Lecture 2: Graph Similarity

Part 2.1: Introduction, Structural Approaches

Professor Dr. Petra Mutzel

Computational Analytics

Computer Science

University of Bonn

UNIVERSITÄT BONN

RHEINISCHE FRIEDRICH-WILHELMS-UNIVERSITÄT BONN

INSTITUT FÜR INFORMATIK DER UNIVERSITÄT BONN

Hausdorff School: Computational Combinatorial Optimization, September 12-16, 2022

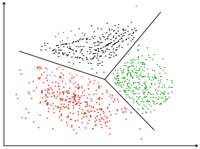
Literature: Subgraph-Isomorphism Approaches

- N. Kriege, A. Droschinsky, P. Mutzel: A note on block-and-bridge preserving maximum common subgraph algorithms for outerplanar graphs, J. Graph Algorithms Appl. 22(4), 2018
- L. Humbeck, S. Weigang, T. Schäfer, P. Mutzel, O. Koch: CHIPMUNK: A virtual synthesizable small molecule library for medicinal chemistry exploitable for protein-protein interaction modulators, MFCS 2018, ChemMedChem 6/2018, vol. 13 (6), Very Important Paper, 2018
- A. Droschinsky, N. Kriege, P. Mutzel: Largest Weight Common Subtree Embeddings with Distance Penalties, MFCS 2018
- N. Kriege, F. Kurpicz, P. Mutzel: On Maximum Common Subgraph Problems in Series-Parallel Graphs, European Journal of Combinatorics, vol. 68, 2018
- A. Droschinsky, N. Kriege, P. Mutzel: Finding Largest Common Substructures of Molecules in Quadratic Time, SOFSEM-FOCS 1017, LNCS 10139, 2017

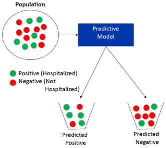
Literature: Graph Edit Distance, Frobenius Distance

- Lei Yang and Lei Zou: Noah: Neural-optimized A^* Search Algorithm for Graph Edit Distance Computation, IEEE 37th Int. Conf. on Data Engineering (ICDE) 2021, 576-587
- Xiaoyang Chen, Hongwei Huo, Jun Huan, Jeffrey Scott Vitter: An efficient algorithm for graph edit distance computation, Knowledge-Based Systems, vol. 163, 2019
- David B. Blumenthal, Johann Gamper: On the exact computation of the graph edit distance, Pattern Recognition Letters 134, 46-57, 2020 [A^* and ILP]
- Julien Lerouge, Zeina Abu-Aisheh, Romain Raveaux, Pierre Héroux, and Sébastien Adam: New binary linear programming formulation to compute the graph edit distance, Pattern Recognition 72, 254-265, 2017 [ILP]
- Martin Grohe, Gaurav Rattan, Gerhard J. Woeginger: Graph Similarity and Approximate Isomorphism, 43rd International Symposium on Mathematical Foundations of Computer Science (MFCS) 2018, LIPIcs 117, 20:1-20:16 [NP-completeness]

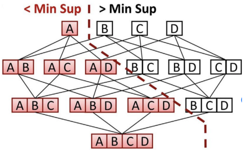
Motivation: Four fundamental problems for analyzing data (Aggarwal 2015)



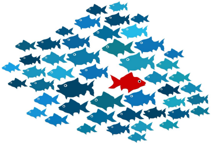
Clustering



Classification



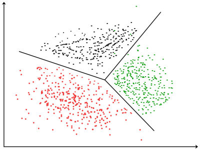
Association Pattern Mining

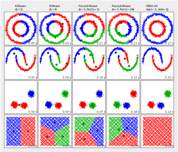


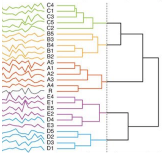
Outlier Detection

Sources: www.geeksforgeeks.org/, towardsdatascience.com/, www.researchgate.net/

Clustering (Unsupervised Learning)





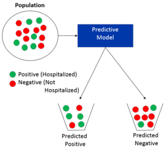


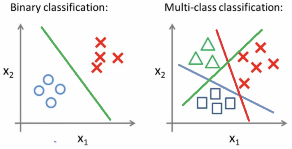
Clustering is the task of grouping a set of objects such that objects in the same group (cluster) are more similar to each other than objects in different groups.

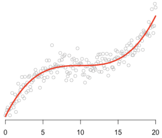
- many possibilities for formal definition
- studied for a long time (statistics, data bases, machine learning)
- recently also in TCS (e.g., data streams)
- useful for data sparsification and sampling approaches → big data
- popular approaches: distance-based, centroid-based (k-Means)

Sources: www.geeksforgeeks.org/, www.hipparchus.org/, www.nature.com/

Classification (Supervised Learning)





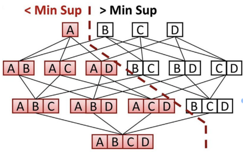


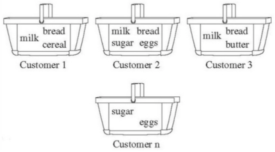
Given: Training set with labelled items (classes).
Goal: Train a classifier so that a new item will be assigned to its correct class.

- many variations (decisions, predictions)
- discrete labels: mostly studied in ML community
- continuous labels: regression mostly studied in statistics
- popular approaches: SVMs (kernel), deep learning, k-NN, logistic regression, Bayes classifiers, decision tree method, clustering

Sources: towardsdatascience.com/, www.ritchieng.com/logistic-regression/, medium.freecodecamp.org/

Association Pattern Mining





ID	Rule	Support	Confidence
r1	$\{a, b, c\} \Rightarrow \{e\}$	0.5	1.0
r2	$\{a\} \rightarrow \{c, e, f\}$	0.5	0.66
r3	$\{a, b\} \rightarrow \{e, f\}$	0.5	1.0
r4	$\{b\} \rightarrow \{e, f\}$	0.75	0.75
r5	$\{a\} \rightarrow \{e, f\}$	0.75	1.0
r6	$\{c\} \rightarrow \{f\}$	0.5	1.0
r7	$\{a\} \rightarrow \{b\}$	0.5	0.66
...	...	...	...

Goal: find inherent regularities in the data

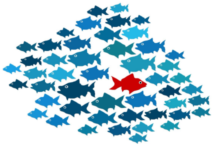
Frequent pattern mining: find frequent items (occuring at least minimum support times)

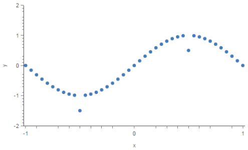
Association pattern mining: generalization also based on association rules

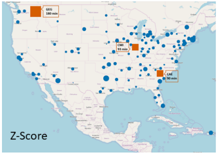
- widely studied in data mining
- useful for deriving similarity measures on data sets
- useful for clustering, classification, outlier detection

Sources: medium.freecodecamp.org/, www.kdnuggets.com/

Outlier Detection





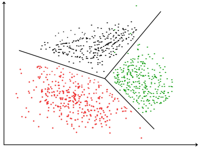


Find data items that are different from the majority of the given data.

