |V|.$$

In any graph, $\sigma_{nss,N}$ satisfies strong 1-boundedness, because for any path $p \in \mathcal{P}_V$, it holds

$$\sigma_{nss,N}(p,p) = \frac{|V(p)|}{|V(p)|} = 1.$$

□

NON-NEGATIVITY σ_{nss} and $\sigma_{nss,N}$ satisfy non-negativity.

Proof. Since for any set M, $|M| \geq 0$ holds, $\sigma_{nss} \geq 0$ and $\sigma_{nss,N} \geq 0$ holds. □

COINCIDENCE σ_{nss} and $\sigma_{nss,N}$ do not satisfy coincidence.

Proof. Consider the paths in figure 8b, i.e. let $p = (v_1, v_3, v_6, v_4, v_2, v_5)$ and $q = (v_1, v_2, v_3, v_4, v_5, v_6)$. Assume, that the graph does not contain any further nodes. Then

$$\sigma_{nss}(p,q) = |V_p \cap V_q| = |V|,$$

but $p \neq q$. Furthermore,

$$\sigma_{nss,N}(p,q) = \frac{|V_p \cap V_q|}{|V_p \cup V_q|} = \frac{|V|}{|V|} = 1,$$

but $p \neq q$. □

SYMMETRY σ_{nss} and $\sigma_{nss,N}$ satisfy symmetry.

Proof. For two paths $p, q \in \mathcal{P}_V$, it holds

$$\sigma_{nss}(p,q) = |V_p \cap V_q| = |V_p \cap V_q| = \sigma_{nss}(q,p)$$

and

$$\sigma_{nss,N}(p,q) = \frac{|V_p \cap V_q|}{|V_p \cup V_q|} = \frac{|V_q \cap V_p|}{|V_q \cup V_p|} = \sigma_{nss,N}(q,p)$$

□

TRIANGLE INEQUALITY σ_{nss} satisfies the triangle inequality.

Proof. We will prove the statement by contradiction. Assume that there are paths $p, q, r \in \mathcal{P}_V$ for which the node set similarity does not satisfy the triangle inequality, i.e.

$$\sigma_{nss}(p,r) < \sigma_{nss}(p,q) + \sigma_{nss}(q,r) - C$$

for the C-bounded σ_{nss} with $C = |V|$. Then

$$\begin{aligned} & |V_p \cap V_r| < |V_p \cap V_q| + |V_q \cap V_r| - |V| \\ \Leftrightarrow\ & |V| + |V_p \cap V_r| < |V_p \cap V_q| + |V_q \cap V_r| \end{aligned} \tag{1}$$

Since $0 \leq |V| + |V_p \cap V_r|$, it follows that

$$0 < |V_p \cap V_q| + |V_q \cap V_r|.$$

Therefore, there are three cases,

(i) $V_p \cap V_q \neq \emptyset$ and $V_q \cap V_r = \emptyset$,
(ii) $V_p \cap V_q = \emptyset$ and $V_q \cap V_r \neq \emptyset$,
(iii) $V_p \cap V_q \neq \emptyset$ and $V_q \cap V_r \neq \emptyset$.

Case (i) From the fact $V_p \cap V_q \neq \emptyset$, it follows that $V_p \neq \emptyset$ and $V_q \neq \emptyset$. Then equation (1) becomes to

$$\begin{aligned} & |V| + |V_p \cap V_r| < \underbrace{|V_p \cap V_q|}_{>0} + \underbrace{|V_q \cap V_r|}_{=0} \\ \Rightarrow\ & |V| + |V_p \cap V_r| < \underbrace{|V_p \cap V_q|}_{\substack{\leq |V| \\ \text{since} \\ V_p \cap V_q \subseteq V}} \\ \Rightarrow\ & |V| + \underbrace{|V_p \cap V_r|}_{\geq 0} < |V| \end{aligned}$$

which is a contradiction.

Case (ii) Similarly to case (i).

Case (iii) Similarly to the cases before, from the fact that both intersections are non-empty, it follows that the node sets of p, q and r can not be empty, i.e. $V_p \neq \emptyset$ and $V_q \neq \emptyset$ and $V_r \neq \emptyset$, hence equation (1) turns to

$$|V| + |V_p \cap V_r| < \underbrace{|V_p \cap V_q|}_{>0} + \underbrace{|V_q \cap V_r|}_{>0}.$$

In order to bring this case to a contradiction, some changes to the node sets are made which will not change the inequation, but will lead to a contradiction. It can be observed that any changes of V_q will not in- or decrease the left side of the equation. Furthermore note that adding elements to the set V_q will only increase (and not decrease) the right side of the equation which preserves the inequality. Therefore the set $V_{q'} := V_p \cup V_q \cup V_r$ is defined and get

$$\begin{aligned} |V| + |V_p \cap V_r| &< |V_p \cap V_q| + |V_q \cap V_r| \\ &\leq |V_p \cap V_{q'}| + |V_{q'} \cap V_r| \\ &= |V_p| + |V_r| \\ \Leftrightarrow |V| &< |V_p| + |V_r| - |V_p \cap V_r| \\ &= |V_p \cup V_r| \leq |V| \end{aligned}$$

which is a contradiction. Therefore, such paths p, q, r for which the triangle inequality does not hold, can not exist. Hence, the triangle inequality holds for σ_{nss}.

□

FURTHER PROPERTIES The node set similarity is a set based measure which simply counts how many nodes the two paths have in common, in the normalized case, the number of common nodes is then divided by the number of nodes the paths *could* have in common. Since it is a set based measure, the order of the nodes in which they occur in the paths is totally neglected which is also the reason why it does not satisfy coincidence: two paths are not necessarily the same if they consist of the same set of nodes, the order in which they occur and which edges the paths use, are also characteristics of paths which need to be identical in order to satisfy equality. This additionally causes that there might be cases in which neither a high value of the node set similarity nor a low value have a significant meaning: despite a high value, the paths can be different, and despite a low value, the paths might be considered as similar, for example if they "run parallel" to each other. This is not measurable with the node set similarity which is why it is important to carefully choose a similarity or distance measure for an analysis in a application domain. Furthermore, the node set similarity is not a metric because the coincidence property is not satisfied. Still, the other requirements for being a metric – non-negativity, symmetry, and the triangle inequality – are satisfied.

3.4.4 *Edge set similarity*

The formulas for the node set similarity and the edge set similarity are of the same structure, they are both a union of two sets – either the set contains nodes or edges. Also the formulas of the normalized node set similarity and the normalized edge set similarity have the same structure, each of them is the intersection of two sets divided by the union of the same two sets. Therefore, in the following, all proofs of properties which only use the properties of sets and do not use the fact that the sets contain edges, can be transferred from section 3.4.3.

PREFIX AND SUFFIX CONSISTENCY σ_{ess} and $\sigma_{ess,N}$ satisfy prefix and suffix consistency.

Proof. We will show that σ_{ess} satisfies suffix consistency, the proof for prefix consistency is similar.

It needs to be proven that for paths $p, q, r \in \mathcal{P}_V$ with $(\omega_p, \alpha_r), (\omega_q, \alpha_r) \in E$, it holds

$$\sigma_{ess}(p, q) \leq \sigma_{ess}(p \oplus r, q \oplus r).$$

The proof is quite similar to the proof of prefix consistency for the node set similarity, but it needs to be considered that concatenating two paths might add an additional edge to the edge sets.

In the following, let

$$E_p := E(p)$$

for any path $p \in \mathcal{P}_V$.

It holds

$$\begin{aligned}
& \sigma_{ess}(p \oplus r, q \oplus r) \\
= \quad & |E(p \oplus r) \cap E(q \oplus r)| \\
= \quad & |(E_p \cup E_r \cup \{(\omega_p, \alpha_r)\}) \cap (E_q \cup E_r \cup \{(\omega_q, \alpha_r)\})| \\
= \quad & |((E_p \cup \{(\omega_p, \alpha_r)\}) \cap (E_q \cup \{(\omega_q, \alpha_r)\})) \cup E_r| \\
= \quad & |E_r| + \underbrace{|E_p \cap E_q|}_{=\sigma_{ess}(p,q)} \\
& - |\underbrace{(E_p \cup \{(\omega_p, \alpha_r)\}) \cap (E_q \cup \{(\omega_q, \alpha_r)\}) \cap E_r}_{=:R}| \\
= \quad & |E_r| + \sigma_{ess}(p, q) - |R|.
\end{aligned}$$

Since $R \subseteq E_r$, $|E_r| - |R| \geq 0$ holds, and therefore,

$$\sigma_{ess}(p \oplus r, q \oplus r) = \sigma_{ess}(p, q) + \underbrace{E_r - |R|}_{\geq 0} \geq \sigma_{ess}(p, q)$$

which was to show. □

INSERTION CONSISTENCY σ_{ess} and $\sigma_{ess,N}$ do not satisfy insertion consistency.

Proof. The reason why insertion consistency does not hold for the edge set similarity, neither for the unnormalized nor for the normalized measure, is that insertion of a common subpath r adds the edges of r to the paths, and might remove one edge from both of the paths. The edge which connects the parts of the paths between which the common subpath r is inserted might be removed from the paths. If it happens that the insertion of r does not contribute any new edges in the set of common edges, the removal of this one edge is critical and decreases the edge set similarity of the two paths.

This case occurs in the example shown in figure 9a. For a better readability, the edges are drawn directed although we consider undirected graphs. Edges which occur twice in a path are also drawn twice although they occur only once in the graph, since we only consider simple graphs. Let

$$p = q = (v_1, v_2, v_3)$$

drawn in blue,

$$p' = (v_4, v_5, v_6, v_2, v_3, v_7, v_8, v_4, v_9)$$

and

$$q' = (v_4, v_5, v_6, v_2, v_3, v_7, v_8, v_4, v_{10})$$

drawn in green, and

$$r = (v_7, v_8).$$

Then,

$$p \oplus p' = (v_1, v_2, v_3, v_4, v_5, v_6, v_2, v_3, v_7, v_8, v_4, v_9)$$

and

$$q \oplus q' = (v_1, v_2, v_3, v_4, v_5, v_6, v_2, v_3, v_7, v_8, v_4, v_{10})$$

and

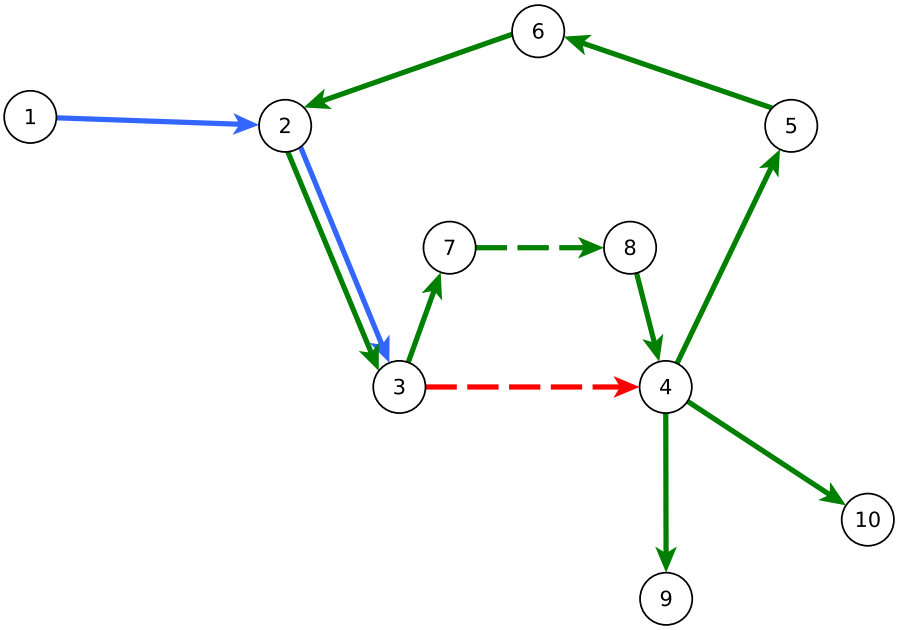
$$\begin{aligned} \sigma_{ess}(p \oplus p', q \oplus q') &= \left|E(p \oplus p') \cap E(q \oplus q')\right| = 9 \\ \sigma_{ess}(p \oplus r \oplus p', q \oplus r \oplus q') &= \left|E(p \oplus r \oplus p') \cap E(q \oplus r \oplus q')\right| \\ &= \left|\left(E(p \oplus p') \cap E(q \oplus q')\right) \setminus \{(v_3, v_4)\}\right| = 8 \end{aligned}$$

Therefore, for this example, it holds

$$\sigma_{ess}(p \oplus r \oplus p', q \oplus r \oplus q') < \sigma_{ess}(p \oplus p', q \oplus q').$$

For the normalized edge set similarity, the example in figure 9a can also serve as counterexample. There, it holds

$$\begin{aligned} \sigma_{ess,N}(p \oplus p', q \oplus q') &= \frac{|E(p \oplus p') \cap E(q \oplus q')|}{|E(p \oplus p') \cup E(q \oplus q')|} = \frac{9}{11} \\ \sigma_{ess,N}(p \oplus r \oplus p', q \oplus r \oplus q') &= \frac{|E(p \oplus r \oplus p') \cap E(q \oplus r \oplus q')|}{|E(p \oplus r \oplus p') \cup E(q \oplus r \oplus q')|} = \frac{8}{10} \end{aligned}$$



(a) An example in which the edge set similarity does not satisfy insertion consistency.

Figure 9: Examples for properties of the edge set similarity.

and

$$\sigma_{ess,N}(p \oplus p', q \oplus q') = \frac{9}{11} > \frac{8}{10} = \sigma_{ess,N}(p \oplus r \oplus p', q \oplus r \oplus q').$$

□

CONCATENATION CONSISTENCY σ_{ess} satisfies concatenation consistency.

Proof. Let

$$\begin{aligned}\sigma_{ess}(p \oplus p', q \oplus q') &= \left|E(p \oplus p') \cap E(q \oplus q')\right| \\ &= \left|\left(E_p \cup E_{p'}\right) \cap \left(E_q \cup E_{q'}\right)\right| \\ &\geq \min\left\{|E_p \cap E_q|, |E_{p'} \cap E_{q'}|\right\}\end{aligned}$$

since

$$(E_p \cap E_q) \subseteq (E_p \cup E_{p'}) \cap (E_q \cup E_{q'})$$

and

$$E_{p'} \cap E_{q'} \subseteq (E_p \cup E_{p'}) \cap (E_q \cup E_{q'}),$$

and therefore

$$\sigma_{ess}(p \oplus p', q \oplus q') \geq \min\left\{\sigma_{ess}(p, q), \sigma_{ess}(p', q')\right\}.$$

□

EDGE IMBALANCE CONSISTENCY σ_{ess} satisfies edge imbalance consistency.

Proof. Given paths $p, q, r \in \mathcal{P}_V$ with $|p| = |q| = |r|$ and

$$|E_p \cap E_q| > |E_p \cap E_r|,$$

it directly follows that

$$\sigma_{ess}(p, q) = |E_p \cap E_q| > |E_p \cap E_r| = \sigma_{ess}(p, r).$$

□

BOUNDEDNESS σ_{ess} and $\sigma_{ess,N}$ satisfy $|E|$-boundedness and 1-boundedness, respectively.

Proof. Since for any paths $p, q \in \mathcal{P}_V$, it holds $\sigma_{ess}(p, q) = |E_p \cap E_q| \leq |E|$ because $E_p \cap E_q \subseteq E$, σ_{ess} is E-bounded. There can be paths which contain all edges of a graph, therefore, σ_{ess} is also strongly $|E|$-bounded. Furthermore, $\sigma_{ess,N}$ is 1-bounded, because $\sigma_{ess,N}(p, q) = \frac{|E_p \cap E_q|}{|E_p \cup E_q|} \leq 1$, since $E_p \cap E_q \subseteq E_p \cup E_q$. In connected graphs, σ_{ess} also satisfies strong $|E|$-boundedness, since there is always a path p with $E(p) = E$, then it holds

$$\sigma_{ess}(p, p) = |E(p)| = |E|.$$

$\sigma_{ess,N}$ always satisfies strong 1-boundedness, since for any path $p \in \mathcal{P}_V$, it holds

$$\sigma_{ess,N}(p, p) = \frac{|E(p)|}{|E(p)|} = 1.$$

□

NON-NEGATIVITY σ_{ess} and $\sigma_{ess,N}$ satisfy non-negativity.

Proof. For any set M, it holds $|M| \geq 0$. Therefore, $\sigma_{ess}(p, q) \geq 0$ and $\sigma_{ess,N} \geq 0$. □

COINCIDENCE σ_{ess} and $\sigma_{ess,N}$ do not satisfy coincidence.

Proof. Consider any path p which does not contain all edges of the graph, i.e. $E_p \subsetneq E$. Then $\sigma_{ess}(p, p) = |E_p| < |E|$ which contradicts the requirement for coincidence. The normalized edge set similarity does not satisfy coincidence, either, which can be seen by considering any path p and its inverse $inv(p)$. Because only paths in undirected graphs are analyzed, the edge sets of p and $inv(p)$ are the same, therefore p and $inv(p)$ have a normalized edge set similarity of 1, i.e. $\sigma_{ess,N}(p, inv(p)) = \frac{|E_p|}{|E_p|} = 1$ although they are not equal. □

SYMMETRY σ_{ess} and $\sigma_{ess,N}$ satisfy symmetry.

Proof. For any paths $p, q \in \mathcal{P}_V$, it holds

$$\sigma_{ess}(p, q) = |E_p \cap E_q| = |E_q \cap E_p| = \sigma_{ess}(q, p)$$

and

$$\sigma_{ess,N}(p, q) = \frac{|E_p \cap E_q|}{|E_p \cup E_q|} = \frac{|E_q \cap E_p|}{|E_q \cup E_p|} = \sigma_{ess,N}(q, p).$$

□

TRIANGLE INEQUALITY σ_{ess} satisfies the triangle inequality.

Proof. The proof for the triangle inequality for edge set similarity is completely the same as the proof that the node set similarity satisfies the triangle inequality except that sets of edges instead of sets of nodes are considered. Though, the fact that the sets contain nodes is not used in the proof, but properties of general sets. Therefore, proving that the edge set similarity satisfies the triangle inequality can be done in the same way than for the node set similarity. □

FURTHER PROPERTIES Although the formulas for the node set similarity and the edge set similarity look alike, it makes a difference whether the sets of edges or the the sets of nodes of the paths are taken to compute a similarity. While the node set similarity satisfies insertion consistency and do not satisfy edge imbalance consistency, the edge set similarity behaves exactly contrarily. The problem why the edge set similarity does not satisfy the property of insertion consistency, but prefix and suffix consistency, is that the definition of insertion consistency allows the removal of one edge from the two paths. Although the insertion of a subpath comes with the addition of the edges of the subpath and two edges for concatenation, the edge set similarity might decrease after insertion. This case happens when the inserted subpath and and the two additional edges are already elements of the edge sets of the two paths, i.e. the insertion does not contribute any new edges to the edge sets. Then the removal of the one edge is a critical operation which decreases the edge set similarity. This case does not apply for the node set similarity because the node set similarity does not consider edges, only nodes, and there is no node removed from the node sets by inserting a subpath.