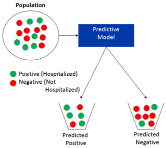
- outlier may reflect errors in the data or belong to rare events
- methods: statistical tests, models based on spatial proximity (k-NN), density-based methods, ...
- methods: clustering, classification, association pattern mining

Sources: towardsdatascience.com/, www.kdnuggets.com/

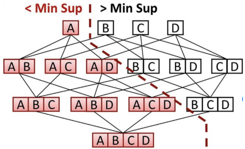
Four fundamental problems for analyzing data (Aggarwal 2015)



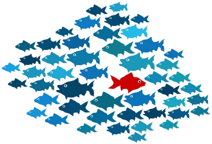
Clustering



Classification



Association Pattern Mining

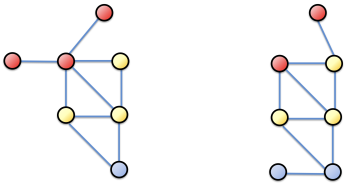


Outlier Detection

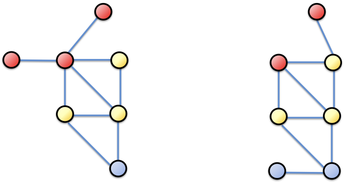
Important basis for all these: Distances and Similarities

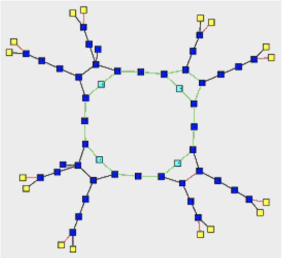
Sources: www.geeksforgeeks.org/, towardsdatascience.com/, www.researchgate.net/

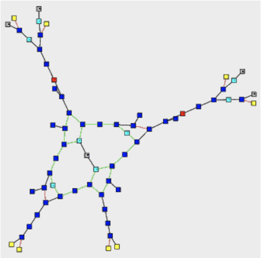
When are two graphs similar?



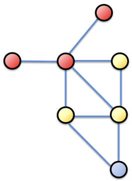
When are two graphs similar?

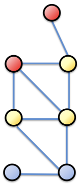


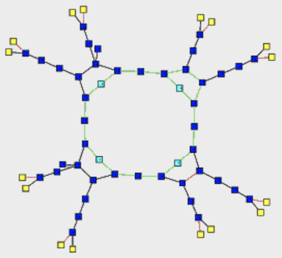


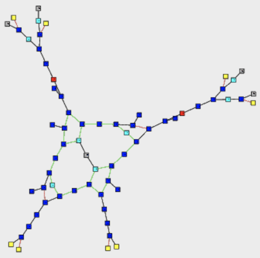


When are two graphs similar?









SURVEY

Approaches to Graph Similarity

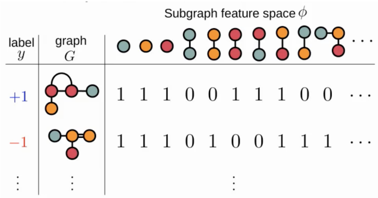
Structural

- isomorphism-based
- Frobenius distance
- graph edit distance

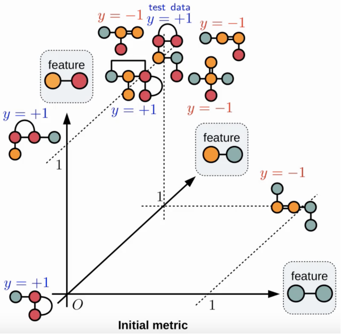
Graph embedding

- Weisfeiler-Leman
- fingerprint

Graph Embedding



feature vector



feature space

- Generate a feature vector for each graph in the given graph data set
- Use your favorite (data) classification/clustering method

Sources: https://www.kdd.org/kdd2019/accepted-papers/view/learning-interpretable-metric-between-graphs-convex-formulation-and-computa

Outline for Structural Similarity Approaches

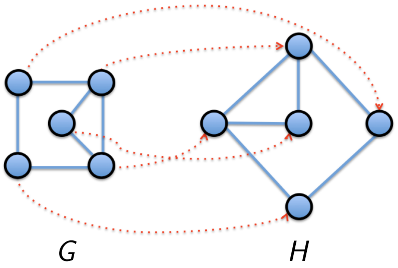
1. **Introduction**
2. **Isomorphism-based Approaches**
   - Graph isomorphism
   - Subgraph Isomorphism based Approaches
   - Structural Clustering of Sets of Graphs
3. **Frobenius Distance**
4. **Graph Edit Distance**
   - A^* Algorithm for GED
   - Integer Linear Programs for GED
   - Computational Study

When are two graphs identical?

Graph Isomorphism

Let $G = (V_G, E_G)$ and $H = (V_H, E_H)$ be simple graphs. A bijective mapping $\pi : V_G \rightarrow V_H$ is called graph isomorphism if the following holds:

$$\forall\, v, w \in V_G : (v, w) \in E_G \iff (\pi(v), \pi(w)) \in E_H$$

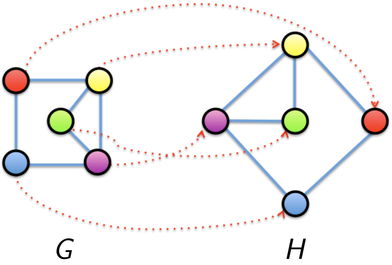


Graph Isomorphism

Graph Isomorphism

Let $G = (V_G, E_G)$ and $H = (V_H, E_H)$ be simple graphs. A bijective mapping $\pi : V_G \to V_H$ is called graph isomorphism if the following holds:

$$\forall\, v, w \in V_G : (v, w) \in E_G \iff (\pi(v), \pi(w)) \in E_H$$

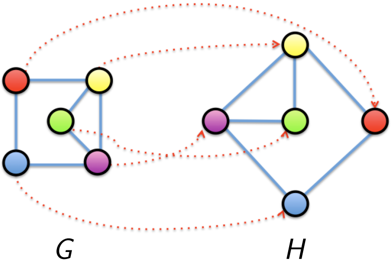


Graph Isomorphism

Graph Isomorphism

Let $G = (V_G, E_G)$ and $H = (V_H, E_H)$ be simple graphs. A bijective mapping $\pi : V_G \to V_H$ is called graph isomorphism if the following holds:

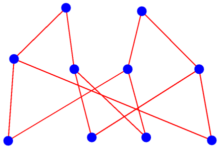
$$\forall\, v, w \in V_G : (v, w) \in E_G \iff (\pi(v), \pi(w)) \in E_H$$

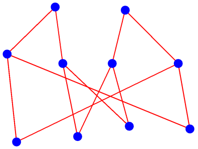


$\Rightarrow$ Natural extension to graphs with labels and attributes

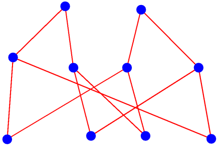
Two graphs are called isomorphic ($G_1 \simeq G_2$), if a graph isomorphism exists.

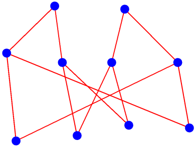
Are these graphs isomorphic to each other?

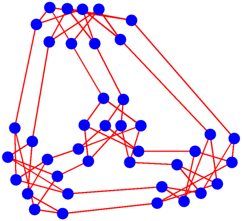


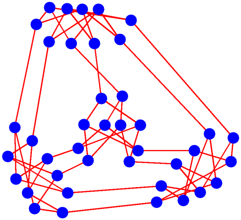


Are these graphs isomorphic to each other?

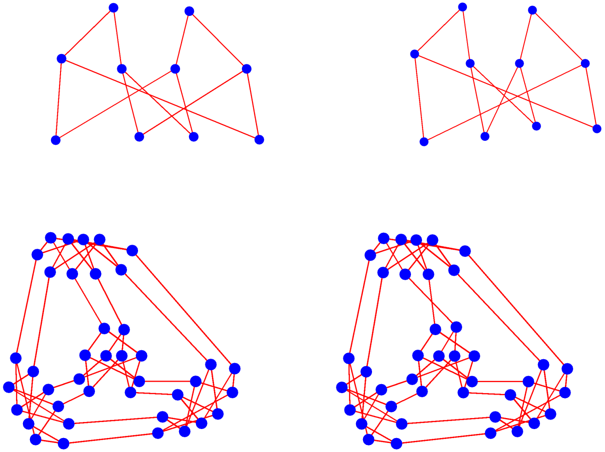








Are these graphs isomorphic to each other?



SURVEY

Graph Isomorphism Problem

Definition (Graph Isomorphism Problem)

Given: simple graphs G and H
Find: Is G isomorphic to H, i.e., $G \simeq H$?

Graph Isomorphism Problem

Definition (Graph Isomorphism Problem)

Given: simple graphs G and H
Find: Is G isomorphic to H, i.e., $G \simeq H$?

Remarks:

- If graphs have vertex and/or edge labels or attributes, definition will be extended to the labels (various possibilities)

Graph Isomorphism Problem

Definition (Graph Isomorphism Problem)

Given: simple graphs G and H
Find: Is G isomorphic to H, i.e., $G \simeq H$?

Remarks:

- If graphs have vertex and/or edge labels or attributes, definition will be extended to the labels (various possibilities)
- Complexity: OPEN

Graph Isomorphism Problem

Definition (Graph Isomorphism Problem)

Given: simple graphs G and H
Find: Is G isomorphic to H, i.e., $G \simeq H$?

Remarks:

- If graphs have vertex and/or edge labels or attributes, definition will be extended to the labels (various possibilities)
- Complexity: OPEN
- Quasipolynomial algorithm: $n^{(\log n)^{O(1)}}$ [Babai 2015/17]

Graph Isomorphism Problem

Definition (Graph Isomorphism Problem)

Given: simple graphs G and H
Find: Is G isomorphic to H, i.e., $G \simeq H$?

Remarks:

- If graphs have vertex and/or edge labels or attributes, definition will be extended to the labels (various possibilities)
- Complexity: OPEN
- Quasipolynomial algorithm: $n^{(\log n)^{O(1)}}$ [Babai 2015/17]
- Practice: mostly solvable very fast

Graph Isomorphism Problem

Definition (Graph Isomorphism Problem)

Given: simple graphs G and H
Find: Is G isomorphic to H, i.e., $G \simeq H$?

Remarks:

- If graphs have vertex and/or edge labels or attributes, definition will be extended to the labels (various possibilities)
- Complexity: OPEN
- Quasipolynomial algorithm: $n^{(\log n)^{O(1)}}$ [Babai 2015/17]
- Practice: mostly solvable very fast

→ Weisfeiler-Leman algorithm (Lecture on Friday)

Graph Isomorphism Problem

Definition (Graph Isomorphism Problem)

Given: simple graphs G and H
Find: Is G isomorphic to H, i.e., $G \simeq H$?

Remarks:

- If graphs have vertex and/or edge labels or attributes, definition will be extended to the labels (various possibilities)
- Complexity: OPEN
- Quasipolynomial algorithm: $n^{(\log n)^{O(1)}}$ [Babai 2015/17]
- Practice: mostly solvable very fast

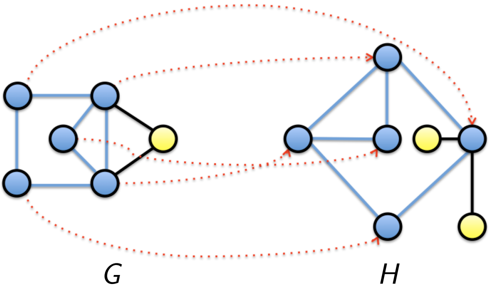
$\rightarrow$ Weisfeiler-Leman algorithm (Lecture on Friday)

$\rightarrow$ We want: graph similarity

Maximum Common Subgraph Problem (MCS)

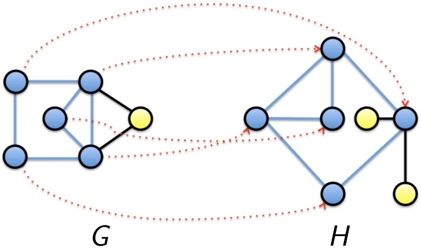
Definition (MCS)

Given: $G = (V_G, E_G)$ und $H = (V_H, E_H)$
Find: Largest vertex sets $R \subseteq V_G$ and $S \subseteq V_H$, such that the induced subgraphs $G[R]$ and $H[S]$ are isomorphic to each other.