3.4.5 *LCSS similarity*

PREFIX AND SUFFIX CONSISTENCY σ_{lcss} and $\sigma_{lcss,N}$ satisfy prefix and suffix consistency.

Proof. As for the previous measures, we only show suffix consistency because the proofs for prefix consistency are very similar. For the proof that σ_{lcss} satisfies prefix consistency, we can make use of the properties of the function $P(p,q)$ which is the length of the longest common subsequence in p and q. It can be observed that a common subsequence of two paths is preserved if nodes are added to the paths. By adding nodes to a path (without removing any), the length of the longest common subsequence can become larger, since the new nodes can contribute to a common subsequence. This is also true if the added nodes are already contained at any other position in the path. Hence, the length of a common subsequence can not decrease by adding nodes to the paths, because they can be but do not need to be included into a common subsequence. Then,

for the unnormalized LCSS similarity, it holds for any paths $p, q, r \in \mathcal{P}_V$ with $(\omega_p, \alpha_r), (\omega_q, \alpha_r) \in E$,

$$\begin{aligned}\sigma_{lcss}(p \oplus r, q \oplus r) &= P(p \oplus r, q \oplus r) \\ &= P(p, q) + |r| + 1 \\ &> P(p, q) = \sigma_{lcss}(p, q).\end{aligned}$$

For the normalized measure, it holds

$$\begin{aligned}\sigma_{lcss,N}(p \oplus r, q \oplus r) &= \frac{P(p \oplus r, q \oplus r)}{\max\{|p \oplus r| + 1, |q \oplus r| + 1\}} \\ &= \frac{P(p, q) + |r| + 1}{\max\{|p| + 1, |q| + 1\} + |r| + 1}.\end{aligned}$$

Therefore, since

$$\begin{aligned}& \max\{|p| + 1, |q| + 1\} \geq P(p, q) \text{ and } |r| + 1 > 0 \\ \Leftrightarrow \quad & \frac{P(p, q) + |r| + 1}{\max\{|p| + 1, |q| + 1\} + |r| + 1} \geq \frac{P(p, q)}{\max\{|p| + 1, |q| + 1\}} \\ \Leftrightarrow \quad & \sigma_{lcss,N}(p \oplus r, q \oplus r) \geq \sigma_{lcss,N}(p, q)\end{aligned}$$

which was to show. □

INSERTION CONSISTENCY σ_{lcss} and $\sigma_{lcss,N}$ satisfy insertion consistency.

Proof. For any paths $p, p', q, q', r \in \mathcal{P}_V$ which can be concatenated to the paths $p \oplus p'$, $q \oplus q'$, $p \oplus r \oplus p'$ and $q \oplus r \oplus q'$, the LCSS similarity measure shows the following characteristic:

$$\begin{aligned}\sigma_{lcss}(p \oplus r \oplus p', q \oplus r \oplus q') &= P(p \oplus r \oplus p', q \oplus r \oplus q') \\ &= P(p \oplus p', q \oplus q') + |r| + 1 \\ &> P(p \oplus p', q \oplus q') = \sigma_{lcss}(p \oplus p', q \oplus q').\end{aligned}$$

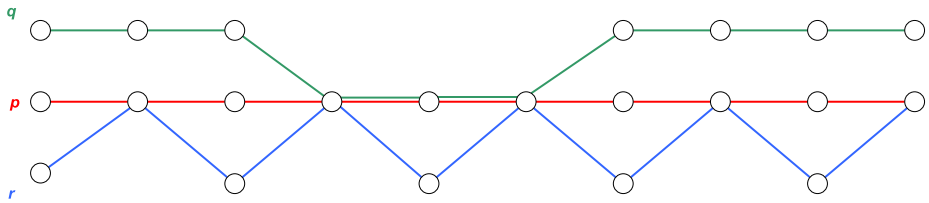
The normalized measure also satisfies insertion consistency, since

$$\begin{aligned}\sigma_{lcss,N}(p \oplus r \oplus p', q \oplus r \oplus q') &= \frac{P(p \oplus r \oplus p', q \oplus r \oplus q')}{\max\{|p \oplus r \oplus p'| + 1, |q \oplus r \oplus q'| + 1\}} \\ &= \frac{P(p \oplus p', q \oplus q') + |r| + 1}{\max\{|p \oplus p'| + 1, |q \oplus q'| + 1\} + |r| + 1}.\end{aligned}$$

Therefore, with the same principle as in the proof for suffix consistency,

$$\begin{aligned}& \max\{|p \oplus p'| + 1, |q \oplus q'| + 1\} \geq P(p \oplus p', q \oplus q') \text{ and } |r| + 1 > 0 \\ \Leftrightarrow \quad & \frac{P(p \oplus p', q \oplus q') + |r| + 1}{\max\{|p \oplus p'| + 1, |q \oplus q'| + 1\} + |r| + 1} \geq \frac{P(p \oplus p', q \oplus q')}{\max\{|p \oplus p'| + 1, |q \oplus q'| + 1\}} \\ \Leftrightarrow \quad & \sigma_{lcss,N}(p \oplus r \oplus p', q \oplus r \oplus q') \geq \sigma_{lcss,N}(p \oplus p', q \oplus q')\end{aligned}$$

which was to show. □



(a) An example in which the LCSS similarity does not satisfy edge imbalance consistency.

Figure 10: Examples for properties of the LCSS similarity

CONCATENATION CONSISTENCY σ_{lcss} satisfies concatenation consistency.

Proof. The observation that $P(p,q)$ for two paths $p,q \in \mathcal{P}_V$ cannot decrease by adding nodes to any of the two paths, is also needed in this proof. Then, for σ_{lss} and paths $p,p',q,q' \in \mathcal{P}_V$ with $(\omega_p,\alpha_{p'}),(\omega_q,\alpha_{q'}) \in E$, it directly follows

$$\begin{aligned}\sigma_{lcss}(p \oplus p', q \oplus q') &= P(p \oplus p', q \oplus q') \\ &\geq \max\{P(p,q), P(p',q')\} \\ &\geq \min\{P(p,q), P(p',q')\} \\ &= \min\{\sigma_{lcss}(p,q), \sigma_{lcss}(p',q')\}\end{aligned}$$

which was to show.

□

EDGE IMBALANCE CONSISTENCY σ_{lcss} and $\sigma_{lcss,N}$ do not satisfy edge imbalance consistency.

Proof. Consider the paths p,q,r in figure 10a. It holds $|p| = |q| = |r| = 9$ and

$$|E(p) \cup E(q)| = 2 > 0 = |E(p) \cup E(r)|,$$

but

$$\sigma_{lcss}(p,q) = 3 < 5 = \sigma_{lcss}(p,r)$$

and

$$\sigma_{lcss,N}(p,q) = \frac{3}{10} < \frac{1}{2} = \frac{5}{10} = \sigma_{lcss,N}(p,r).$$

□

BOUNDEDNESS σ_{lcss} does not satisfy C-boundedness for any constant C, $\sigma_{lcss,N}$ satisfies 1-boundedness, though.

Proof. Consider the paths $p = (v_1 \dots v_k) = q$ with $(v_k, v_1) \in E$. Let

$$p^l := \underbrace{p \oplus \dots \oplus p}_{l \text{ times}}$$

It holds

$$\begin{aligned} \sigma_{lcss}(p,q) &= k, \\ \sigma_{lcss}(p \oplus p, q \oplus q) &= 2k, \\ &\dots \\ \sigma_{lcss}(p^n, q^n) &= nk. \end{aligned}$$

Because n is not bounded, σ_{lcss} can not be bounded by any constant C.

The normalized $\sigma_{lcss,N}$ is 1-bounded, though. For any paths p, q $\sigma_{lcss}(p,q) \leq \min\{|p|+1, |q|+1\}$ and therefore,

$$\sigma_{lcss,N}(p,q) \leq 1.$$

The normalized $\sigma_{lcss,N}$ is even strongly 1-bounded, because for any path $p \in \mathcal{P}_V$, it holds

$$\sigma_{lcss,N}(p,p) = \frac{P(p,p)}{|p|+1} = 1.$$

□

NON-NEGATIVITY σ_{lcss} and $\sigma_{lcss,N}$ satisfy non-negativity.

Proof. Non-negativity directly follows from the fact that $P(p,q) \geq 0$ and $\max\{|p|+1, |q|+1\} \geq 0$. □

COINCIDENCE σ_{lcss} does not satisfy coincidence, $\sigma_{lcss,N}$ satisfies coincidence, though.

Proof. σ_{lcss} can not satisfy coincidence because σ_{lcss} is not C-bounded for any constant C and coincidence is only defined for strongly bounded similarity measures. For the normalized measure, for any paths $p, q \in \mathcal{P}_V$, it holds

$$\begin{aligned} & \sigma_{lcss,N}(p,q) = 1 \\ \Leftrightarrow \quad & \frac{P(p,q)}{\max\{|p|+1, |q|+1\}} = 1 \\ \Leftrightarrow \quad & P(p,q) = \max\{|p|+1, |q|+1\} \\ \Leftrightarrow \quad & v_{p_1} = v_{q_1}, \dots, v_{p_k} = v_{q_k} \\ \Leftrightarrow \quad & p = q, \text{ because the graph is simple.} \end{aligned}$$

□

SYMMETRY σ_{lcss} and $\sigma_{lcss,N}$ satisfy symmetry.

Proof. Let

$$\sigma_{lcss}(p,q) = P(p,q) = P(q,p) = \sigma_{lcss}(q,p)$$

and similarly

$$\sigma_{lcss,N}(p,q) = \frac{\sigma_{lcss}(p,q)}{\max\{|p|+1, |q|+1\}} = \frac{\sigma_{lcss}(q,p)}{\max\{|q|+1, |p|+1\}} = \sigma_{lcss,N}(q,p).$$

P is symmetric because, otherwise there would be a common subsequence in p and q which is not found in q and p which is not possible. □

TRIANGLE INEQUALITY σ_{lcss} does not satisfy the triangle inequality.

Proof. Since σ_{lcss} is not C-bounded, there is no appropriate formulation of the triangle inequality for the LCSS similarity. □

FURTHER PROPERTIES The unnormalized LCSS similarity is the only measure among the proposed measures which does not satisfy boundedness. This is due to the fact that in the unnormalized LCSS similarity, one node can contribute several times to the value of the LCSS similarity. While a single node appearing several times in two paths contribute only once to the set based similarities, the LCSS is computed differently: here, a single node can can be contribute arbitrarily often to the value of the longest common subsequence of two paths. Therefore, the property of boundedness can not be satisfied by the unnormalized LCSS similarity. From the fact that the unnormalized LCSS similarity does not satisfy boundedness, it follows directly that it also does not satisfy coincidence and the triangle inequality because these properties are only appropriate for C-bounded similarity measures. The normalized LCSS similarity on the other hand does satisfy boundedness and also coincidence.

4

CLUSTERING PATHS BY THEIR SIMILARITY

The previous chapter introduced several distance and similarity measures for paths and checked them for their properties. The goal is to group paths according to their similarity. Therefore, the following section gives an overview of existing approaches to cluster data, before a description of the available path data is given.

4.1 CLUSTERING APPROACHES

The main commonality of all clustering algorithms is the task of, given a set of n objects, finding groups of objects such that the objects within a group are similar to each other while the different groups are as dissimilar as possible. Although the general idea of clustering is very intuitive, the methods which are used are manifold: there is not *the one* correct clustering algorithm, each one is applicable in certain scenarios and not applicable in others. It is also important to note that there is not even *the one* best grouping of a given data set. For a fixed set of objects, there might be several meaningful groupings of which each one might reflect a different aspect of the objects' properties. Therefore, there are many approaches, methods and algorithms available, for example depending on (i) which requirements are put on the structure of the group (for example, if the groups are allowed to be overlapping or need to be a partition of the set of objects), (ii) which kind of data are available, or (iii) which further knowledge of the data is available. They all have in common that they are given a set of n objects and return groups of the objects. The objects might be represented by m attributes such that the algorithm is given an $n \times m$ matrix of object attribute values. It is also possible that the algorithm is only given the distance or similarity of each pair of objects, i.e. a $n \times n$ matrix of similarity or distance values. The latter is the case in the clustering of paths in this thesis.

In section 2.1, we presented the work of Kleinberg [23] who developed an axiomatic framework for clustering and was able to show that there does not exist a clustering function which satisfies scale-invariance, richness, and consistency. This is one of the very few works about clustering which approaches the topic from an abstract view instead of a technology-, data- or algorithm-driven way.

The following section, however, is meant to give a short overview of the available methods of finding a clustering of objects in order to justify our choice of methods for the results presented in section 4.4. More detailed explanations are, for example, given in various reviews [2, 21, 14, 16].

4.1.1 *Methods of clustering*

There are several criteria by which the available methods of clustering can be distinguished, the most common one is the underlying model on which the clustering algorithm is based on. There are

- hierarchical approaches
- centroid-based approaches
- distribution-based approaches
- density-based approaches

Hierarchical clustering approaches operate on a matrix of distance or similarity values for each pair of objects. There are agglomerative and divisive hierarchical clustering methods. In both cases, a binary tree is built whose leaves are the n objects and the inner leaves represent merging or dividing of existing clusters. The tree represents the procedure of hierarchical clustering: in agglomerative approaches, the tree is built from the bottom to the top, i.e. existing clusters are successively merged, in divisive methods, the tree is constructed from the top to the bottom, i.e. existing clusters are successively divided. Hence, each level of the tree represents one stage of the clustering procedure, all objects which are in one subtree at this stage, are in one cluster in that clustering stage. To be precise, an agglomerative hierarchical clustering methods starts with n clusters, each containing exactly one object. In each step, the clusters with the largest similarity or smallest distance are merged to one cluster. This procedure is continued until all clusters are merged to one which contains all n elements. From this conceptual description, there are already some observations about this approach possible: a big advantage and a big disadvantage of hierarchical clustering methods is the property that objects do not change their cluster identity during the (agglomerative) clustering process – objects which are put into the same cluster in the beginning of the process, will stay in the same cluster. This means early mistakes can not be corrected later in the process and might decrease the clustering quality. On the other hand, the decisions made during the process are permanent which enormously decreases the number of possibilities which need to be considered in later stages of the algorithm. A further advantage and disadvantage is the fact that the algorithm does not determine an appropriate number of clusters. This means that the number of clusters does not need to be known in advance, but also that the number of clusters, i.e. the height where the tree needs to be cut in order to get the respective clusters, needs to be determined afterwards.

In the description of the agglomerative hierarchical clustering method above, it is left open how to determine *the most similar pair of clusters* for merging. There are three common approaches to determine which two clusters are the most similar:

- *Single-linkage methods* The distance or similarity of two clusters is defined as the distance of their closest members or the similarity of their most similar members, i.e. for the two clusters C_1 and C_2 and a distance function δ on the objects, $\delta(C_1, C_2) := \min_{i \in C_1, j \in C_2} \delta(i, j)$, and for a similarity function σ on the objects, $\sigma(C_1, C_2) := \max_{i \in C_1, j \in C_2} \sigma(i, j)$.
- *Complete-linkage methods* The distance or similarity of two clusters is defined as the distance of their members which are furthest

apart or the least similar, i.e. $\delta(C_1, C_2) := \max_{i \in C_1, j \in C_2} \delta(i, j)$ and $\sigma(C_1, C_2) := \min_{i \in C_1, j \in C_2} \sigma(i, j)$.

- *Average-linkage methods* Instead of considering the members of the clusters which are most extreme, the average-linkage method considers how far the members of C_1 are in average to the members of C_2, i.e. $\delta(C_1, C_2) = \frac{1}{|C_1| \cdot |C_2|} \sum_{i \in C_1} \sum_{j \in C_2} \delta(i, j)$.

Non-hierarchical clustering methods have in common that they start with an (often randomly selected) initial clustering of the objects and the algorithm then alters the objects' cluster membership in order to find a better partition. By which criterion the partition is optimized then depends on the method.

The idea of *centroid-based clustering methods* is to represent each cluster by its "center" and in each step, the objects are assigned to the cluster whose center is closest to them. The general procedure for iterative centroid-based clustering methods is the following as described by Jain and Dubes [16]:

(i) select an initial partition with k clusters.
(ii) generate a new partition by assigning each object to the cluster which center is closest.
(iii) compute new cluster centers as centroids of the clusters.
(iv) repeat steps (ii) and (iii) until an optimum value of the criterion function is found.
(v) adjust number of clusters by merging and splitting existing clusters or by removing small or outlier clusters.

The commonly used *k-means* clustering algorithm (which goes back to Steinhaus [42] and Lloyd [33]) for example, assigns the objects to the clusters such that the sum of squared errors within the cluster is minimized, i.e. the sum of the squared distances of each object to its cluster center. Formally, for any set of objects S, any $k \geq 2$ and any distance function $d : S \times S \to \mathbb{R}$, initially k centroid points $T \subseteq S$ are chosen for which the function $\Lambda_d(T) = \sum_{i \in S} d^2(i, T)$ with $d(i, T) = \min_{j \in T} d(i, j)$ is minimized. Then, each object is assigned to that element of T which is closest to it [23]. This procedure is repeated until the partition does not change anymore. It can be shown that the algorithm always finds an at least local optimum.

Further clustering approaches include distribution-based and density-based approaches which make different assumptions of the data. In distribution-based approaches, it is assumed that objects in the same cluster come – most likely – from the same distribution, and clusters are determined based on this assumption, for example by the expectation-maximization-algorithm [8]. Density-based approaches [24] consider clusters as areas with a higher density of objects, which is for example implemented in the *DBSCAN* algorithm by Ester et al. [9].

4.2 AVAILABLE DATA

This section contains a description and a summary of the data set that is used to cluster paths. The following subsections contain information about the data's source, the preprocessing steps and a summary of its structure.

4.2.1 *Source and summary of the data*

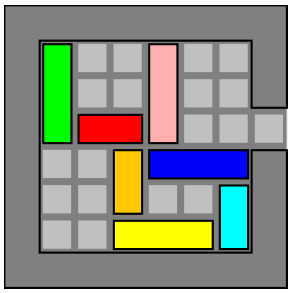


Figure 11: An example for a *Rush Hour* board. The yellow car needs to be removed from the board.

We are testing the proposed similarity and distance measures and the clustering approach on paths in a network which represents the problem space of a game. We consider the board game *Rush Hour*[1] which is a one-player block sliding puzzle. It takes place on a board of 6×6 cells with one designated exit on which blocks are placed which can have a size of 1×2, 1×3, 2×1 or 3×1 cells. The board and the blocks represent a parking lot with parking cars. The goal of the game is to find a sequence of moves which allows a particular car (which will be called *target car* in the following) to exit the board through the designated exit. An allowed move is to move a car at a time an arbitrary number of cells forwards or backwards, but not sideways. An example for a possible *Rush Hour* game is shown in figure 11.

We generate a graph from a *Rush Hour* start configuration by the following: let $\mathcal{C}$ be the set of all possible *Rush Hour* configurations for a board of the given size. We define $move : \mathcal{C} \to \mathcal{P}(\mathcal{C})$, with $\mathcal{P}(\mathcal{C})$ the power set of $\mathcal{C}$, as the function which returns all configurations which are reachable from a given configuration by an allowed move. Furthermore, for a configuration set $C \subseteq \mathcal{C}$, we can define inductively

$$\begin{aligned} move^0(C) &:= C \\ move^n(C) &:= move^{n-1}(C) \cup \bigcup_{v \in move^{n-1}(C)} move(v) \end{aligned}$$

the configurations which can be reached in at most n moves.

Then, for a board configuration $c \in \mathcal{C}$, we define the associated problem space as $G^c := (V^c, E^c)$ with

$$V^c = move^m(\{c\})$$

with $m \in \mathbb{N}$ such that

$$move^m(\{c\}) = move^{m+1}(\{c\})$$

(such a m always exists) and with

$$E^c = \{(c_1, c_2) \in V^c \times V^c | c_2 \in move(c_1)\}.$$

Hence, the problem space for a given board configuration consists of all board configurations which can be reached from the given configuration by allowed moves. We consider a *Rush Hour* game instance as solved when the cars on the board are in such positions that the target car can be removed from the board with one additional move. We call configurations in which this is the case, *solution states* or *final states* $V_f^c \in V^c$. With the concept of the problem space, solving

1 The game was invented by Nob Yoshigahara and is distributed by ThinkFun Inc. and HCM Kinzel (Germany).

a *Rush Hour* game instance can then be understood as finding a path from c to a solution state $v_f \in V_f^c$, i.e. a path $p \in \bigcup_{v_f \in V_f^c} \mathcal{P}_{c \to v_f}$. In the optimal case, the found path is as short as possible.

The data set we used for analysis was collected by Pelánek and Jarušek from the Masaryk university in Czech republic [18]. They developed a *problem solving tutor*[2] which is a web-based tool for learning by problem solving and is used in educational contexts. Students who use the system can solve problems in a game-based manner while the system is adapting to the student's capabilities and recommending game instances of an appropriate difficulty. A detailed description can be found in Jarušek [17].

Among others, the system contains *Rush Hour* game instances of different degrees of difficulty. We got access to the log data of all users of the system how they solved (or attempted to solve) the *Rush Hour* instances. The log data were accessed on January 09, 2015 and contain for each user, each instance and each trial, all moves done by the user with a time stamp for each move. The next subsection describes all necessary preprocessing steps of the raw log data.

2 It can be found under tutor.fi.muni.cz.

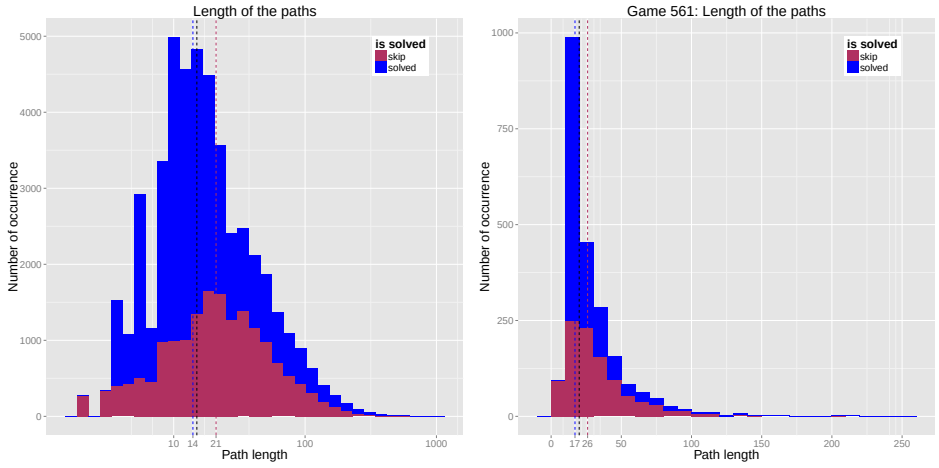
GAME	NODES	EDGES	GAME	NODES	EDGES
Game 121	4405	33 302	Game 561	4374	37 078
Game 152	3008	25 288	Game 578	2853	24 732
Game 197	2962	26 496	Game 579	4573	35 232
Game 19	1169	8620	Game 5	5872	51 603
Game 202	4635	38 176	Game 64	2952	21 017
Game 246	3003	22 418	Game 655	1075	11 599
Game 254	8648	61 530	Game 674	6090	53 537
Game 260	3095	24 919	Game 692	887	5226
Game 266	451	3983	Game 722	2241	14 517
Game 27	10 653	101 441	Game 723	830	7978
Game 326	3493	27 529	Game 727	2784	29 481
Game 32	2954	27 900	Game 765	1327	10 143
Game 342	364	2996	Game 779	2376	18 892
Game 355	9121	85 951	Game 820	7235	63 551
Game 357	4426	37 649	Game 841	1050	5957
Game 37	950	6461	Game 878	4259	35 858
Game 393	4533	30 587	Game 906	864	6934
Game 441	4533	30 587	Game 939	3268	22 956
Game 551	8542	88 706	Game 96	496	2849
Game 559	5322	51 977	Game 981	4925	35 126

Table 3: The sizes of the problem spaces of the games.

4.2.2 *Preprocessing the data*

We performed the following data processing steps.

- *Extract the level information* For each of the 61 game instances, we extracted the information of how the start configuration looks like, i.e. how many cars of which size are placed on which position on the board. This gives a set of configurations $\mathcal{C}$ of size 61.
- *Compute the problem spaces* For each configuration $c \in \mathcal{C}$, we computed the associated state space if it was feasible. 21 configurations were rejected because their problem spaces are too large for a further feasible analysis. Table 3 shows the size of the problem spaces of the remaining configurations, i.e. the number of contained nodes and edges.
- *Extract and filter paths* Any user who attempts or achieves to solve a game instance creates a path in the problem space of the configuration. For each user, each configuration and each attempt, we extract the generated path from the log data. Any move which is done after a final state was reached is not considered anymore, but the path is considered as solved path. Probably due to temporary network connections problems, some paths are incomplete or inconsistent. Therefore, all extracted paths are checked for completeness and consistency (i.e. , that the timestamps are consistent, there are no moves missing, all done moves are allowed moves), and all inconsistent or incomplete paths are excluded from the further analysis. Table 4 shows for each configuration how many paths were extracted and how many paths remain after filtering the inconsistent and incomplete ones out.
- *Cut paths* To be able to compute the simple average distance as it is defined in section 3.3.1, the paths need to have equal length. However, even for one configurations, the extracted paths have a broad range of length as it can be seen in figure 12. We therefore cut all extracted paths of each configuration $c \in \mathcal{C}$ to a particular length k_c and consider only the first k_c nodes of the paths. For determining the cutting point k_c for each configuration, the following idea is used: if a cutting point k_c is chosen, all paths which are shorter than k_c are completely rejected because they are too short; all paths which are longer than k_c are cut to this length and all path nodes by which the paths are longer than k_c are rejected. For this reason, k_c should not be chosen too large because then many short paths are rejected, but also not too small because then the major part of the paths is cut off and the paths become meaningless. In order to find a good choice for this trade-off, we chose the cutting point k_c for each configuration which minimizes the number of rejected path nodes. It should be kept in mind that this method especially rejects all optimal solutions. But since the optimal solutions are only a small part of the paths, this rejection is acceptable.An example for this method is shown in figure 13. With increasing cutting point, the number of rejected paths from too short paths increases, while the number of rejected nodes from too long paths decreases. The sum of both numbers has a minimum, the corresponding cutting point for the configuration is chosen, and all available paths are cut to this length. Table 4 shows the



(a) For all configurations. (b) For a specific configuration.

Figure 12: The distribution of the path lengths for all extracted paths, after filtering. The colors encode whether the paths ends in a final state or not. The dashed lines show the mean path length for all (black), solving or non-solving (skipped) paths.

chosen cutting point for each configuration and how many paths remain for further analysis.

The following subsection will give an exploratory overview of the data which were extracted and preprocessed in the way described above.

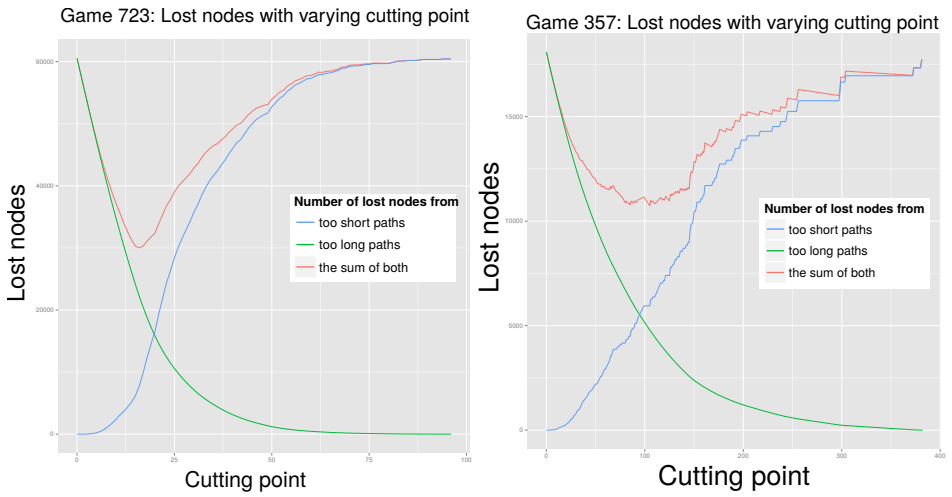
4.2.3 *Summary of the extracted data*

This subsection is meant to give an impression of the available extracted path data and their underlying networks. A purely number-based overview of the extracted paths and networks can be found in the tables 3 and 4. A more visual summary is given in the following.

THE UNDERLYING NETWORKS As already mentioned, the problem spaces of the available configurations contain several thousands nodes which makes it difficult to visualize them properly. For a configuration with a relatively small problem space, a visualization of the problem space is shown in figure 14. The nodes which occur in at least one of the extracted paths are coloured in green, yellow, red, or blue, respectively, depending whether they are the start configuration (green), they are nodes in which at least one user canceled the game (red), or whether they are final states (blue). All nodes which do not occur in any of the extracted paths are shown in grey. The edge colour also depends on whether this edge is part of any of the extracted paths or not. The edges are drawn as often as they occur in the extracted paths. It is obvious that this visualization is only useful to get a first impression of the state space and the contained paths and only applicable to small state spaces. Nonetheless, figure 14 clearly shows that the user do not explore the whole state space: even for more than 2500 paths (of which over 2000 are solved) in a state space of over 450 nodes, about 190 nodes do not occur in any of the paths.

	NUMBER OF AVAILABLE PATHS							OPTIMAL PATH LENGTH	CUT THRESHOLD
	total	after filtering			after cutting				
		total	*solved*	*not solved*	*total*	*solved*	*not solved*		
Game 121	331	270	39	231	70	33	37	47	112
Game 152	241	187	59	128	102	52	50	36	62
Game 19	762	662	213	449	308	185	123	31	48
Game 197	218	167	46	121	81	42	39	44	66
Game 202	417	359	89	270	153	76	77	41	61
Game 246	637	552	158	394	315	153	162	33	45
Game 254	463	394	40	354	102	35	67	43	82
Game 260	306	247	54	193	103	49	54	48	66
Game 266	2643	2523	2098	425	1887	1763	124	8	11
Game 27	262	209	68	141	125	67	58	44	61
Game 32	2383	2284	1917	367	2050	1873	177	8	10
Game 326	358	290	48	242	92	40	52	50	84
Game 342	3271	3171	3044	127	3132	3044	88	3	4
Game 355	218	154	53	101	84	47	37	26	64
Game 357	261	205	58	147	70	39	31	42	105
Game 37	2633	2458	1697	761	1979	1485	494	11	14
Game 393	233	175	53	122	104	53	51	49	65
Game 441	238	178	59	119	94	54	40	49	76
Game 5	1760	1614	428	1186	757	408	349	25	32
Game 551	2706	2589	2345	244	2517	2345	172	5	6
Game 559	2495	2369	2024	345	1963	1754	209	7	11
Game 561	2432	2259	1258	1001	1177	518	659	9	20
Game 578	1005	904	230	674	497	228	269	31	36
Game 579	603	511	150	361	298	141	157	30	38
Game 64	3064	2934	2592	342	2849	2592	257	5	6
Game 655	2908	2769	2556	213	2402	2302	100	7	10
Game 674	370	306	90	216	161	82	79	44	55
Game 692	475	404	89	315	161	86	75	46	53
Game 722	202	156	47	109	72	43	29	48	76
Game 723	2876	2704	1472	1232	1909	1388	521	13	16
Game 727	2554	2397	1980	417	1886	1641	245	8	11
Game 765	534	462	109	353	195	91	104	30	48
Game 779	1585	1429	399	1030	635	387	248	29	36
Game 820	259	212	44	168	67	32	35	41	92
Game 841	258	203	65	138	115	62	53	45	59
Game 878	1064	926	306	620	573	247	326	20	36
Game 906	2214	2013	520	1493	1060	519	541	24	29
Game 939	1393	1198	301	897	677	211	466	21	32
Game 896	2254	2164	1746	418	1980	1746	234	8	9
Game 981	2156	1993	1178	815	1306	907	399	9	20
Total	51 042	47 001	29 722	17 279	34 108	26 820	7288		

Table 4: Summary of the available paths: it is shown how many paths for each game are available by the used data set. The first column shows the total number, the second column contains the number of paths after the inconsistent and incomplete paths were filtered out, the third column contains the number of remaining paths after the filtered paths were cut to equal length according to the described method. The determined cut threshold to which length the paths are cut can be seen in the last column of the table. Furthermore, the length of the optimal solution path can be found in the second but last column.



(a) For game 723.

(b) For game 357.

Figure 13: The number of rejected nodes with varying cutting point: when the cutting threshold is large, all nodes of paths which are shorter than the cutting point are rejected; when the cutting point is small, all nodes by which the longer paths are longer than the cutting point are rejected. The absolute number of rejected nodes for the respective configuration is plotted against the cutting point, either for the too short paths (in blue), or for the too long paths (in green), or the sum of both (in red).

PROPERTIES OF THE EXTRACTED PATHS In section 1.3, we defined several path properties, particularly that paths can be simple and elementary, i.e. each node or edge occurs at most once in a path. For the present data set where a path represents a solution path for a game, a path which is not simple or not elementary is clearly not an optimal path, since such a path contains at least one cycle, and this can not be the case in an optimal solution path.

Figure 15 shows the composition of the available paths with respect to the properties of being simple and/or elementary, once for all complete paths (figure 15a) and once for the paths which were cut according to the described method (figure 15b). For the latter, only the part of the cut paths which is not rejected is checked for being simple and/or elementary, any edge or node which is occurring more than once in the removed part of the paths, is not considered. There are several insights that can be gained from these two visualizations: at first, it gives an impression of the dimension of the available path data, particularly the large differences in available paths for the different configurations, i.e. there are either around 2000 paths available or around 200. This is due to the way the data was collected: the problem solving tutor where the path data is extracted from, suggests games to the user according to the user's skill, starting from easier games continuing with more difficult games. Hence, the easier games in the beginning are played by far more users than the other ones. There is even a subset of configurations which is not accessible for the user before certain other games are solved. Knowing this, the large differences between the number of available paths is not surprising. Additionally, the figure visualizes in which proportions the extracted paths are *(i)* simple and elementary, or *(ii)* simple, but not elementary, or *(iii)* neither simple nor elementary. It is astonishing how many of

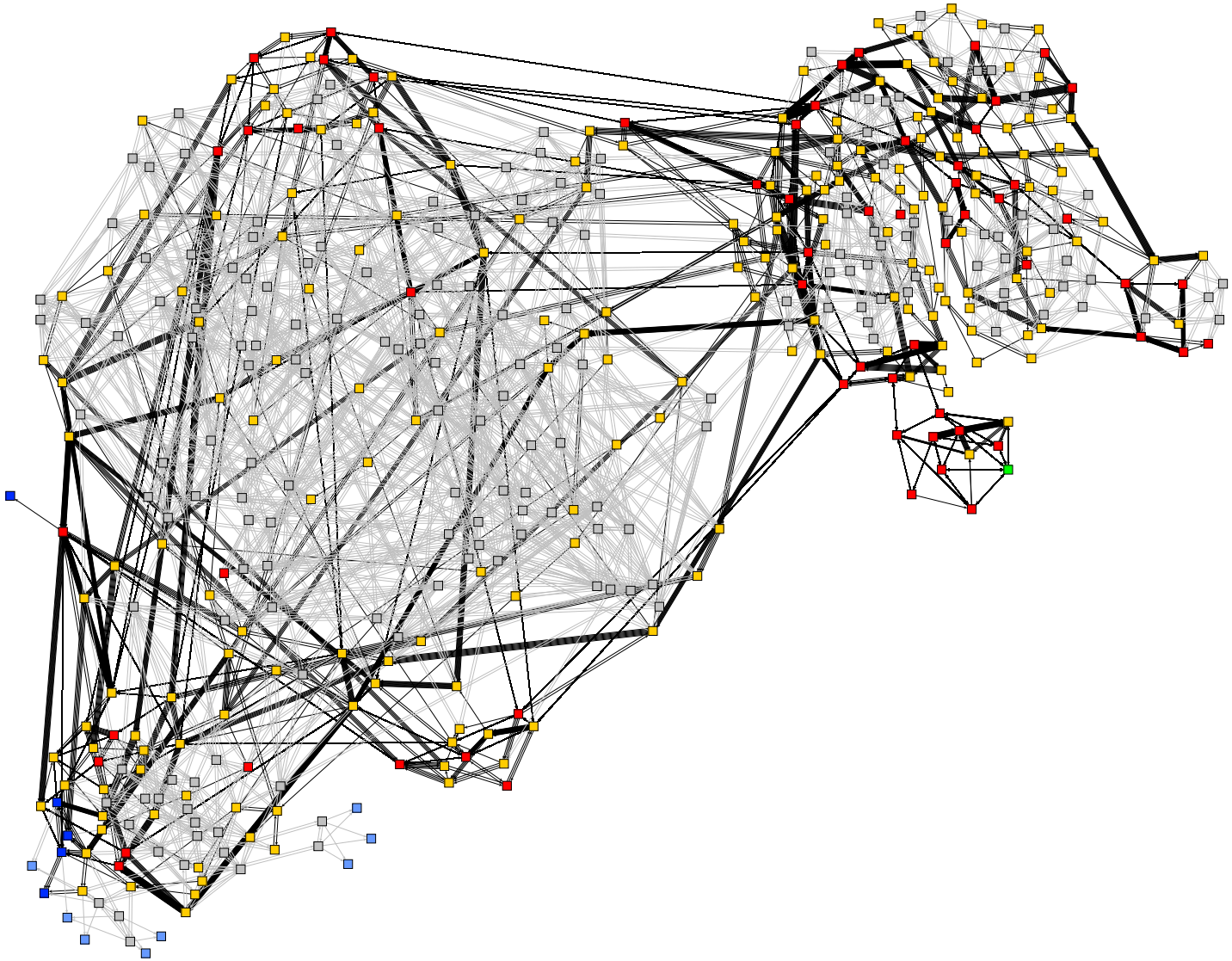
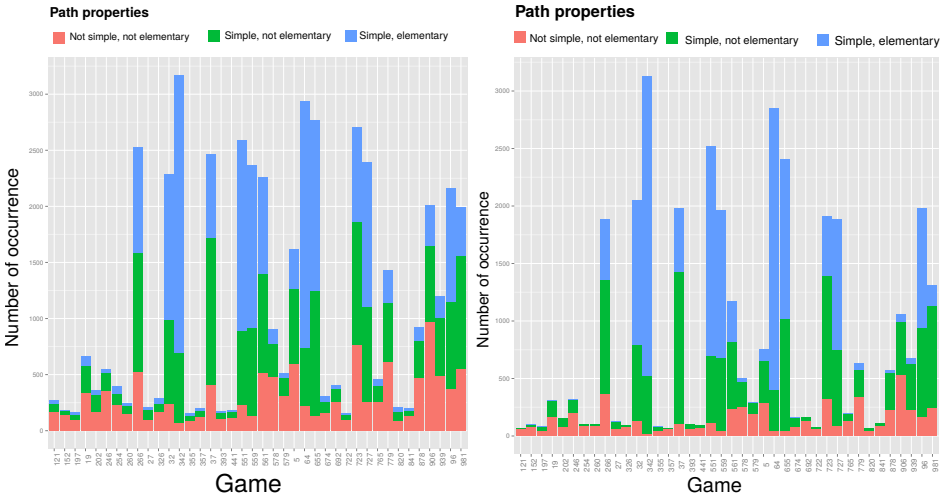


Figure 14: A visualization of the problem space of game 266 (the corresponding configuration is depicted in figure 11) and the all paths given in the data set within. The start configuration is the green node on the right, final states are blue. Nodes which do not occur in any of the paths are grey, nodes in which a user quit to solve the game are colored red, all other nodes which occur in any path are yellow.