Complexity: Decision problem is NP-complete

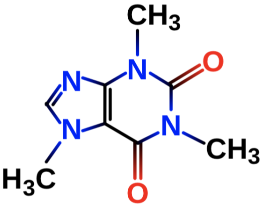
Maximum Common Subgraph Problem (MCS)

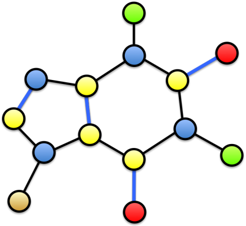


- many variants: vertex or edge induced subgraphs
- in practice: consider node/edge labels and attributes
- complexity: decision problem is NP-complete
- widely studied in Cheminformatics

Motivation: Rational Drug Design

- Which molecules are active against disease X?
- Which molecules have a similar function/effect? (Reduction of side effects)
- Which molecules may have an increased effectiveness?
- High-throughput screening for promising candidates

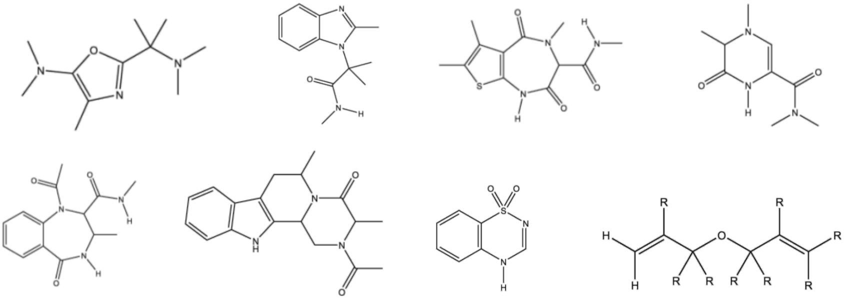




- Molecules can be modelled as graphs with attributes
- Direct relationship between structure and effects

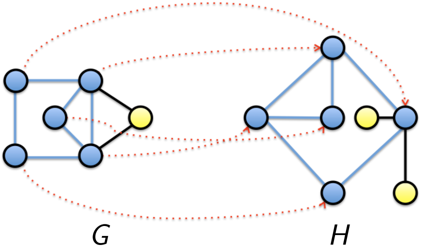
$\rightarrow$ Graph similarity

Properties of Molecule Graphs



- almost always planar, often outerplanar
- bounded tree width
- bounded degree
- have vertex and edge labels (e.g. activity attributes)

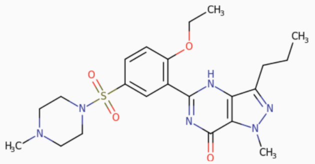
Maximum Common Subgraph Problem (MCS)



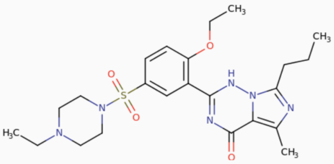
- polynomial time algorithms for trees and outerplanar G
- NP-complete for partial k-trees for $k = 11$ with bounded node degree

Kriege, Kurpicz, Mutzel 2018

Analysis of the Chemical Problem



(a) Sildenafil

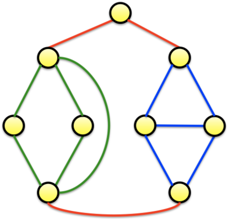


(b) Vardenafil

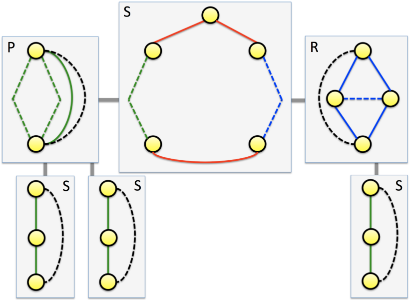
- Rings and bridges shall be preserved
- very important in Cheminformatics

$\rightarrow$ Block-and-bridge preserving MCS $\rightarrow$ simpler

Block/Bridge MCS



G

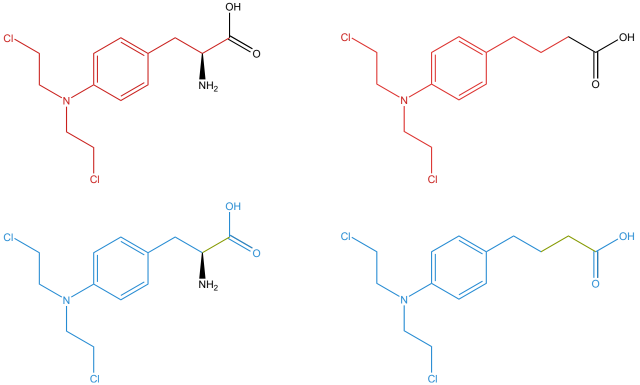


SPQR-tree of G

- polynomial algorithm for block/bridge MCS for partial 2-trees
- generalizes results for trees and outerplanar G
- Idea: BC-tree and SPQR-tree decomposition
- NP-completeness of MCS for biconnected partial 2-trees with almost all vertices of degree bounded by 3

Kriege, Kurpicz, Mutzel 2018, Gutwenger, Mutzel 2000, Kriege, Mutzel 2014

Maximum Common Subgraph Embedding

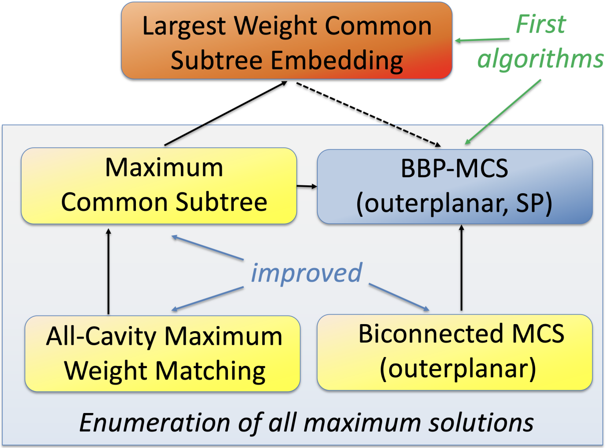


Upper: MCS, Lower: MCS Embedding of Melphalan in Chlorambucil

- bridges (single edges) maybe mapped to paths of bridges
- → Largest Weight Common Subtree Embeddings

Droschinsky, Kriege, Mutzel 2018

Overview of Algorithmic Results



Droschinsky Dissertation 2021, Droschinsky, Kriege, Mutzel 2016, 2017, 2018, Kriege, Kurpicz, Mutzel 2017, 2018

Clustering of Graphs with Subgraph-Isomorphism

Clustering of graphs

Given: set of graphs $\mathcal{X} = \{G_1, \ldots, G_n\}$

Goal: find a clustering of $\mathcal{X}$ that

- maximizes cluster homogenity
- separates cluster from each other

Clustering of Graphs with Subgraph-Isomorphism

Clustering of graphs

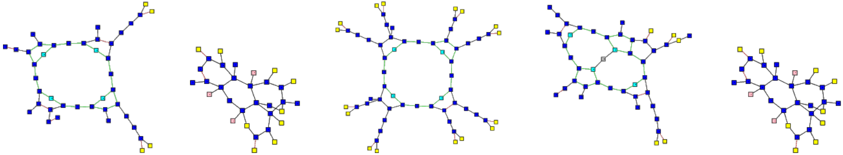
Given: set of graphs $\mathcal{X} = \{G_1, \ldots, G_n\}$

Goal: find a clustering of $\mathcal{X}$ that

- maximizes cluster homogenity
- separates cluster from each other

Setting in drug design

- small graphs
- huge number of graphs ($\gg 10^6$)



Clustering of Graphs with Subgraph-Isomorphism

Clustering of graphs

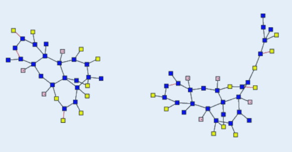
Given: set of graphs $\mathcal{X} = \{G_1, \ldots, G_n\}$

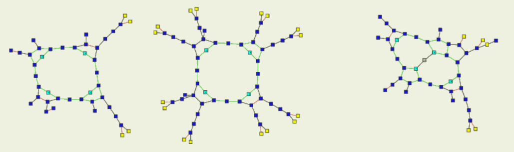
Goal: find a clustering of $\mathcal{X}$ that

- maximizes cluster homogenity
- separates cluster from each other

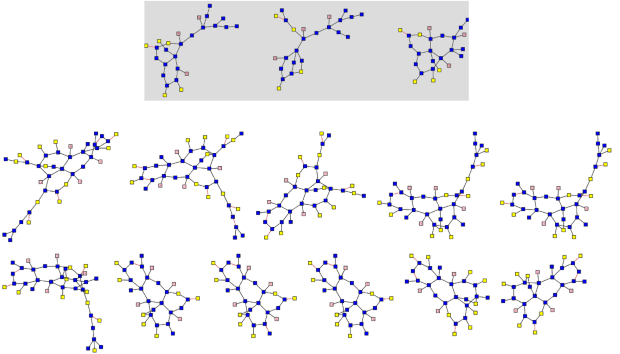
Setting in drug design

- small graphs
- huge number of graphs ($\gg 10^6$)





Structural Clustering (StruClus)

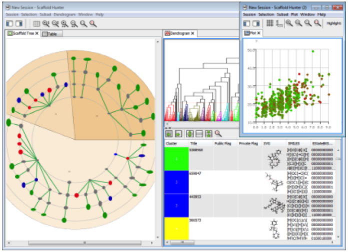


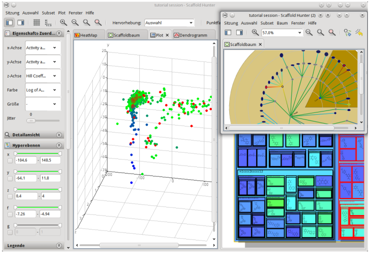
- sets of representatives for clusters → interpretability
- similarity measure based on common subgraphs
- new error-bounded sampling strategy for *support counting*
- linear running time → scalable
- parallelisable → very fast in practice

Mutzel, Schäfer, 2017, 2022

Rational Drug Design: GraBaDrug

DFG SPP Algorithms for Big Data

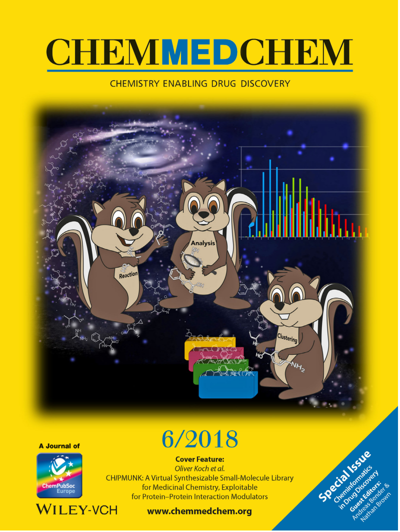




- explorative analysis of molecule data bases ← Scaffold Hunter
- similarity search on molecular structures ← graph similarity, clustering
- creation of virtual molecule data bases for drug design
  ← CHIPMUNK: 95 mio. molecules with $\leq$ 700 atoms, 90 attributes

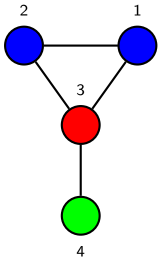
Nature Chem. Biol. 2009: Wetzel, Klein, Renner, Rauh, Oprea, Mutzel, Waldmann
ChemMedChem 2018: Humbeck, Weigang, Schäfer, Mutzel Koch + Cover Feature, ...

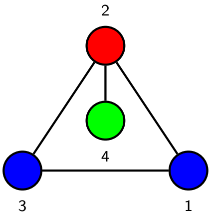
CHIPMUNK: A Virtual Synthesizable Small-Molecule Library for Medicinal Chemistry



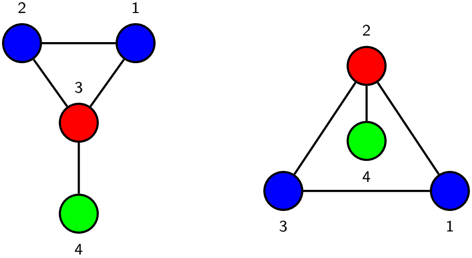
Humbeck, Weigang, Schäfer, Mutzel, Koch ChemMedChem 2018 + Cover Feature

Motivation: Permutation of Adjacency Matrices



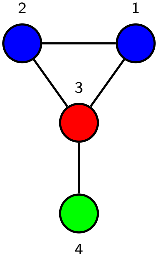


Motivation: Permutation of Adjacency Matrices

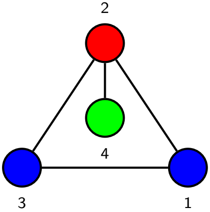


isomorphic graphs, different drawings,

Motivation: Permutation of Adjacency Matrices



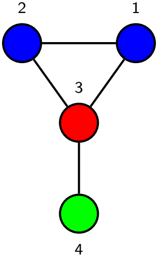
$$\begin{bmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