(a) For all paths.

(b) For the cut paths.

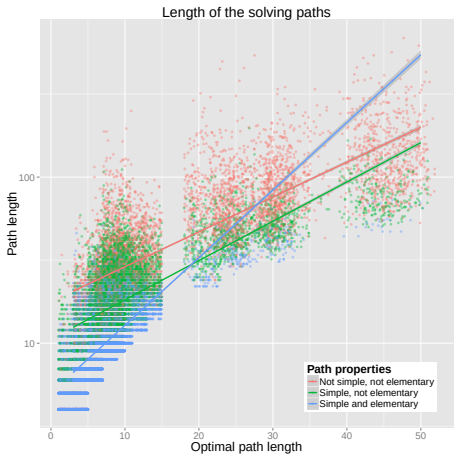
Figure 15: An overview of how many of the extracted paths are simple and/or elementary, as defined in section 1.3.

the extracted paths are neither simple nor elementary. Nevertheless, it is striking that the smaller the number of available paths is, the larger is the fraction of paths which are neither simple nor elementary. Comparing figure 15a and 15b makes clear that cutting the paths – and hereby removing short paths and removing the last part of the long paths – does not significantly change the proportions of how many paths are elementary and/or simple.

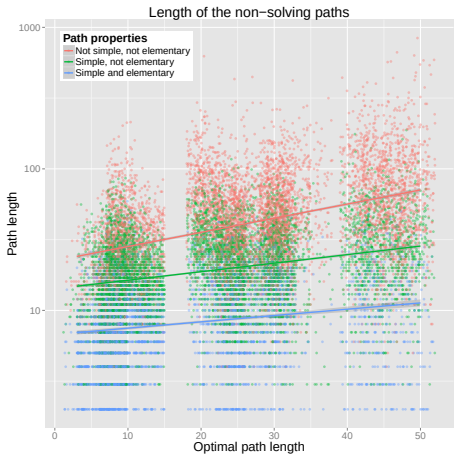
Figure 16 shows the relationship of the length of the available paths with the length of the shortest path from the start configuration to a final state. Figure 16a and 16a show this relationship for all paths, once for the solving paths, once for the non-solving paths. Figure 16c and 16c contains the same for the cut paths. It is not surprising that the length of the extracted paths for the different configurations increases with increasing length of the optimal path for the corresponding configuration. However, especially in figure 16a and 16c, it is noticeable that simple and elementary paths only occur in problem spaces with a rather short optimal solution path. The longer the optimal solution path becomes, the more probable it is that the found solution paths are not elementary anymore. For configurations with an optimal solution longer than 30 steps, there are almost no paths anymore which solve the game and are elementary and simple.

We want to analyze the available paths for more properties than length and being simple and/or elementary. Therefore, for a path, we introduce the following properties among which the first four were already mentioned:

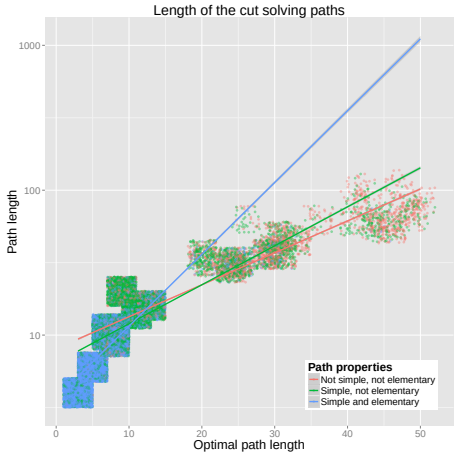
- *Length* is the number of (not necessarily distinct) contained edges in the path.
- *isSolved* indicates whether the path contains a final state or not.
- *isSimple* indicates whether the path is simple
- *isElementary* indicates whether the path is elementary
- *MaxNodeVisitation* is the maximum value of the node visitation values for all nodes contained in the node set of the path, where the node visitation is the number of times this node is contained in the path.



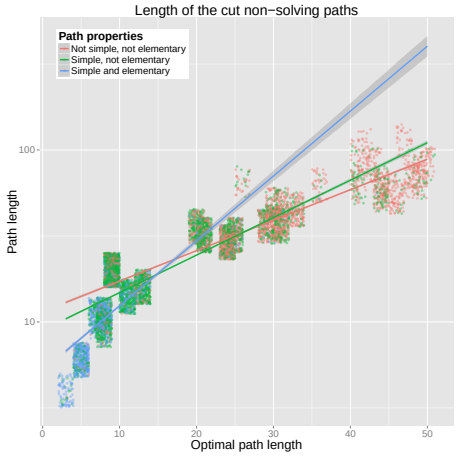
(a) For all solving paths, with linear regression lines for each type of paths.



(b) For all non-solving paths, with linear regression lines for each type of paths.



(c) For all cut solving paths, with linear regression lines for each type of paths.



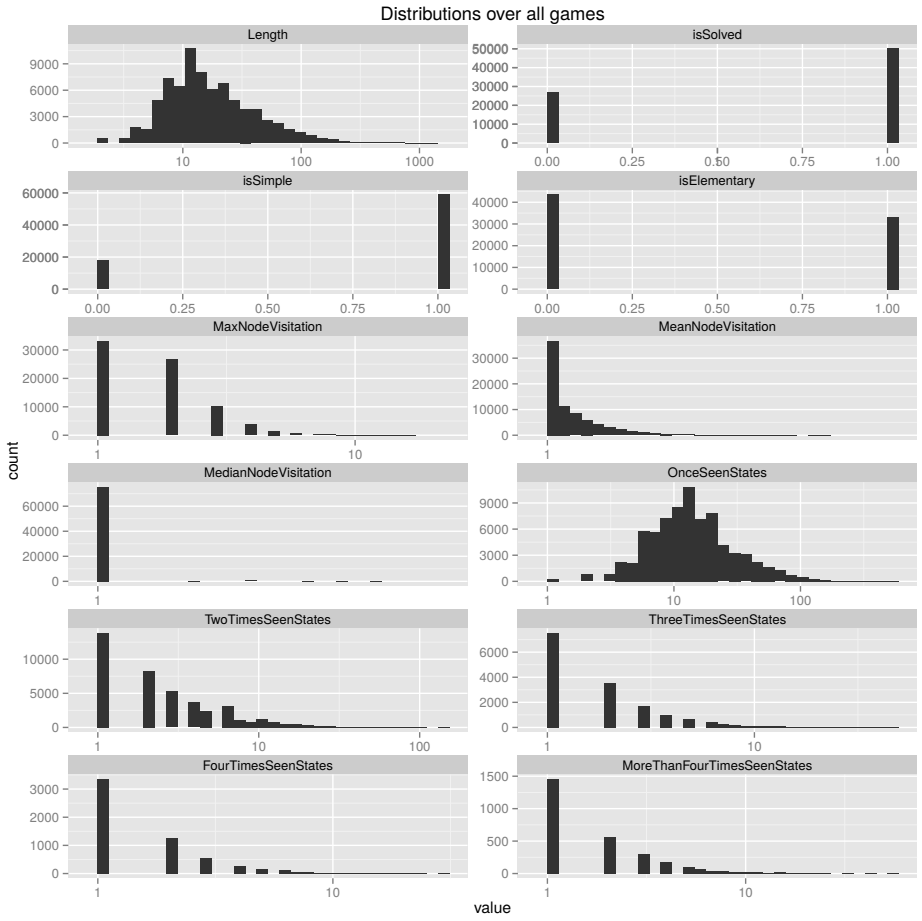
(d) For all cut non-solving paths, with linear regression lines for each type of paths.

Figure 16: The relationship of the length of the extracted paths to the length of the optimal path in the respective problem space, once for the solving and non-solving paths. The colour encodes whether the path is simple and/or elementary. Note the logarithmic scale on the vertical axis and note that for the lower two figures, the paths of one configuration were cut to equal length, which is why all paths of one configuration are jittered in horizontal and vertical direction in these two plots.

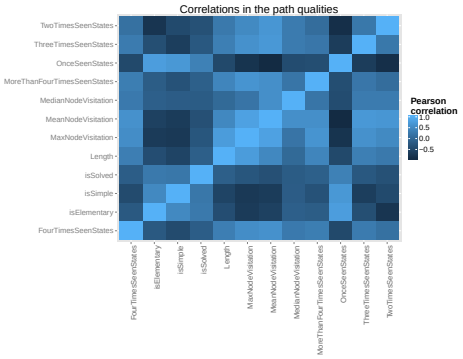
- *MeanNodeVisitation* is the mean value over all node visitation values of all nodes contained in the node set of the path
- *MedianNodeVisitation* is the median of the node visitation values for all nodes contained in the node set of the path
- *OnceSeenStates* is the number of nodes in the node set of the path which are contained exactly once in the path
- *TwoTimesSeenStates* is the number of nodes in the node set of the path which are contained exactly twice in the path
- *ThreeTimesSeenStates* is the number of nodes in the node set of the path which are contained exactly three times in the path
- *FourTimesSeenStates* is the number of nodes in the node set of the path which are contained exactly four times in the path
- *MoreThanFourTimesSeenStates* is the number of nodes in the node set of the path which are contained more than four times in the path

Figure 17 shows how these path properties are distributed in the available data set. In figure 17a, for each of the introduced properties, its distribution over all available (uncut) paths is shown. The scales on the horizontal axes are logarithmic, except for the scales on the plots for the binary properties (i.e. *isSolved*, *isSimple* and *isElementary*). It can be seen that the distribution of the length and the number of exactly once seen states is almost the same and has a peak around 10. Additionally, there are about twice as many solved paths than not solved paths, about two third of the available paths are simple, almost half of the paths is elementary. All the other properties which are related to the concept of node visitation, show a maximal occurrence at value 1 and the occurrence decreases for an increasing value.

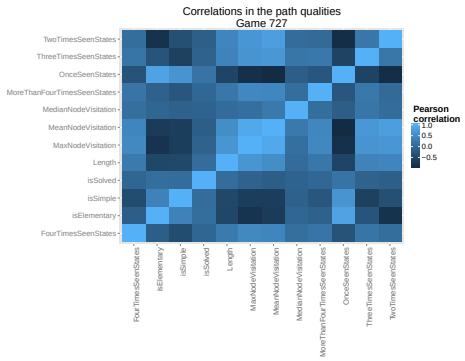
In order to see whether the proposed properties are related to other, the correlation of all pairs of the properties is computed, for all paths of all games and separately for each game. Figure 17b shows the values of the Pearson correlation coefficient of all pairs of properties over all paths of all configurations, figure 17c shows the respective values for an exemplary configuration, both visualized in a heat map where a light blue corresponds to a value of 1, i.e. a high positive correlation, and a dark blue corresponds to a value of -1, i.e. a negative correlation. It is striking that the two heat maps (and also the heat maps for the remaining games which are not shown) have the same colour patterns. This indicates that the relations of the properties are similar for all configurations and do not differ much for the different configurations. Some of the entries of the heat map are striking: The properties *isSimple* and *isElementary* are negatively correlated with the properties *Length*, *MaxNodeVisitation*, *MeanNodeVisitation* and *TwoTimesSeenStates* which is plausible because as soon as a path has a value greater than 1 in any of the latter properties, the path can not be elementary and is probably also not simple. Accordingly, the properties *isSimple* and *isElementary* show a positive correlation with the property *OnceSeenStates*, since: the more states are exactly used once, the less states are used more than once (this negative correlation is also recognizable), and the higher the probability that the path simple or even elementary. Furthermore, the length of a path is positively correlated with the properties *MaxNodeVisitation* and *MeanNodeVisitation* which is an interesting observation and is consistent with figure 16: the longer a path



(a) The distribution of the introduced path properties over all games. Note the logarithmic scale on all horizontal axes except of the ones for the binary attributes.



(b) The Pearson correlation of the path properties of all games, visualized in a heat map.



(c) The Pearson correlation coefficient for the path properties of all paths for a particular configuration, visualized in a heat map.

Figure 17: An overview of the properties of the available path data.

is, the higher the probability that nodes are used more than once and the path is not optimal anymore.

4.3 USED ALGORITHMS

In this section, the used algorithms, tools and methods in the implementations are described. The following tasks were implemented for this thesis:

- *Computation of the state spaces* The problem spaces for the set of configurations are computed by a breadth-first search approach: having implemented the game logic of *Rush Hour*, for each configurations all possible successor configurations (all states that can be reached within one allowed move) can be enumerated. Starting from the start configurations, the whole state space can be explored by a breadth-first search. We here assume that a solution state does not have any successor states, therefore, states which can only be reached after a final state, are excluded from the state space.
- *Visualization of the state spaces and the contained paths* For the visualization of the state space and the contained paths as it can be seen in figure 14, we implemented a method which writes out the computed state spaces and extracted paths in *gml*-format. The layout of the graph is then done by the graph editing tool *yEd*[3].
- *Computation of the proposed similarity measures for the extracted paths* The implementation of most of the introduced similarity and distance measures for paths is straightforward. For the implementation of the matched average distance, for each node of the longer path, the distance to the closest node of the shorter path needs to be found. Naively, this can be done by starting a breadth-first-search from each node of the longer path which is terminated as soon a node from the other path is discovered. Furthermore, the breadth-first-search can keep track of the distance of the explored nodes from the starting node. Further improvement of this algorithm is left for future work. The computation of the LCSS similarity can be done by using a dynamic programming approach described by Cormen [6].
- *Clustering of the paths* Clustering of the paths based on the pair-wise similarities and distances is done by the *hclust* method from the *R*-package *stats*.

4.4 RESULTS OF CLUSTERING PATHS

We computed the similarity and distance measure, proposed in section 3.3, for all available paths described in section 4.2. We restricted the analysis to the paths which were cut to equal length as described before. The goal for future work with this data will be to group the paths in groups according to their similarity in order to be able to distinguish paths with different properties, for example whether the path describes a solved game or not. As a first approach, we present the following results which are still preliminary and need further refinement.

3 see for example here: https://www.yworks.com/

4.4.1 *Method*

For each similarity and distance measure and for each game, a matrix is computed which contains the measure's values for each pair of available paths for the respective game. Furthermore, for each path, there are several binary properties known: whether the path is solved or not, whether it is simple, and whether it is elementary. As a first simple approach to cluster the available paths of each game, we used a hierarchical clustering approach. The matrices with the similarity and distance measure values for each game are preprocessed:

(i) the normalized distance measure values are left unchanged.

(ii) the unnormalized distance measure values are scaled to the interval $[0,1]$ by the following transformation: let $D^c \in \mathbb{R}^{n\times n}$ be the matrix containing the distance values for the n paths of a configuration $c \in \mathcal{C}$. We transform the matrix to the matrix D^c_n by substituting each entry D^c_{ij} of the matrix D^c by the value

$$\frac{D^c_{ij} - \min_{k,l\in\{1,\dots,n\}} D^c_{kl}}{\max_{k,l\in\{1,\dots,n\}} D^c_{kl} - \min_{k,l\in\{1,\dots,n\}} D^c_{kl}}$$

and then get $D^c_N \in [0,1]^{n\times n}$.

(iii) the normalized similarity measure values are transformed to the associated distance value: let $S^c \in [0,1]^{n\times n}$ be the matrix containing the similarity values for the n paths of a configuration $c \in \mathcal{C}$. We transform the matrix by substituting each entry S^c_{ij} by $1 - S^c_{ij}$ and get the matrix S^c_N.

(iv) the unnormalized similarity measure is first scaled to the interval of $[0,1]$, then transformed to its associated distance measure, i.e. let $S^c \in \mathbb{R}^{n\times n}$ be the matrix with the similarity measure values for the n paths of the configuration $c \in \mathcal{C}$. We transform the matrix by substituting each entry S^c_{ij} by

$$1 - \frac{S^c_{ij} - \min_{k,l\in\{1,\dots,n\}} S^c_{kl}}{\max_{k,l\in\{1,\dots,n\}} S^c_{kl} - \min_{k,l\in\{1,\dots,n\}} S^c_{kl}}$$

and get the matrix $S^c_N \in [0,1]^{n\times n}$.

The therefore obtained matrices all contain values from the interval $[0,1]$ which are close to 0 if the respective paths are very close (similar) to each other and close to 1 if the paths are very distant (dissimilar) to each other – in the respective measure.

The distributions of the measures for an exemplary game can be found in figure 18 and a summary of the values in table 5. There can be made several observations:

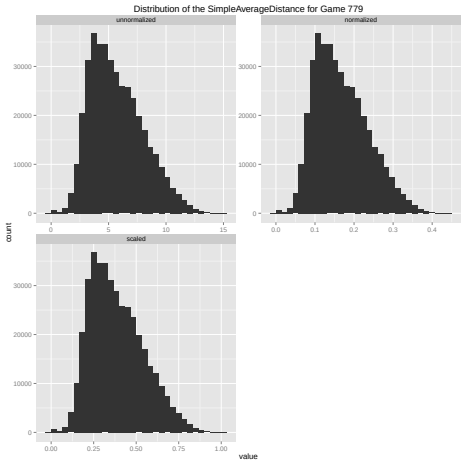
- For the distance measures, the distributions of the normalized, the unnormalized, and the scaled measure are completely the same – except for the scale. This is not surprising, since these two measures are normalized by the diameter of the graph, therefore, for each pair of paths, the value of the distance measure is divided by the same number. As expected, scaling by the above formula does not change the distribution.

- It is remarkable that the distributions for the two distance measures look so different: while the distribution for the simple average distance is almost symmetric, the distribution for the matched average distance is much skewed to the left. It is surprising that the matched average distance – although it is supposed to yield a better matching of the path nodes – takes smaller values for the majority of the pairs of paths than the simple average distance. Almost all pairs of paths have a matched average distance value smaller than 5 while approximately half of the pairs of paths have a simple average distance value larger than 5 (both in the unnormalized case).
- The distributions for the similarity measures look different. Not surprisingly, for each of the similarity measures, the distributions for the normalized measure and the inverted measure are the same, but mirrored, which is due to their computation. The same holds for the distributions for the unnormalized measure and the scaled and inverted measure, except that for these cases, the scales are different. But since the scaled and inverted measure is computed by scaling the unnormalized measure to the interval $[0, 1]$ and inverting it, this is consistent with the shown distributions.
- The fact that for the similarity measures the distributions of the normalized and inverted measure are different to the distributions of the unnormalized measure and the scaled and inverted measure is due to the fact the normalization is done by a value which is specific for each pair of paths. This explains the differences in the distributions.
- Although the LCSS similarity is also normalized by a path pair specific values, namely the length of the longer path, we do not get a change in the distribution here – only the scale changes.
- It is remarkable that the node set similarity and the LCSS similarity show similar characteristics, contrarily to the edge set similarity.

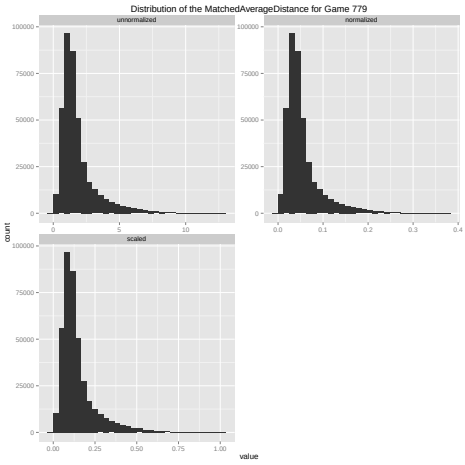
These matrices are the input for an hierarchical clustering algorithm with complete and average linkage. For each game and each obtained matrix, the algorithm (see section 4.3 for information about used algorithms) builds a dendrogram representing the clusters in different stages of the algorithm and which could then be cut at a certain height in order to obtain the desired number of clusters. In future work, it could be a worthwhile approach to consider other clustering and classification approaches.

4.4.2 *Evaluation of the clustering results*

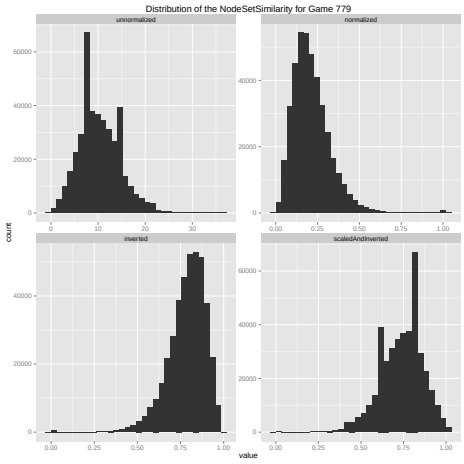
The computed dendrograms of the clustering algorithm with complete linkage can be found in figure 19 and 20, the respective dendrograms of the algorithm using average linkage can be found in the appendix. As a first observation, it can be seen that there is almost no difference whether the normalized or the unnormalized distance measure is taken as input for the algorithm, the resulting dendrograms are almost the same. Though, visualizing the clustering results as dendrograms might be not the right choice to get any insight of the path data, since the number of paths is too large. Thus, merely looking at the dendrograms as they are, will not bring any further knowledge.



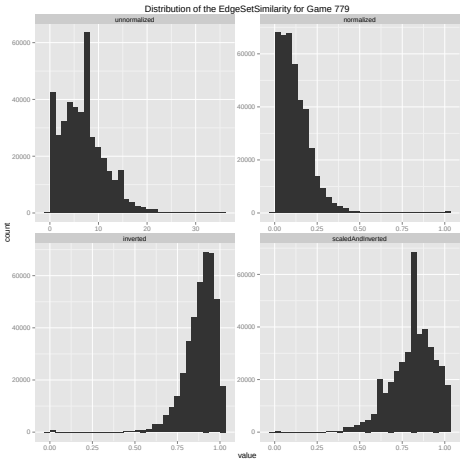
(a) Simple average distance



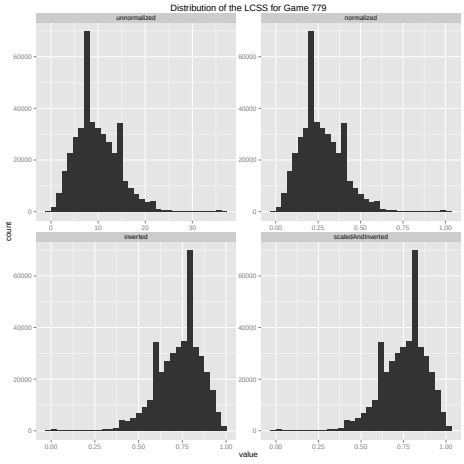
(b) Matched average distance



(c) Node set similarity



(d) Edge set similarity

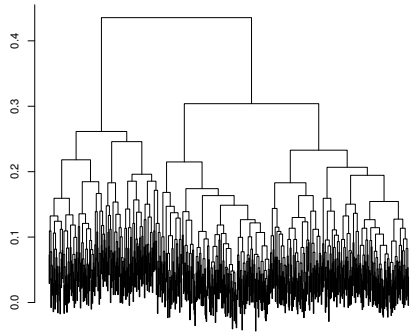


(e) LCSS similarity

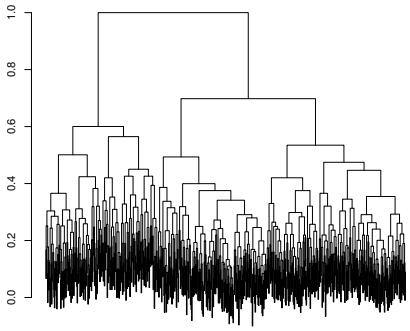
Figure 18: The distribution of the values of the computed similarity and distance values for the game 779. For the distance measures, the figure contains the distribution for the unnormalized distance measure, the normalized distance measure, and the distance measure scaled to $[0, 1]$ by the described transformation. For the similarity measures, the figures contain the unnormalized similarity measure, the normalized similarity measure, the inverted normalized similarity measure, and the inverted and scaled unnormalized similarity measure (see section 4.4.1 for the details).

MEASURE	MIN	MAX	MEAN	MEDIAN	SD
Node set similarity					
unnormalized	1	36	10.21	10	4.36
normalized	0.01	1	0.21	0.2	0.11
inverted	0	0.99	0.79	0.8	0.11
scaled and inverted	0	1	0.74	0.74	0.12
Edge set similarity					
unnormalized	0	35	6.76	6	4.48
normalized	0	1	0.12	0.1	0.09
inverted	0	1	0.88	0.9	0.09
scaled and inverted	0	1	0.81	0.83	0.13
LCSS similarity					
unnormalized	1	36	9.7	9	4.61
normalized	0.03	1	0.27	0.25	0.13
inverted	0	0.97	0.73	0.75	0.13
scaled and inverted	0	1	0.75	0.77	0.13
Simple average distance					
unnormalized	0	14.81	5.72	5.36	2.31
normalized	0	0.44	0.17	0.16	0.07
scaled	0	1	0.39	0.36	0.16
Matched average distance					
unnormalized	0	12.61	1.84	1.44	1.39
normalized	0	0.37	0.05	0.04	0.04
scaled	0	1	0.15	0.11	0.11

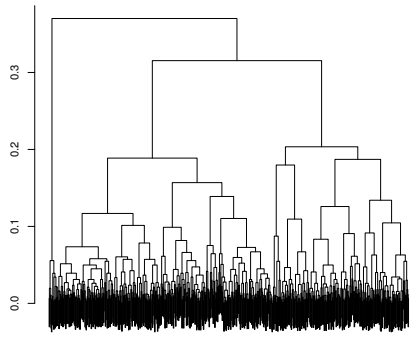
Table 5: A summary of the similarity and distance measure values for the available (cut) paths of game 779.



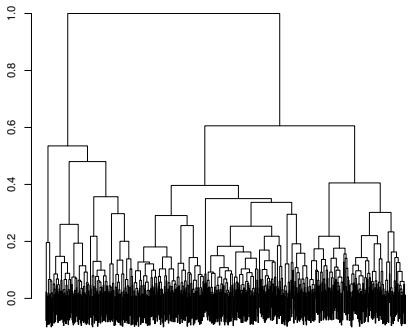
(a) Normalized simple average distance.



(b) Unnormalized simple average distance.

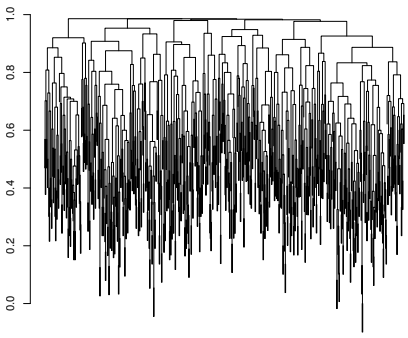


(c) Normalized matched average distance.

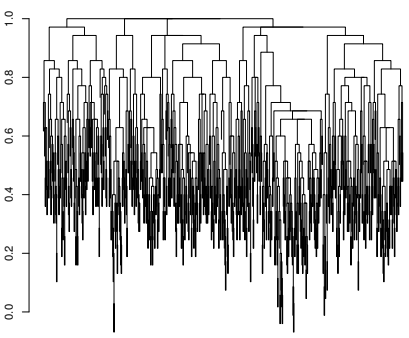


(d) Unnormalized matched average distance.

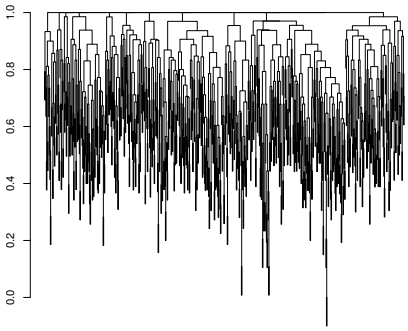
Figure 19: The dendrograms visualizing the hierarchical clustering results using complete linkage for the game 779 and each of the path distance measures.



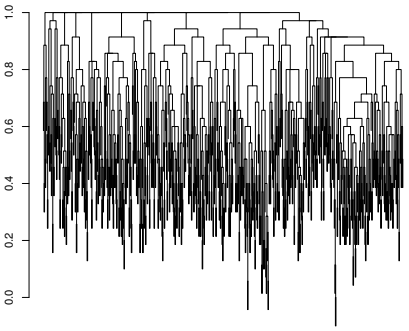
(a) Normalized node set similarity.



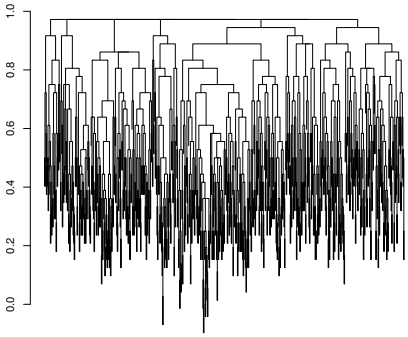
(b) Unnormalized node set similarity.



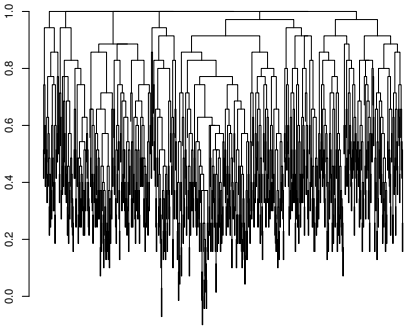
(c) Normalized edge set similarity.



(d) Unnormalized edge set similarity.



(e) Normalized LCSS similarity.



(f) Unnormalized LCSS similarity.

Figure 20: The dendrograms visualizing the hierarchical clustering results using complete linkage for the game 779 and each of the path similarity measures.

In general, evaluating the clustering results is not an easy task at all, especially for the following reasons:

- The main point of clustering the path is an evaluation of the proposed similarity and distance measures for paths – and not an evaluation of the *clustering method*. How can it be differentiated between these two in an evaluation?
- The main problem for an evaluation of the measures and the clustering is the fact that there is no ground truth available for the given data or any available path data. This is due to the fact that there does not exist any work that is concerned with the clustering of paths in graphs. In future work, it will be a necessary task to generate a ground truth for path data which can then be used to do an extensive evaluation of the clustering results. In order to do so, it will be unavoidable to develop an appropriate and meaningful visualization of clustered paths in a graph which can then serve for a first visual inspection of the clustering results.
- Paths have several properties by which the clustering algorithm might be supposed to divide the paths into several groups. It might be that the same similarity measures shows totally different performances when considering different properties of the paths as criterion for evaluating the similarity measure. This, however, is an expected and desired behavior, otherwise, it would not be necessary to consider different similarity measures if they all capture the same aspect of the paths. So, evaluation of the measures and the clustering results needs to carefully consider which property the measure is intended to capture in order to evaluate this aspect.

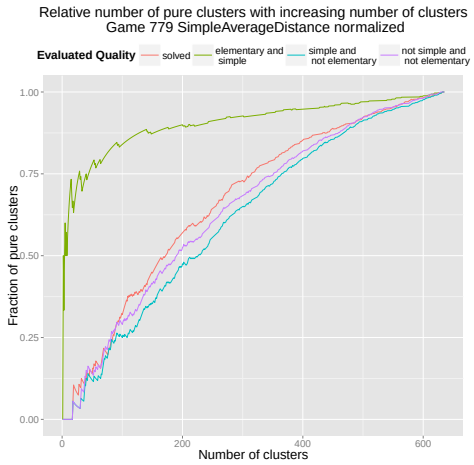
Since developing an evaluation scheme which considers all the above mentioned concerns about evaluation of the clustering results, goes beyond the scope of this work, we propose a simple approach for evaluation which is based on the *purity* of clusters. Let $\mathcal{Q} = \{q : \mathcal{P}_V \to \{0,1\}\}$ be a set of binary properties which a path $p \in \mathcal{P}_V$ can either fulfill, i.e. $q(p) = 1$, or not fulfill, i.e. $q(p) = 0$, for a quality $q \in \mathcal{Q}$. We call a cluster *pure regarding* q if q has the same value for each path in the cluster, formally, a cluster of paths $C = \{p_1, \ldots, p_l\} \subseteq \mathcal{P}_V$ is called pure regarding $q \in \mathcal{Q}$ if either $\sum_{i=1}^{l} q(p_i) = 0$ or $\sum_{i=1}^{l} q(p_i) = l$ holds.

Furthermore, we define the *purity* of a cluster C regarding q as

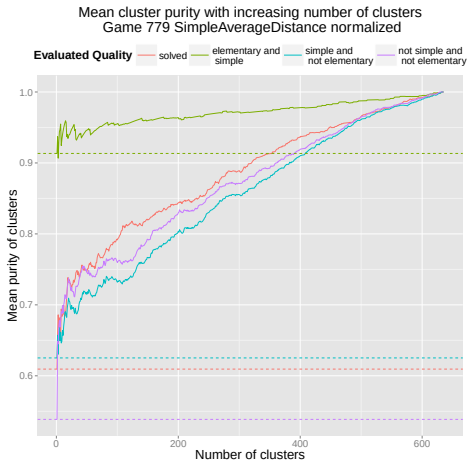
$$purity_q(C) = \frac{1}{l} \max \left\{ \sum_{i=1}^{l} q(p_i), l - \sum_{i=1}^{l} q(p_i) \right\}.$$

We now consider the purity of the clusters in the different stages of the hierarchical clustering process. Having the dendrogram which represents the hierarchical clustering process, an arbitrary number of clusters between 1 (all paths are in the same cluster) and n, with n the number of available paths (each path is in its own cluster of size 1), can be chosen. We consider now the purity of the clusters with increasing cluster number regarding certain path properties. The qualities that we take into account are (i) whether the path is solving or not, (ii) whether it is simple and elementary, (iii) whether it is simple, but not elementary, and (iv) whether it is not elementary and not simple.

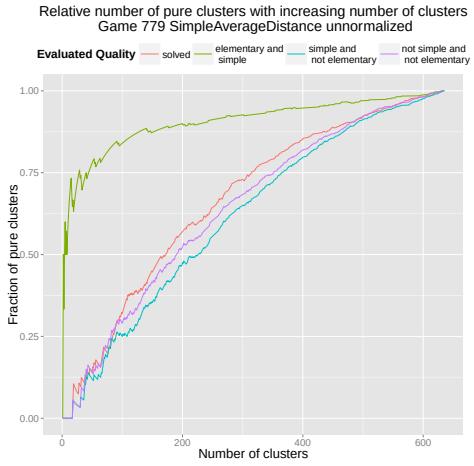
For each similarity measure and distance measure, each game and each of the proposed qualities, we consider the process of hierarchical processing in the



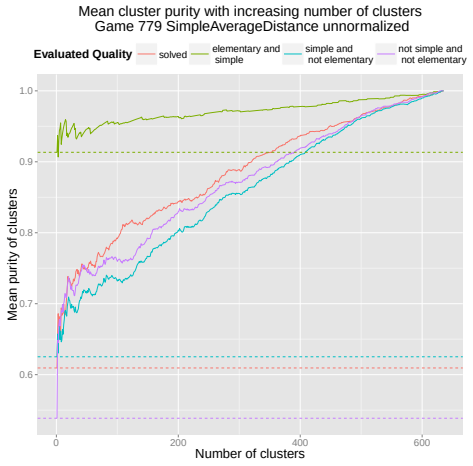
(a) Normalized simple average distance.



(b) Normalized simple average distance.

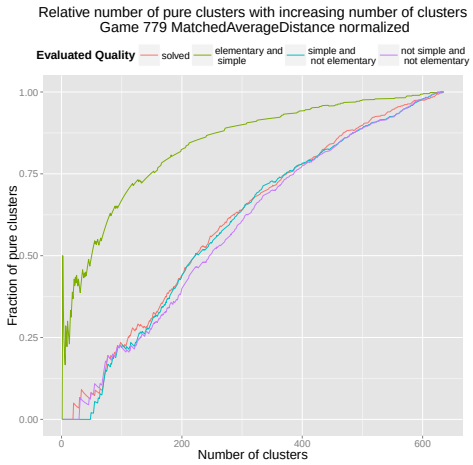


(c) Unnormalized simple average distance.

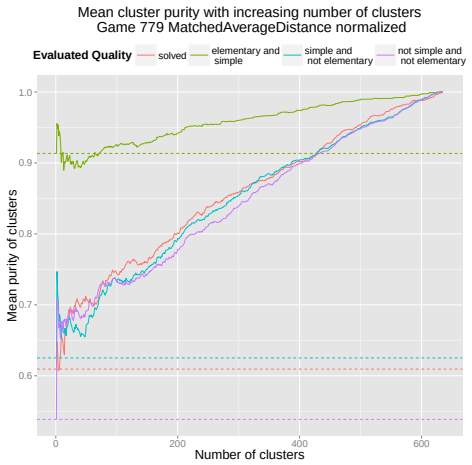


(d) Unnormalized simple average distance.