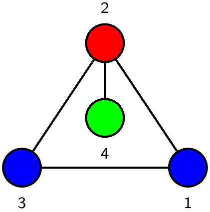


$$\begin{bmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$$

isomorphic graphs, different drawings,

Motivation: Permutation of Adjacency Matrices



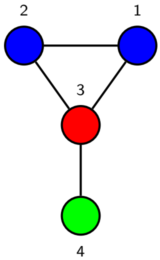


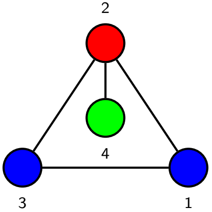
isomorphic graphs, different drawings,

$$\begin{bmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} \qquad \begin{bmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$$

different numbering of vertices, different adjacency matrices

Motivation: Permutation of Adjacency Matrices





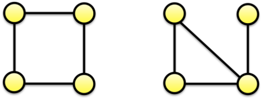
isomorphic graphs, different drawings,

$$\begin{bmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} \qquad \begin{bmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$$

different numbering of vertices, different adjacency matrices

Aim: Permute the vertex set of one of the graphs in order to minimize the number of edge mismatches (0 vs. 1)

Frobenius Distance



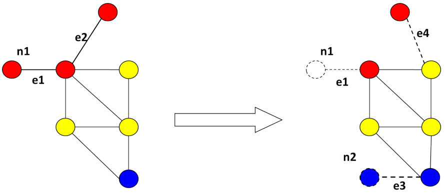
$$\begin{pmatrix} 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{pmatrix} \quad \begin{pmatrix} 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{pmatrix}$$

- Idea: Permute vertex set of G in order to minimize the number of edge mismatches w.r.t. H
- Given G and H with their adjacency matrices A and B
- Search a permutation of rows and columns of A minimizing the Frobenius norm of the matrix $A_\pi - B$:

$$\|A_\pi - B\| = \sqrt{\sum \sum (a_{ij}^\pi - b_{ij})^2}$$

- NP-hard even if G and H are trees or if G is a path [Grohe et al. 2018]

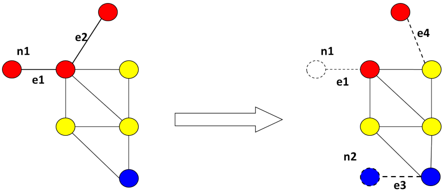
Graph Edit Distance



Idea

- Compute the minimum cost to transform G into H
- allowed operations, e.g.,
  - vertex or edge insertion
  - vertex or edge deletion
  - vertex or edge substitution
- graph editing problem is NP-hard
- used in Cheminformatics and Bioinformatics

Motivation for Graph Edit Distance



Widely used, since

- it can precisely capture the structural differences between graphs
- it is very flexible due to arbitrary edit costs
- error-tolerant
- controllable sensitivity to changes
- can be applied to all types of graphs

Source: gedevo.mpi-inf.mpg.de

Edit Path for Labelled Graphs

Definition (Cost of an edit path)

Edit Path for Labelled Graphs

Definition (Cost of an edit path)

- A labeled graph is denoted as $G = (V, E, L_V, L_E)$, where L_V and L_E are label functions that assign labels to vertices and edges, resp.

Edit Path for Labelled Graphs

Definition (Cost of an edit path)

- A labeled graph is denoted as $G = (V, E, L_V, L_E)$, where L_V and L_E are label functions that assign labels to vertices and edges, resp.
- Given are two labeled graphs. An edit path is given by a sequence of primitive edit operations to transform G_1 to G_2, such as $G_1 = G_1^0 \rightarrow G_1^1 \rightarrow \cdots \rightarrow G_1^k = G_2$.

Edit Path for Labelled Graphs

Definition (Cost of an edit path)

- A labeled graph is denoted as $G = (V, E, L_V, L_E)$, where L_V and L_E are label functions that assign labels to vertices and edges, resp.
- Given are two labeled graphs. An edit path is given by a sequence of primitive edit operations to transform G_1 to G_2, such as $G_1 = G_1^0 \rightarrow G_1^1 \rightarrow \cdots \rightarrow G_1^k = G_2$.
- Primitive edit operations are
  - deleting $u_i \rightarrow \epsilon$ or inserting an α-labeled node $(\epsilon \rightarrow u_i)$
  - deleting or inserting an α-labeled edge
  - changing a node's $(u_i \rightarrow v_j)$ or an edge's label from α to β

Edit Path for Labelled Graphs

Definition (Cost of an edit path)

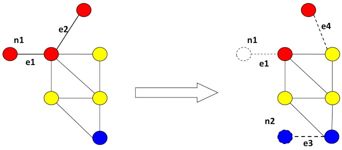
- A labeled graph is denoted as $G = (V, E, L_V, L_E)$, where L_V and L_E are label functions that assign labels to vertices and edges, resp.
- Given are two labeled graphs. An edit path is given by a sequence of primitive edit operations to transform G_1 to G_2, such as $G_1 = G_1^0 \to G_1^1 \to \cdots \to G_1^k = G_2$.
- Primitive edit operations are
  - deleting $u_i \to \epsilon$ or inserting an α-labeled node $(\epsilon \to u_i)$
  - deleting or inserting an α-labeled edge
  - changing a node's $(u_i \to v_j)$ or an edge's label from α to β
- Edit operations on vertices and edges come with associated non-negative edit costs c_V and c_E.

Edit Path for Labelled Graphs

Definition (Cost of an edit path)

- A labeled graph is denoted as $G = (V, E, L_V, L_E)$, where L_V and L_E are label functions that assign labels to vertices and edges, resp.
- Given are two labeled graphs. An edit path is given by a sequence of primitive edit operations to transform G_1 to G_2, such as $G_1 = G_1^0 \to G_1^1 \to \cdots \to G_1^k = G_2$.
- Primitive edit operations are
  - deleting $u_i \to \epsilon$ or inserting an α-labeled node $(\epsilon \to u_i)$
  - deleting or inserting an α-labeled edge
  - changing a node's $(u_i \to v_j)$ or an edge's label from α to β
- Edit operations on vertices and edges come with associated non-negative edit costs c_V and c_E.
- The cost of an edit path is defined as the sum of the costs of its edit operations.

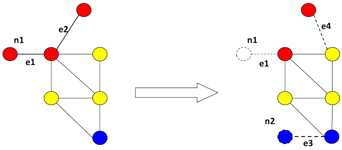
Graph Edit Distance



Definition (Graph Edit Distance (GED))

- The GED(G_1, G_2) of two labeled graphs G_1 and G_2 is defined as the minimum cost of an edit path from G_1 to G_2.

Graph Edit Distance

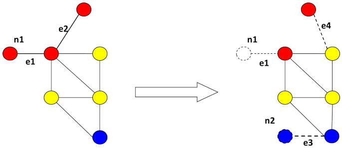


Definition (Graph Edit Distance (GED))

- The GED(G_1, G_2) of two labeled graphs G_1 and G_2 is defined as the minimum cost of an edit path from G_1 to G_2.

- GEDs are not unique in general.
- Algorithms for GED restrict their attention to those edit paths induced by a node map between G and H.

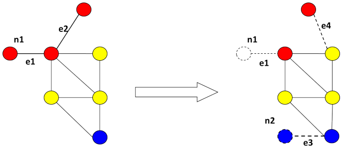
Approaches for computing an optimal GED



Approaches for computing an optimal GED

- A^* algorithm for graph edit distance
- Integer Linear Programming for GEDs

Approaches for computing an optimal GED



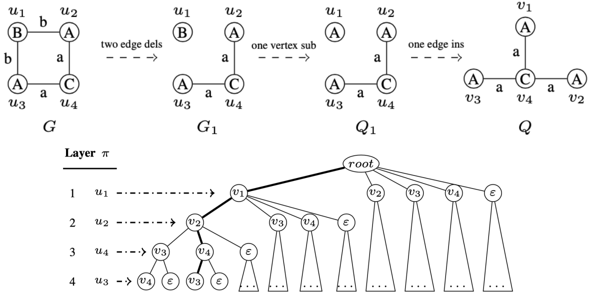
Approaches for computing an optimal GED

- A^* algorithm for graph edit distance
- Integer Linear Programming for GEDs

Algorithms for GED restrict their attention to those edit paths induced by a node map between G and H.

A^* Algorithm for Graph Edit Distance

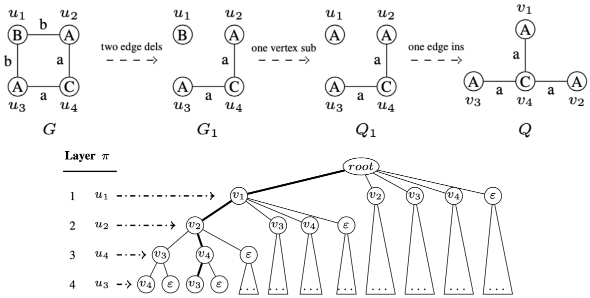
Idea of A^*



A^* Algorithm for Graph Edit Distance

Idea of A^*

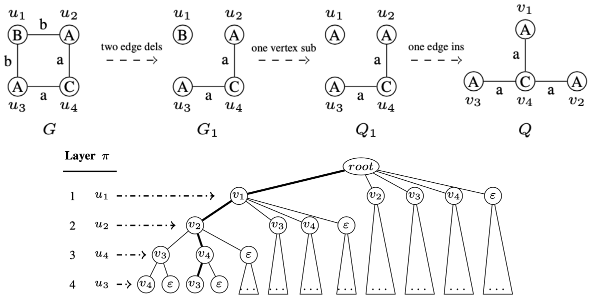
- compute all possible mappings between the vertices of the given two graphs by means of an ordered search tree



A^* Algorithm for Graph Edit Distance

Idea of A^*

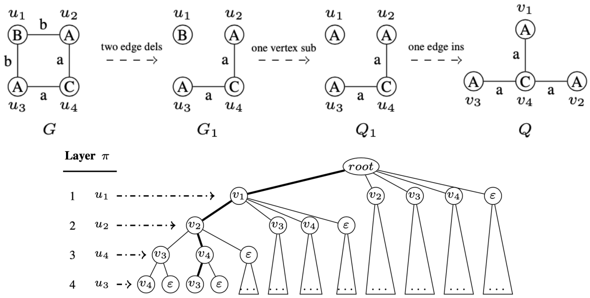
- compute all possible mappings between the vertices of the given two graphs by means of an ordered search tree
- vertices of G_1 are processed in the order $\{u_1, \ldots, u_{|V_1|}\}$



A^* Algorithm for Graph Edit Distance

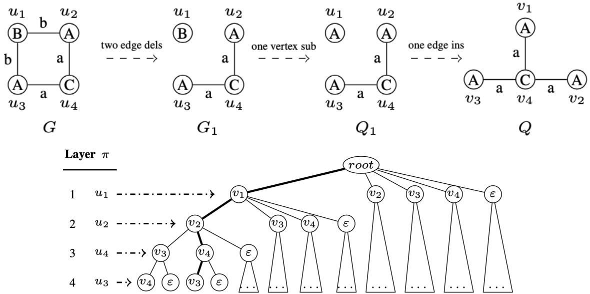
Idea of A^*

- compute all possible mappings between the vertices of the given two graphs by means of an ordered search tree
- vertices of G_1 are processed in the order $\{u_1, \ldots, u_{|V_1|}\}$
- start with u_1, and iteratively construct partial edit paths mapping u_i to vertices $v_j, j = 1, \ldots, |V_2|$ [source: Chen et al. 2019]



A^* Algorithm for Graph Edit Distance

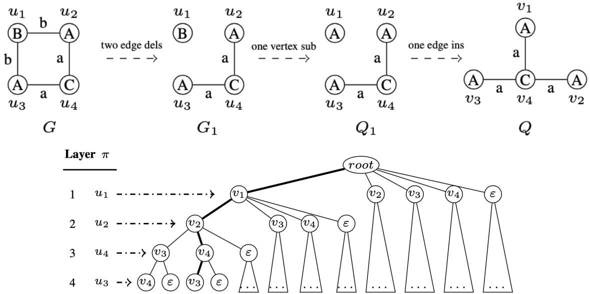
Idea of best-first search paradigm A^*



A^* Algorithm for Graph Edit Distance

Idea of best-first search paradigm A^*

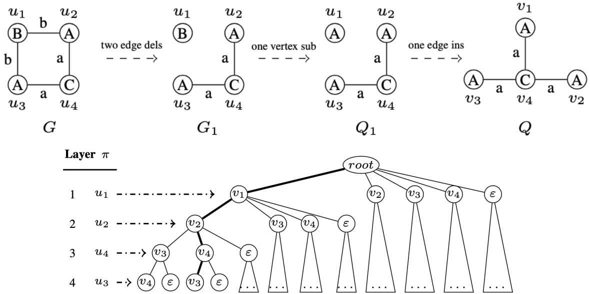
- let p be a partial edit path from G_1 to current vertex v



A^* Algorithm for Graph Edit Distance

Idea of best-first search paradigm A^*

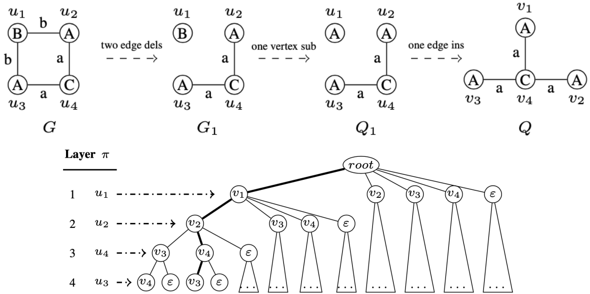
- let p be a partial edit path from G_1 to current vertex v
- let $g(p)$ be the cost of p from G_1 to current vertex v,