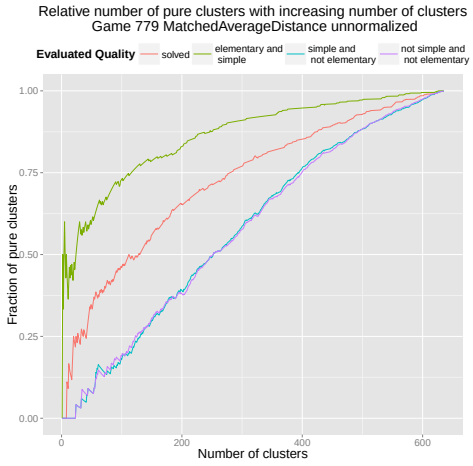
Figure 21: The number of pure clusters and the average purity of the clusters with increasing number of clusters for each of the proposed qualities and the simple average distance. The dashed lines indicate the fraction of the paths which satisfy the respective quality relative to the total number of paths. For each plot, the colors encode the evaluated quality: red – solved; green – elementary and simple; blue – simple and not elementary; purple – not simple and not elementary.



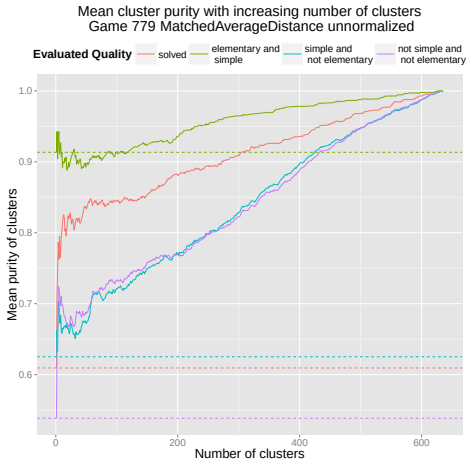
(a) Normalized matched average distance.



(b) Normalized matched average distance.

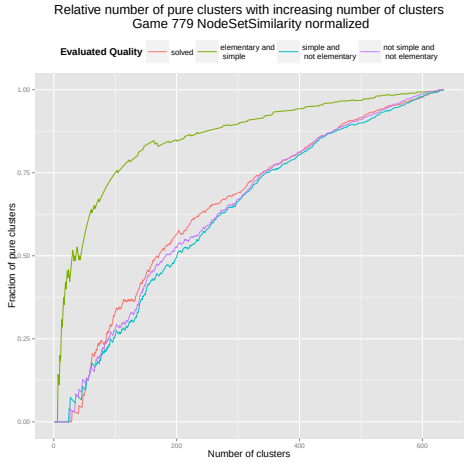


(c) Unnormalized matched average distance.

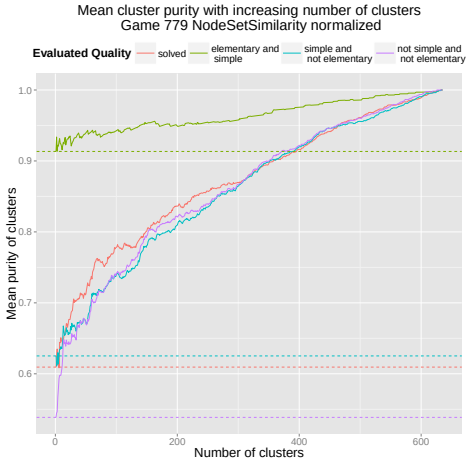


(d) Unnormalized matched average distance.

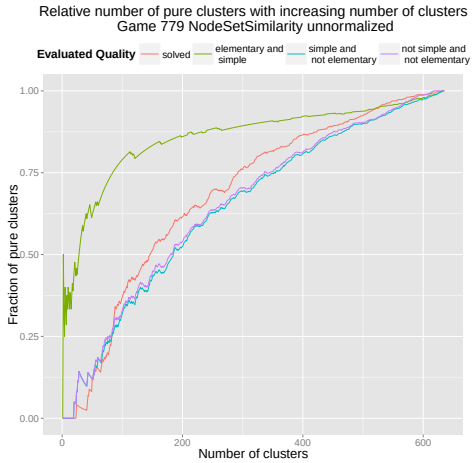
Figure 22: The number of pure clusters and the average purity of the clusters with increasing number of clusters for each of the proposed qualities and the matched average distance. The dashed lines indicate the fraction of the paths which satisfy the respective quality relative to the total number of paths. For each plot, the colors encode the evaluated quality: red – solved; green – elementary and simple; blue – simple and not elementary; purple – not simple and not elementary.



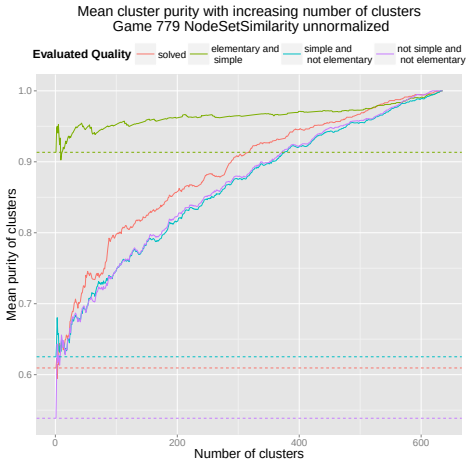
(a) Normalized node set similarity.



(b) Normalized node set similarity.

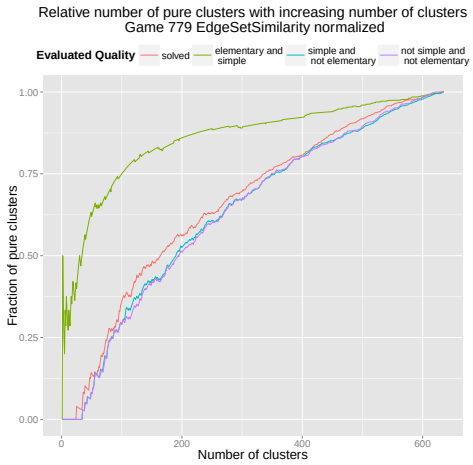


(c) Unnormalized node set similarity.

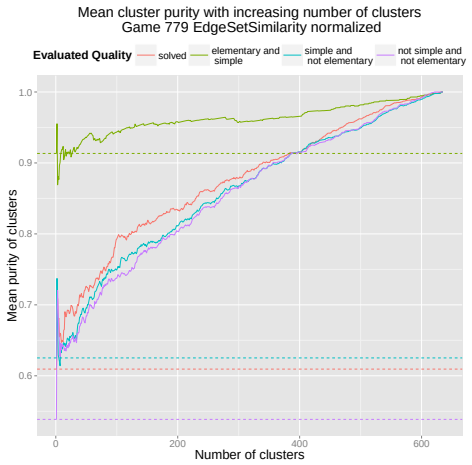


(d) Unnormalized node set similarity.

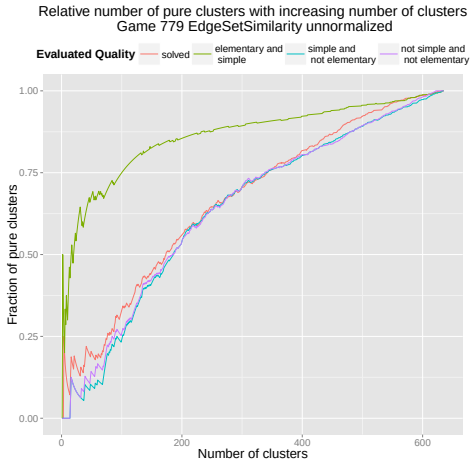
Figure 23: The number of pure clusters and the average purity of the clusters with increasing number of clusters for each of the proposed qualities and the node set similarity. The dashed lines indicate the fraction of the paths which satisfy the respective quality relative to the total number of paths. For each plot, the colors encode the evaluated quality: red – solved; green – elementary and simple; blue – simple and not elementary; purple – not simple and not elementary.



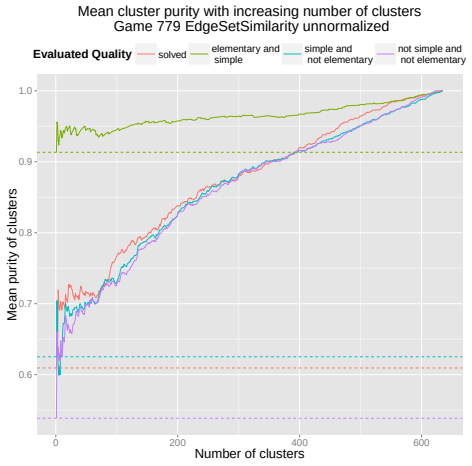
(a) Normalized edge set similarity.



(b) Normalized edge set similarity.

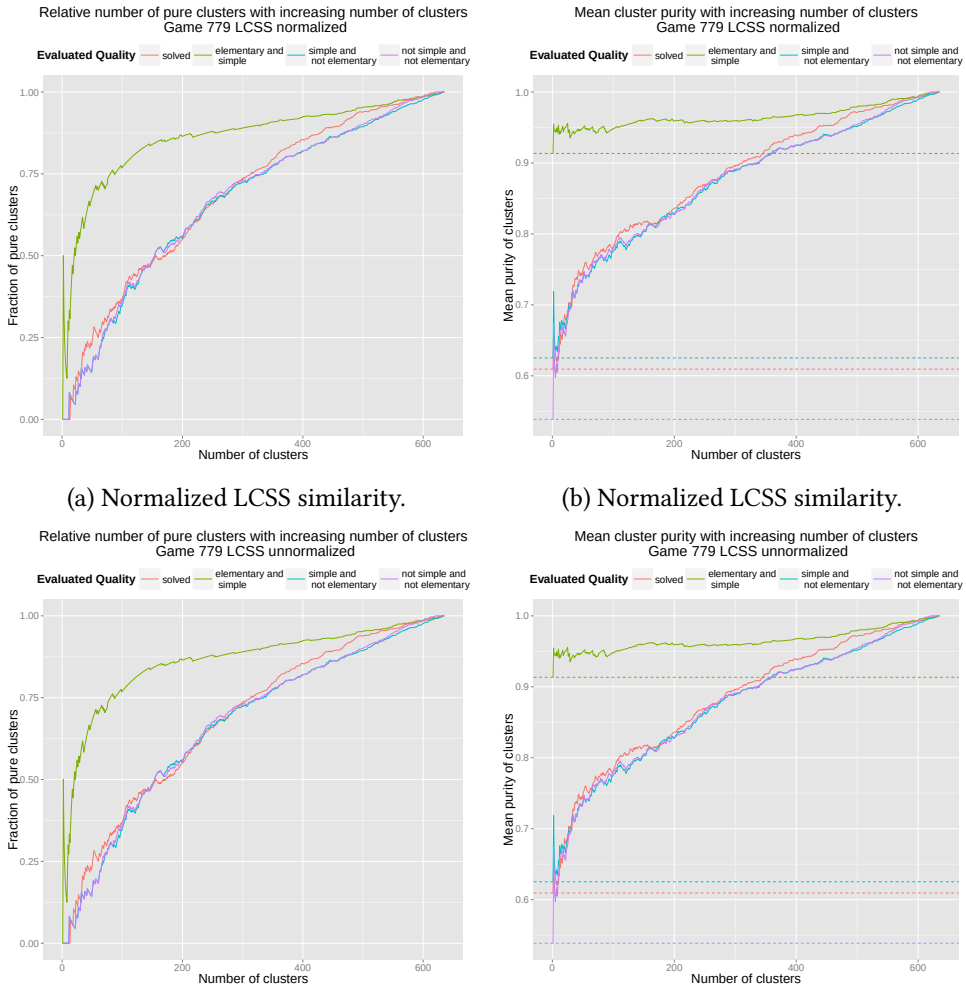


(c) Unnormalized edge set similarity.



(d) Unnormalized edge set similarity.

Figure 24: The number of pure clusters and the average purity of the clusters with increasing number of clusters for each of the proposed qualities and the edge set similarity. The dashed lines indicate the fraction of the paths which satisfy the respective quality relative to the total number of paths. For each plot, the colors encode the evaluated quality: red – solved; green – elementary and simple; blue – simple and not elementary; purple – not simple and not elementary.



(a) Normalized LCSS similarity.

(b) Normalized LCSS similarity.

(c) Unnormalized LCSS similarity.

(d) Unnormalized LCSS similarity.

Figure 25: The number of pure clusters and the average purity of the clusters with increasing number of clusters for each of the proposed qualities for the LCSS similarity. The dashed lines indicate the fraction of the paths which satisfy the respective quality relative to the total number of paths. For each plot, the colors encode the evaluated quality: red – solved; green – elementary and simple; blue – simple and not elementary; purple – not simple and not elementary.

sense that the number of clusters is increased in each step – the height on which the dendrogram is cut in order to obtain the clusters is lowered with each step. In the first step, the dendrogram is cut at a height such that there is only one cluster which contains all paths, in the second step such that there are exactly two clusters, and so forth. It is clear that in the first step, all existing clusters are not pure – assuming that there is no trivial quality, i.e. a quality in which all considered paths have the same value. In the last step, all clusters are pure since each cluster contains exactly one path. The question is then if there is a similarity or distance measure which is – in combination with the hierarchical clustering algorithm – able to preserve a high number of pure clusters with decreasing number of clusters, or which is able to achieve a high number of pure clusters with a small number of clusters. This relationship for one exemplary game is depicted in the left columns of figure 21, 22, 23, 24 and 25 for each of the similarity and distance measures. The increasing number of clusters in each step is plotted on the horizontal axis, the fraction of pure clusters to the total number of clusters is plotted on the vertical axis. It can be observed that – as expected – the number of pure clusters increases with increasing number of clusters, for each measure and for each quality, though not monotonically. Furthermore, all measures have in common that the number of pure clusters regarding the quality of being simple and elementary increases considerably faster with increasing number of clusters than all other qualities. Especially for the simple average distance, for both the normalized and unnormalized version, the number of pure clusters rises steeply until a percentage of 75 % is reached with 50 clusters, before the curve flattens. For the remaining measures, the value of 75 % is reached considerably later, but the rate of growth for this quality is qualitatively the same for all measures. Yet, needing approximately 50 clusters for 635 paths in order to get 75 % pure clusters is still not a satisfying result.

Interestingly, although the quality of being solved is correlated with neither the quality of being simple or elementary (cf. section 4.2), the curves of the number of pure clusters show a similar behavior for these qualities for almost all measures. For all measures except the unnormalized matched average distance, the remaining three qualities are almost identical with regard to the number of pure clusters and show an approximately linear growth. Only for the unnormalized matched average distance, the number of pure clusters regarding the quality of being solved grows faster than regarding the other two qualities. Therefore, it might be a worthwhile approach to analyze this measure more closely and to build a more sophisticated classifier based on the matched average distance in order to distinguish between solved and not solved paths by comparing the existing paths.

Additionally, it is remarkable that a perfect clustering, i.e. a value of almost 100 % of pure clusters, is not reached until the maximum number of possible clusters.

However, the evaluation measure of the number of pure clusters is a very strict one: independent of the size of the cluster, as soon as only one "wrong" path is added to the cluster, the cluster is not pure anymore. For a large number of path, it might not be appropriate to give such a high weight to a single path. It might be more desirable to allow some flexibility and allow a certain

rate of misclassified paths in a cluster. For this reason, we consider the average purity of the clusters in the process of the hierarchical clusters. For each measure, each game and each quality, we consider the mean purity of the clusters with increasing number of clusters. This relationship can be found in the right column of the figures 21, 22, 23, 24 and 25: the horizontal axis again contains the number of clusters, on the vertical axis, the average purity of the existing clusters (as defined above) is plotted. The dashed lines indicate how many of the available paths satisfy the respective quality, i.e. for a quality q and all available paths $P = \{p_1, \dots, p_k\} \subseteq \mathcal{P}_V$, the dashed line is plotted at the value $v_q = \max\left\{\sum_{i=0}^{k} q(p_i), \sum_{i=0}^{k} 1 - q(p_i)\right\}$. It follows from the definition that the mean purity of the clusters is always greater or equal 50 %. Furthermore, it is obvious that for any nontrivial quality, the mean purity is v_q if there is only one cluster which contains all paths, and is 1 if there is exactly one cluster for each path.

It can be observed that these figures are consistent with the plots in the left columns: For all similarity measures, the mean purity is highest for the quality of being elementary and simple. Though, this can be explained by the fact that there are only very few paths which are elementary and simple, more than 90 % of the paths are either not simple or not elementary. The remaining qualities are rather equally distributed over the paths, i.e. the fraction of paths which fulfill (or do not fulfill) the other three qualities is between 50 % and 65 %. The clusters' mean purity regarding the quality of being simple and elementary grows the least fast for increasing number of clusters, there is actually a range in each of the figures in which the mean purity regarding this quality is nearly constant. Furthermore, for the unnormalized and normalized matched average distance as well as for the normalized edge set similarity (and – to a lesser extent – for the unnormalized node set similarity), it can be observed that the mean purity of the clusters regarding the quality of being simple and elementary drops in the very beginning, even drops below the dashed line and does not exceed this value for the next steps. In the case of the unnormalized matched average distance, the mean purity does not exceed the respective value until the number of clusters is greater than 100.

As in the left column of the figures, the mean purity regarding the qualities of being solved, of being neither simple nor elementary, and of being simple and not elementary, shows a very similar behavior for all similarity and distance measures – except for the unnormalized matched average distance. For this distance measure, the mean purity regarding the quality of being solved increases steeply in the very first steps and exceeds the value of 80 % mean cluster purity with less than 20 different clusters. This mean purity regarding the quality of being solved is not reached by the other measures with a number of clusters smaller than 100 to 200. For the remaining measures, the mean purity regarding these three qualities grows nearly in the same way and also – as in the left column – approximately linearly.

Additionally, it can be noticed that except for the described initial drops of the mean purity regarding the quality of being simple and elementary in the mentioned four measures, the mean purity for each measure and for each quality

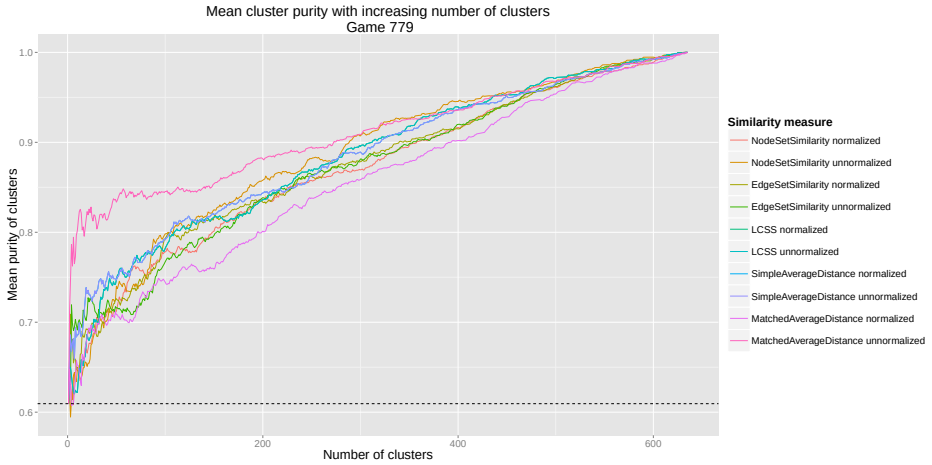


Figure 26: An aggregation of the mean cluster purity regarding the quality whether the paths is solving or non-solving, for each of the proposed similarity and distance measures for the example game 779. The dashed line indicates the fraction of paths which are solving in relation to the total number of paths.

increases with increasing number of clusters and is always considerably above the value v_q for the respective quality.

The most interesting quality for which the clusters should be as pure as possible is certainly the quality whether the path is solving or non-solving. In order to compare the different similarity and distance measures and to compare the purity of the clusters regarding this quality which are computed based on the measures, figure 26 contains an aggregation of the clusters' mean purity for all distance and similarity measures. The different curves represent the clusters' mean purity with increasing cluster number, one curve for each similarity and distance measure. The dashed line is again the value v_q for the quality of interest q. It can be observed that most of the measures show a similar behavior for the clusters' purity with an increasing number of clusters, however, the unnormalized matched average distance is the only measure which is able to produce clusters with a mean purity of 80 % with only a few clusters. But nevertheless, also the other measures yield clusters which have a considerably higher purity than they would have by chance.

5

SUMMARY

5.1 SUMMARY OF THE WORK

This section gives to give a brief summary of the present work before the next section will give an outlook for possible future work.

To the best of our knowledge, the present work is the first approach which is concerned with the similarity of paths in graphs. Although there is published research about similarity of trajectories in a Euclidean space or of time series, there does not exist any approach which systematically analyses similarity measures of paths.

The present work aims at giving a starting point for the research in the area of path similarity in graphs. For this reason, it is structured in two main parts. The first part provides theoretical foundations for the second part in which the measures developed in the first part, are applied to real-world data and evaluated. The theoretical part follows an axiomatic approach by starting with the proposal of possible properties a similarity or distance measure for paths could have. Some of these properties are well-known properties of generic distance metrics, for example coincidence, symmetry, or the triangle inequality. We additionally introduce four properties which are specifically developed for distance and similarity measures of paths in graphs. Having identified possible properties of *measures*, we identify possible features of *paths* which could be captured in distance and similarity measures or on which the respective measure can build on. There are at least three principles by which a similarity or distance measure can be developed: considering the elements of the paths, considering the order of the elements of the paths, or considering the position of the elements of the paths in the underlying graph. These principles might also be combined in order to get an appropriate similarity or distance measure. In any case, it needs to be emphasized that the meaning of the considered paths must not neglected in the choice of principle for the similarity or distance measure.

Based on the three main principles, we propose similarity and distance measures for paths and an reasonable possible normalization for each of the measures. Five distance and similarity measure are selected and analyzed for their properties: the node set similarity, the edge set similarity, the LCSS similarity, the simple average distance, and the matched average distance are each checked which of the proposed properties they satisfy, each measure in the normalized and the unnormalized variant. Table 2 shows an overview of the results of the analysis. For some combinations of measure and property, it is not shown yet whether the measure satisfies the properties, which is marked with a question

mark in the table. For all other combinations, we give the proofs that the measure satisfies the property or provide a counterexample in which the measure does not satisfy the respective property.

The second part of the present work applies the developed similarity and distance measures on paths from real-world data and presents a simple preliminary approach to cluster the available paths according to their computed similarity. For this reason, we processed the log data of students playing a board game and extracted the students' path through the problem space of the game which represent the students' solution for the game. In order to get an overview of the available path data, we present several different possibilities to visualize the paths and explore their properties. For each pair of paths of one game, we computed the similarity or distance value for the several proposed measures. Based on the similarity or distance of the available path, we used an hierarchical clustering approach to partition the paths into groups of similar paths. Since there is no ground truth available which paths should be in the same cluster or which paths are similar to each other – because this concept does not exist yet, but is to be developed in this work –, it is a difficult task to evaluate the results of the clustering algorithm. We approach this problem by developing the concept of cluster purity regarding several qualities and evaluate the similarity and distance measures by observing the (mean) cluster purity during the process of hierarchical clustering.

Regarding this evaluation measure, the similarity and distance measures show on this data a similar behavior – except for the matched average distance which seems to be more appropriate than the others to distinguish between solved and not solved paths. However, there is hardly a definite statement possible without a proper evaluation measure for the similarity measures. This and other issues which are still left open, are discussed in the next section.

5.2 FUTURE WORK

Since the present work is meant as a starting point for future research in the area of path similarity, there are several directions into which further work can lead and various ideas which can be pursued in the near and in the distant future. We will give a short collection of concrete and visionary ideas for future work:

- Section 3.4 in which the proposed similarity and distance measures are analyzed for their properties, is left with a few gaps: for some combinations of measure and property, it is not proven yet whether the measure satisfies the property. Table 2 provides the information for which combination the proof is missing. These combinations need to be checked in future work.
- Section 3.3 proposes more similarity and distance measures than were used in this work. There are also formulated several ideas for measures which are not elaborated further. It might be worth to spend time on developing and evaluating the remaining ideas for similarity and distance measures. For example, we proposed an edit distance for paths which allows the transformation operations of inserting, deleting and substituting nodes in the path. It should be revised whether these are the most appropriate edit operations on paths, or whether other operations reflect better the pro-

cess of path evaluation. Furthermore, it is an interesting question to which extent the structure of the underlying needs to be taken into account when transforming a path into another and which constraints should be posed on the intermediate stages of the transformation. Another example for a similarity measure which needs further development is the tuple similarity proposed in section 3.3.3, since it is not clear whether it captures the desired properties of paths.

- Besides the systematic analysis of the available measures for their properties, it might be useful to provide an extensive examination of the measures' computational complexity and appropriate approximation algorithms, if needed.
- In section 3.1, we proposed a set of properties a similarity or distance measure for paths might satisfy. However, neither of these properties considers changes of the underlying graph although it is in reality not an unusual phenomenon that nodes or edges are added or removed in the graph. Therefore, it is an interesting question to be considered in future work, which changes of the underlying graph should affect the similarity of two existing paths and which changes of the graph should leave the value of the similarity for the two paths unaffected. Certainly, the similarity measure value should change if the graph changes involves any nodes or paths of one of the paths. But in what extent should changes of the graph have an effect on the similarity of paths not directly involved? For example, adding or removing edges in the underlying might considerably change the length of the shortest path between nodes contained in the paths. It could make sense that such a changes has an influence on the similarity value of the paths. Therefore, future work can aim at answering the question when modifications of the graph should affect similarity measure values and when they should not affect the measure value, as well as whether such a general distinction is possible at all.
- In section 3.4, it is shown that neither of the proposed measures satisfies all proposed properties. It is an open question whether there exists a measure which can satisfy all properties, at all. If there is one (or several), it would be interesting to find this measure and analyze it for its further properties. In the other case, it would be an achievement to prove that there does not exist such a measure.
- In section 3.3.4, the idea of parametrized similarity and distance measures is proposed, i.e. an extension of the measures which allows to adjust the flexibility of the measures regarding certain features. Taking the node set similarity as an example, it might be reasonable to increase the flexibility when two nodes count as identical and contribute to the intersection of the paths' node sets. One could imagine that nodes of the two paths are considered as the same (in the sense that they contribute to the node set intersection) if their connection in the graph is shorter than the respective parameter. Also for other similarity and distance measures, it could be enhancing to introduce parameters.
- A rather technical than creative task which should be done in future research is to adapt the existing measures and properties to directed and

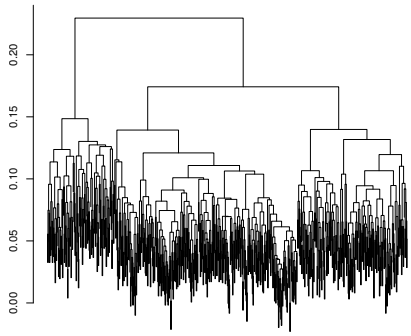
weighted graphs, since all measures and properties were restricted to simple, undirected and unweighted graphs. It needs to be considered how the extension to a more general graph class will change the meaning of paths, of path similarity, and how this can be transfered to the respective measures.

- Having a variety of different similarity and distance measures and an analysis of their properties at hand, it would be interesting to develop a kind of guideline in which settings which similarity or distance measure should be used: which assumptions are hidden in the measures, which prerequisites need to be fulfilled in order to apply a measure, which meaning does a path in the situation of interest have, and which measure is then appropriate? In the ideal case, we can develop a general scheme which can then be followed in order to find an appropriate measure.
- Section 4, we present a first approach of clustering paths by their similarity. There are a lot of approaches by which this work can be continued. Though the most important aspect which should be considered is the lacking ground truth for path clusters. Generating an appropriate ground truth should have a high priority in future work, in order to allow an adequate evaluation of the measures and the clustering results. It needs to be investigated whether a ground truth for each data set needs to be generated manually or whether there is a general approach to achieve this. Otherwise (or rather: additionally), further methods for a proper evaluation of the measures are needed.
- In order to develop evaluation methods or a ground truth for the paths, it might be helpful to have tools at hand which allow a meaningful visualization of the available paths. Section 4.2 proposes a few methods to explore the available path data, though, speaking of paths in *graphs*, a visualization of the paths in the graph could be useful. It will be certainly a challenge to handle a huge amount of paths embedded in graphs of considerable size, and to find a meaningful and useful visualization. Hopefully, a visualization might allow a first visual inspection of found path clusters and therefore a fast preliminary evaluation of clustering results.
- As soon as an appropriate evaluation of the measures and the clustering results is available, it will be exciting to extend the present approach on further data sets as well as by different clustering methods. In the present work, only one data set containing paths in a game's state space were used, however, paths occur in almost all application areas in which a graph representation is reasonable. Thus, validating the method on further data sets from different application areas (and in which paths have different meanings) will yield interesting results. Additionally, in the present work, only a simple hierarchical clustering approach was used. It might be a worthwhile approach to try other existing clustering and classification approaches, for example k-means-clustering or a support vector machine. Depending on the available ground truth or the respective path properties the algorithm is supposed to divide the paths into groups, it might be possible that another algorithm is more appropriate than an hierarchical clustering.

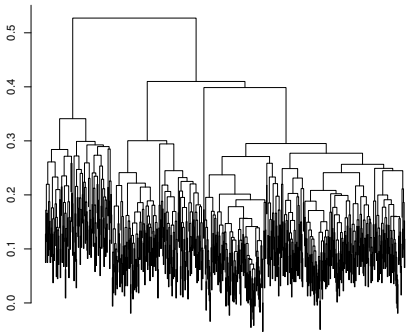
- For certain application scenarios, given a cluster of paths, it might be desirable to deduce a *representative* or *average path* from this group which then can be taken as a typical path for the group. It in not obvious at all how such an average path can be constructed from a given set of paths such that it has all properties characteristic for this group. It is possible that the constructed average path is a member of the given set, but also that is artificially constructed and not element of the given set.

A

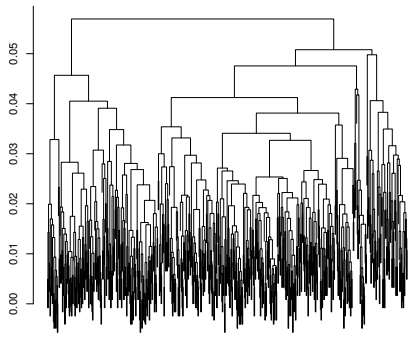
APPENDIX



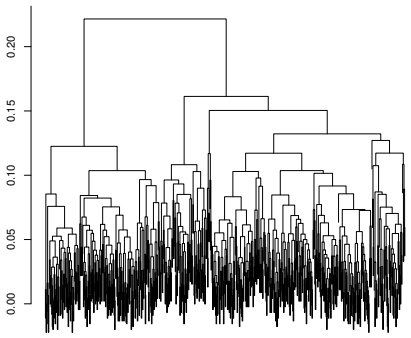
(a) Normalized simple average distance.



(b) Unnormalized simple average distance.

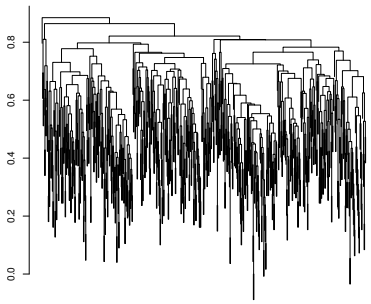


(c) Normalized matched average distance.

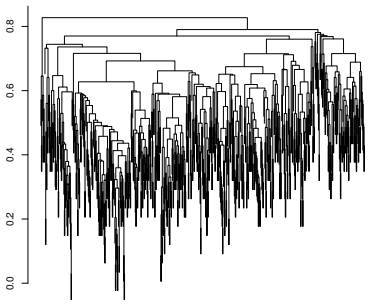


(d) Unnormalized matched average distance.

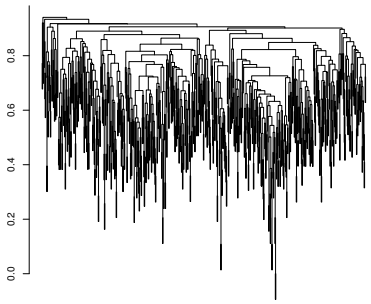
Figure 27: The dendrograms visualizing the hierarchical clustering results using average linkage for the game 779 and each of the path distance measures



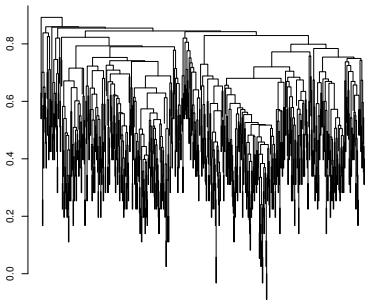
(a) Normalized node set similarity.



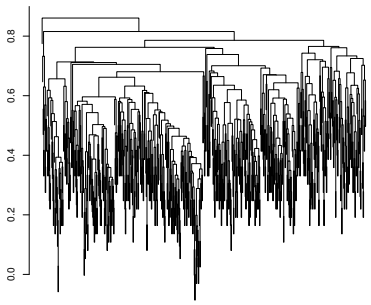
(b) Unnormalized node set similarity.



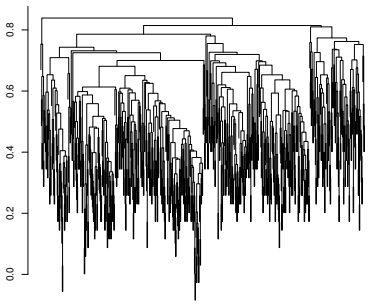
(c) Normalized edge set similarity.



(d) Unnormalized edge set similarity.

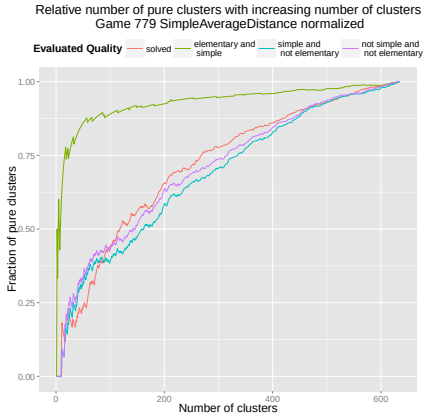


(e) Normalized LCSS similarity.

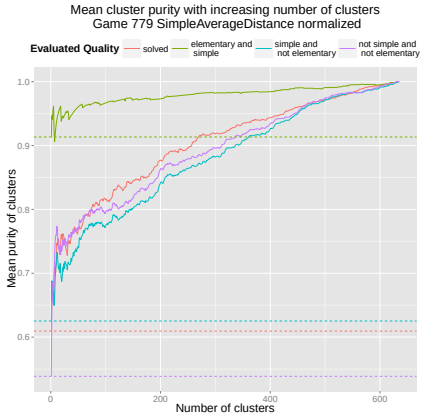


(f) Unnormalized LCSS similarity.

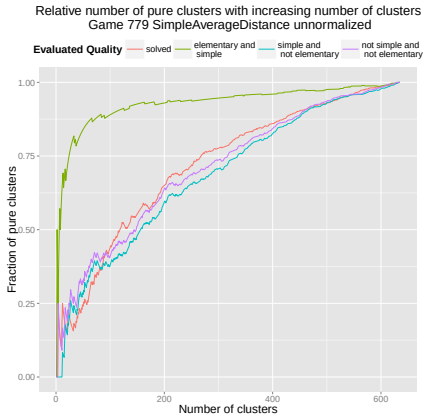
Figure 28: The dendrograms visualizing the hierarchical clustering results using average linkage for the game 779 and each of the path similarity measures



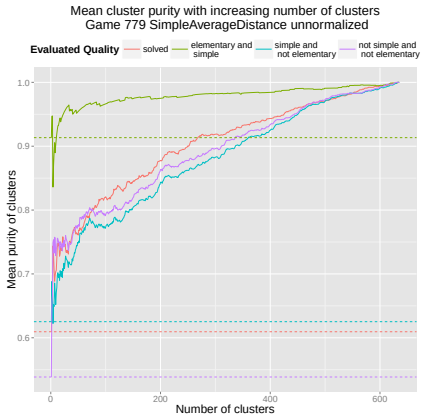
(a) Normalized simple average distance.



(b) Normalized simple average distance.

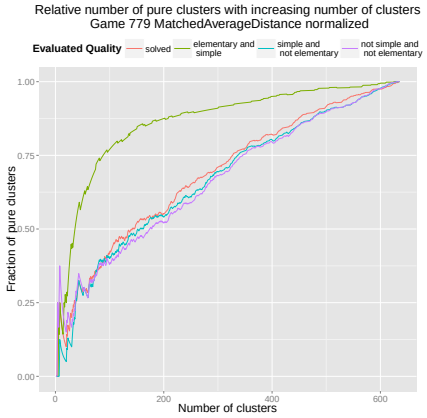


(c) Unnormalized simple average distance.

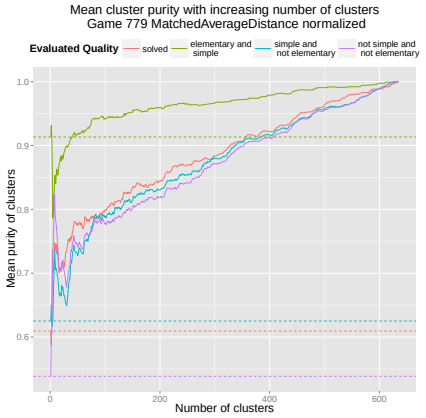


(d) Unnormalized simple average distance.

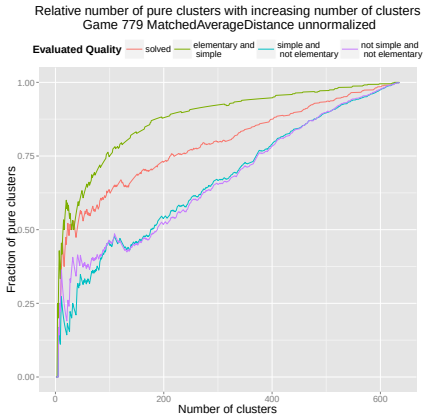
Figure 29: The number of pure clusters and the average purity of the clusters with increasing number of clusters for each of the proposed qualities and the simple average distance for the clusters computed with average linkage. The dashed lines indicate the fraction of the paths which satisfy the respective quality relative to the total number of paths. For each plot, the colors encode the evaluated quality: red – solved; green – elementary and simple; blue – simple and not elementary; purple – not simple and not elementary.



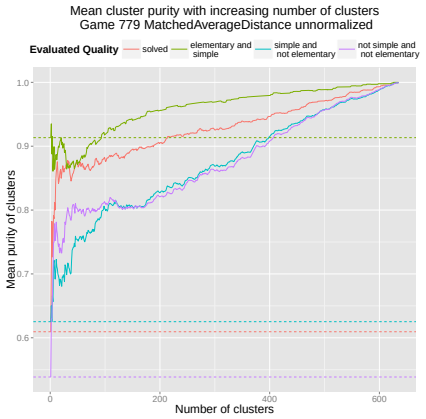
(a) Normalized matched average distance.



(b) Normalized matched average distance.

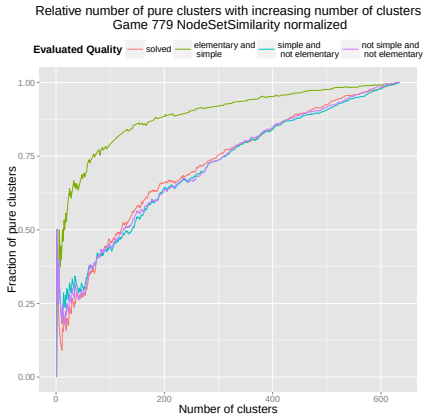


(c) Unnormalized matched average distance.

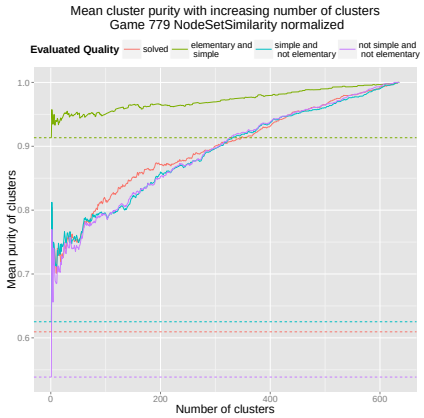


(d) Unnormalized matched average distance.

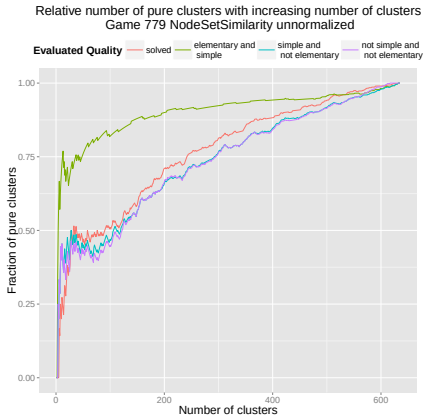
Figure 30: The number of pure clusters and the average purity of the clusters with increasing number of clusters for each of the proposed qualities and the matched average distance for the clusters computed with average linkage. The dashed lines indicate the fraction of the paths which satisfy the respective quality relative to the total number of paths. For each plot, the colors encode the evaluated quality: red – solved; green – elementary and simple; blue – simple and not elementary; purple – not simple and not elementary.



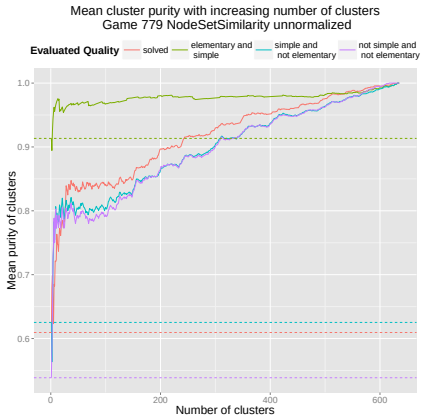
(a) Normalized node set similarity.



(b) Normalized node set similarity.

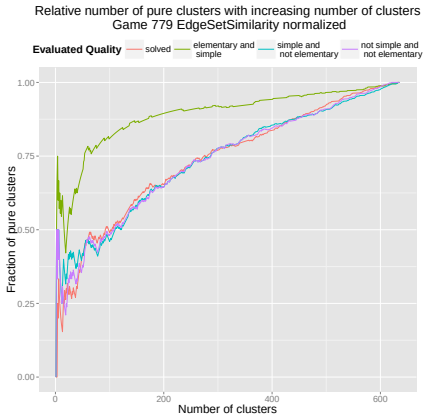


(c) Unnormalized node set similarity.

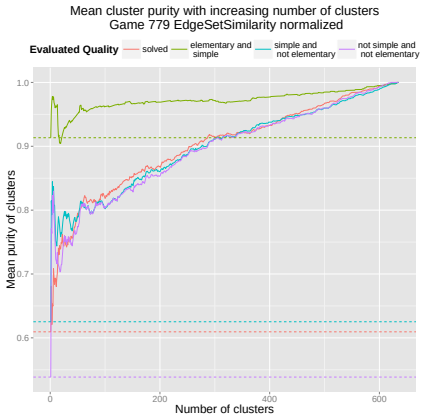


(d) Unnormalized node set similarity.

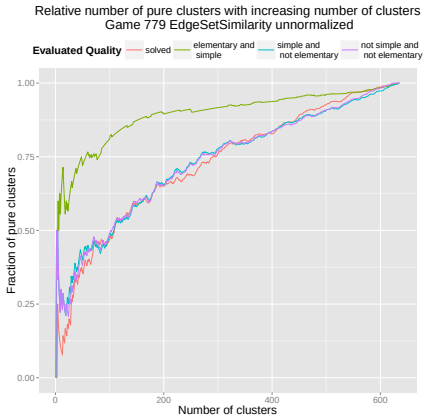
Figure 31: The number of pure clusters and the average purity of the clusters with increasing number of clusters for each of the proposed qualities and the node set similarity for the clusters computed with average linkage. The dashed lines indicate the fraction of the paths which satisfy the respective quality relative to the total number of paths. For each plot, the colors encode the evaluated quality: red – solved; green – elementary and simple; blue – simple and not elementary; purple – not simple and not elementary.



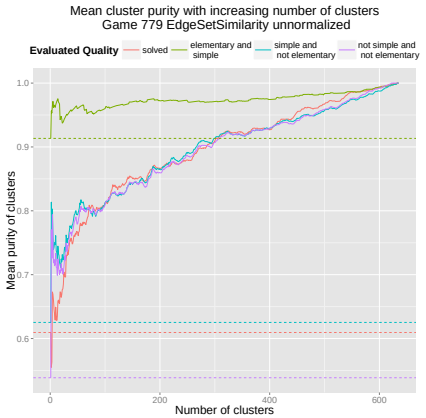
(a) Normalized edge set similarity.



(b) Normalized edge set similarity.

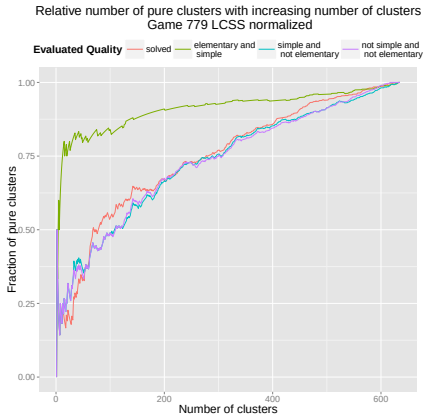


(c) Unnormalized edge set similarity.

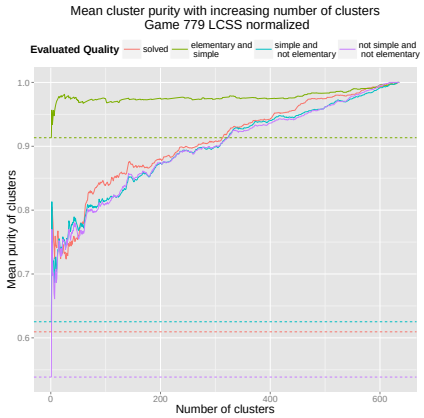


(d) Unnormalized edge set similarity.

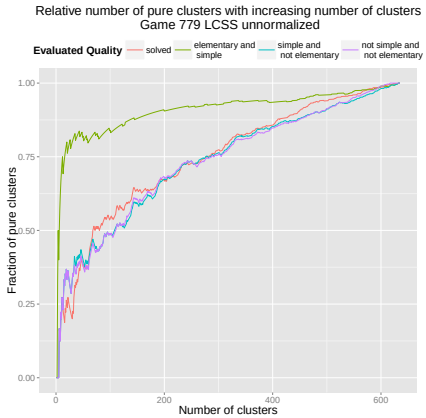
Figure 32: The number of pure clusters and the average purity of the clusters with increasing number of clusters for each of the proposed qualities and the edge set similarity for the clusters computed with average linkage. The dashed lines indicate the fraction of the paths which satisfy the respective quality relative to the total number of paths. For each plot, the colors encode the evaluated quality: red – solved; green – elementary and simple; blue – simple and not elementary; purple – not simple and not elementary.



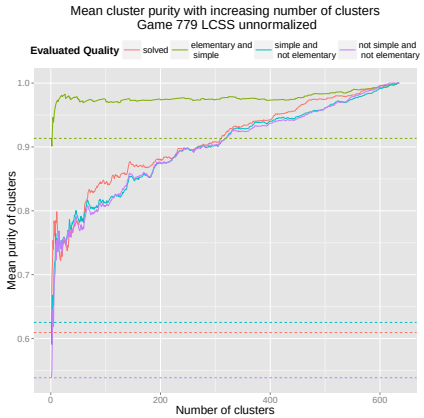
(a) Normalized LCSS similarity.



(b) Normalized LCSS similarity.



(c) Unnormalized LCSS similarity.



(d) Unnormalized LCSS similarity.

Figure 33: The number of pure clusters and the average purity of the clusters with increasing number of clusters for each of the proposed qualities and the LCSS similarity for the clusters computed with average linkage. The dashed lines indicate the fraction of the paths which satisfy the respective quality relative to the total number of paths. For each plot, the colors encode the evaluated quality: red – solved; green – elementary and simple; blue – simple and not elementary; purple – not simple and not elementary.

BIBLIOGRAPHY

[1] Akcora, C. G., and Ferrari, E. Similarity metrics on social networks. In *Encyclopedia of Social Network Analysis and Mining*. Springer, New York, NY, USA, 2014, pp. 1734–1743.

[2] Anderberg, M. R. *Cluster Analysis for Applications: Probability and Mathematical Statistics: A Series of Monographs and Textbooks*, vol. 19. Academic press, 2014.

[3] Bashir, F., Khokhar, A., and Schonfeld, D. Segmented trajectory based indexing and retrieval of video data. In *Proceedings of the International Conference on Image Processing* (2003), vol. 2, IEEE, pp. II–623.

[4] Blondel, V. D., Gajardo, A., Heymans, M., Senellart, P., and Van Dooren, P. A measure of similarity between graph vertices: Applications to synonym extraction and web searching. *SIAM review 46*, 4 (2004), 647–666.

[5] Buzan, D., Sclaroff, S., and Kollios, G. Extraction and clustering of motion trajectories in video. In *Proceedings of the 17th International Conference on Pattern Recognition* (2004), vol. 2, IEEE, pp. 521–524.

[6] Cormen, T. H., Leiserson, C. E., Rivest, R. L., and Stein, C. *Introduction to Algorithms*, 3rd ed. The MIT Press, Cambridge, Massachusetts, 2009.

[7] Das, G., Gunopulos, D., and Mannila, H. Finding similar time series. In *Principles of Data Mining and Knowledge Discovery*, J. Komorowski and J. Zytkow, Eds., vol. 1263 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 1997, pp. 88–100.

[8] Dempster, A. P., Laird, N. M., and Rubin, D. B. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society. Series B (Methodological) 39*, 1 (1977), 1–38.

[9] Ester, M., Kriegel, H.-P., Sander, J., and Xu, X. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of the 2nd International Conference on Knowledge Discovery and Data Mining* (1996), vol. 96, pp. 226–231.

[10] Fu, Z., Hu, W., and Tan, T. Similarity based vehicle trajectory clustering and anomaly detection. In *Proceedings of the IEEE International Conference on Image Processing* (2005), vol. 2, IEEE, pp. II–602.

[11] Gower, J. C., et al. Measures of similarity, dissimilarity and distance. *Encyclopedia of Statistical Sciences 5*, 3 (1985), 397–405.

[12] Gündüz, Ş., and Özsu, M. T. A web page prediction model based on click-stream tree representation of user behavior. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining* (2003), ACM, pp. 535–540.

[13] Gusfield, D. *Algorithms on strings, trees and sequences: computer science and computational biology*. Cambridge University Press, New York, NY, USA, 1997.

[14] Hartigan, J. A. *Clustering algorithms*. John Wiley and Sons, New York, 1975.

[15] Jaccard, P. Etude comparative de la distribution florale dans une portion des alpes et du jura. *Bulletin del la Société Vaudoise des Sciences Naturelles 37* (1901), 547–579.

[16] Jain, A. K., and Dubes, R. C. *Algorithms for Clustering Data*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1988.

[17] Jarušek, P. *Modeling problem solving times in tutoring systems*. PhD thesis, Masarykova univerzita, Fakulta informatiky, 2013.

[18] Jarušek, P., and Pelánek, R. Analysis of a simple model of problem solving times. In *Intelligent Tutoring Systems*, S. Cerri, W. Clancey, G. Papadourakis, and K. Panourgia, Eds., vol. 7315 of *Lecture Notes in Computer Science*. Springer, Berlin Heidelberg, 2012, pp. 379–388.

[19] Jeh, G., and Widom, J. SimRank: A measure of structural-context similarity. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining* (2002), ACM, pp. 538–543.

[20] Junejo, I. N., Javed, O., and Shah, M. Multi feature path modeling for video surveillance. In *Proceedings of the 17th International Conference on Pattern Recognition* (2004), vol. 2, IEEE, pp. 716–719.

[21] Kaufman, L., and Rousseeuw, P. J. *Finding groups in data: an introduction to cluster analysis*, vol. 344. John Wiley and Sons, New York, 2009.

[22] Keogh, E. J., and Pazzani, M. J. Scaling up dynamic time warping for datamining applications. In *Proceedings of the sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (2000), ACM, pp. 285–289.

[23] Kleinberg, J. An impossibility theorem for clustering. *Advances in neural information processing systems* (2003), 463–470.

[24] Kriegel, H.-P., Kröger, P., Sander, J., and Zimek, A. Density-based clustering. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery 1*, 3 (2011), 231–240.

[25] Krumke, S. O., and Noltemeier, H. *Graphentheoretische Konzepte und Algorithmen*, 3rd ed. Vieweg+Teubner Verlag, Wiesbaden, 2012.

[26] Kumar, P. *An investigation of classification and clustering of sequential data.* PhD thesis, University of Hyderabad, 2006.

[27] Laasonen, K. Clustering and prediction of mobile user routes from cellular data. In *Knowledge Discovery in Databases: PKDD 2005*, vol. 3721 of *Lecture Notes in Computer Science*. Springer, Berlin Heidelberg, 2005, pp. 569–576.

[28] Laasonen, K. Route prediction from cellular data. In *Workshop on Context-Awareness for Proactive Systems (CAPS)* (2005), vol. 1617.

[29] Latecki, L. J., and Lakämper, R. Shape similarity measure based on correspondence of visual parts. *IEEE Transactions on Pattern Analysis and Machine Intelligence 22*, 10 (2000), 1185–1190.

[30] Lee, J.-M., and Hwang, B.-Y. Path bitmap indexing for retrieval of XML documents. In *Modeling Decisions for Artificial Intelligence*, vol. 3885 of *Lecture Notes in Computer Science*. Springer, Berlin Heidelberg, 2006, pp. 329–339.

[31] Leicht, E. A., Holme, P., and Newman, M. E. J. Vertex similarity in networks. *Physical Review E 73*, 2 (2006), 026120.

[32] Lin, D. An information-theoretic definition of similarity. In *Proceedings of the Fifteenth International Conference on Machine Learning* (San Francisco, CA, USA, 1998), vol. 98, Morgan Kaufmann Publishers Inc., pp. 296–304.

[33] Lloyd, S. P. Least squares quantization in PCM. *IEEE Transactions on Information Theory 28*, 2 (1982), 129–137.

[34] Lou, J., Liu, Q., Tan, T., and Hu, W. Semantic interpretation of object activities in a surveillance system. In *Proceedings of the 16th International Conference on Pattern Recognition* (2002), vol. 3, IEEE, pp. 777–780.

[35] Makris, D., and Ellis, T. Path detection in video surveillance. *Image and Vision Computing 20*, 12 (2002), 895–903.

[36] Mannila, H., and Moen, P. Similarity between event types in sequences. In *Proceedings of the First International Conference on Data Warehousing and Knowledge Discovery*. Springer, London, 1999, pp. 271–280.

[37] Mannila, H., and Ronkainen, P. Similarity of event sequences. In *Proceedings of the 4th International Workshop on Temporal Representation and Reasoning (TIME)* (1997), IEEE Computer Society, p. 136.

[38] Moen, P. *Attribute, event sequence, and event type similarity notions for data mining.* PhD thesis, University of Helsinki, Department of Computer Science, 2000.

[39] Mor, E., and Minguillón, J. E-learning personalization based on itineraries and long-term navigational behavior. In *Proceedings of the 13th international World Wide Web conference on Alternate track papers & posters* (2004), ACM, pp. 264–265.

[40] Romero, C., and Ventura, S. Educational data mining: a review of the state of the art. *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews 40*, 6 (2010), 601–618.

[41] Salton, G., and Lesk, M. E. Computer evaluation of indexing and text processing. *Journal of the ACM 15*, 1 (1968), 8–36.

[42] Steinhaus, H. Sur la division des corps matériels en parties. *Bull. Acad. Pol. Sci., Cl. III 4* (1957), 801–804.

[43] Tan, P.-N., Steinbach, M., and Kumar, V. *Introduction to Data Mining, (First Edition).* Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2005.

[44] Vlachos, M., Kollios, G., and Gunopulos, D. Discovering similar multidimensional trajectories. In *Proceedings of the 18th International Conference on Data Engineering* (2002), IEEE, pp. 673–684.

[45] Wang, W., and Zaïane, O. R. Clustering web sessions by sequence alignment. In *Proceedings of the 13th International Workshop on Database and Expert Systems Applications* (2002), IEEE, pp. 394–398.

[46] Xing, E. P., Jordan, M. I., Russell, S., and Ng, A. Y. Distance metric learning with application to clustering with side-information. In *Advances in neural information processing systems* (2003), MIT Press, pp. 505–512.

[47] Yang, J., and Wang, W. CLUSEQ: efficient and effective sequence clustering. In *Proceedings of the 19th International Conference on Data Engineering* (2003), IEEE, pp. 101–112.

[48] Zhang, Z., Huang, K., and Tan, T. Comparison of similarity measures for trajectory clustering in outdoor surveillance scenes. In *Proceedings of the 18th International Conference on Pattern Recognition* (2006), vol. 3, IEEE, pp. 1135–1138.

LIST OF FIGURES

LIST OF TABLES

ERKLÄRUNG

Ich versichere hiermit, dass ich die vorliegende Masterarbeit mit dem Thema *Measures for the Similarity of Paths in Complex Networks* selbstständig verfasst und keine anderen als die angegebenen Hilfsmittel benutzt habe. Die Stellen, die anderen Werken dem Wortlaut oder dem Sinn nach entnommen wurden, habe ich durch die Angabe der Quelle, auch der benutzten Sekundärliteratur, als Entlehnung kenntlich gemacht

Kaiserslautern, 12. Oktober 2015

Mareike Bockholt