A^* Algorithm for Graph Edit Distance

Idea of best-first search paradigm A^*

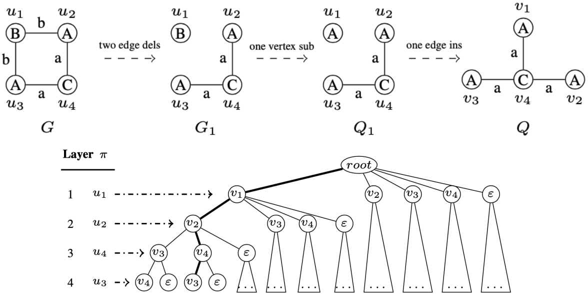
- let p be a partial edit path from G_1 to current vertex v
- let $g(p)$ be the cost of p from G_1 to current vertex v,
- and $h(p)$ be an estimated cost from v to G_2 (a leaf node in tree)



A^* Algorithm for Graph Edit Distance

Idea of best-first search paradigm A^*

- let p be a partial edit path from G_1 to current vertex v
- let $g(p)$ be the cost of p from G_1 to current vertex v,
- and $h(p)$ be an estimated cost from v to G_2 (a leaf node in tree)
- for further expansion choose the partial edit path p that minimizes $g(p) + h(p)$



Estimated cost of edit paths

For ease of notation we assume unit edit costs 1 for all edit operations

Possible approaches for estimating $h(p)$

- label set-based lower bound

Estimated cost of edit paths

For ease of notation we assume unit edit costs 1 for all edit operations

Possible approaches for estimating $h(p)$

- label set-based lower bound
  - compare the labels of the remaining vertices and edges
  - sum up the difference (e.g. 5 vs. 3 α labels $\rightarrow$ 2)

Estimated cost of edit paths

For ease of notation we assume unit edit costs 1 for all edit operations

Possible approaches for estimating $h(p)$

- label set-based lower bound
  - compare the labels of the remaining vertices and edges
  - sum up the difference (e.g. 5 vs. 3 α labels $\rightarrow$ 2)
- star match-based lower bound

Estimated cost of edit paths

For ease of notation we assume unit edit costs 1 for all edit operations

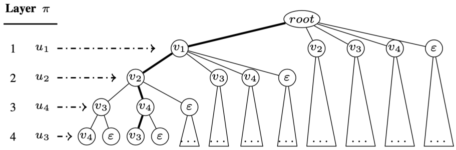
Possible approaches for estimating $h(p)$

- label set-based lower bound
  - compare the labels of the remaining vertices and edges
  - sum up the difference (e.g. 5 vs. 3 α labels $\rightarrow$ 2)
- star match-based lower bound
  - build stars of the remaining vertices by adding the direct neighbors
  - compare the stars using the Hungarian algorithm (weighted bipartite matching)

Discussion

Analysis and Improvements

- the worst case running time is $n!$

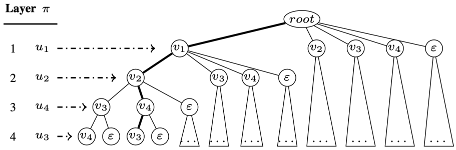


Details: see Chen et al. 2019

Discussion

Analysis and Improvements

- the worst case running time is $n!$
- $\rightarrow$ only very small graph instances can be computed exactly

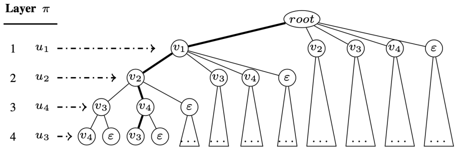


Details: see Chen et al. 2019

Discussion

Analysis and Improvements

- the worst case running time is $n!$
- $\rightarrow$ only very small graph instances can be computed exactly
- reduce the search space

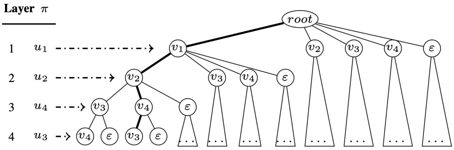


Details: see Chen et al. 2019

Discussion

Analysis and Improvements

- the worst case running time is n!
- $\rightarrow$ only very small graph instances can be computed exactly
- reduce the search space
  - identify redundant and invalid mappings
  - prune the search space
  - heuristic improvement: beam search (only follow a constant number of most promising partial edit paths)



Details: see Chen et al. 2019

1. **Introduction**

2. **Isomorphism-based Approaches**
   - Graph isomorphism
   - Subgraph Isomorphism based Approaches
   - Structural Clustering of Sets of Graphs

3. **Frobenius Distance**

4. **Graph Edit Distance**
   - A^* Algorithm for GED
   - Integer Linear Programs for GED
   - Computational Study

Integer Quadratic Program IQP-GED

GED can naturally be written as an integer quadratic program (IQP).

Introducing dummy nodes:

- Let V^{G+0} denote V^G extended by the dummy node for vertex insertion.
- Let V^{H+0} denote V^H extended by the dummy node for vertex deletion.
- A mapping from dummy node ϵ denotes an insertion to V^H, and
- a mapping to ϵ corresponds to a deletion in V^G.

Integer Quadratic Program IQP-GED

GED can naturally be written as an integer quadratic program (IQP).

Introducing dummy nodes:

- Let V^{G+0} denote V^G extended by the dummy node for vertex insertion.
- Let V^{H+0} denote V^H extended by the dummy node for vertex deletion.
- A mapping from dummy node ϵ denotes an insertion to V^H, and
- a mapping to ϵ corresponds to a deletion in V^G.

Binary variables:

- For all $i \in V^{G+0}, k \in V^{H+0}$ we introduce variables $x_{i,k} = 1$ $\Leftrightarrow$ node i is mapped to k

Integer Quadratic Program IQP-GED

Binary variables:

- For all $i \in V^{G+0}, k \in V^{H+0}$ we introduce variables $x_{i,k} = 1$ $\Leftrightarrow$ node i is mapped to k

$$\min \sum_{i \in V^{G+0}} \sum_{k \in V^{H+0}} c_V(i,k) x_{i,k} + \sum_{i,j \in V^{G+0}} \sum_{k,l \in V^{H+0}} c_E(ij,kl) x_{i,k} x_{j,l}$$

$$\sum_{k \in V^{H+0}} x_{i,k} = 1 \quad \forall i \in V^G \tag{1}$$

$$\sum_{i \in V^{G+0}} x_{i,k} = 1 \quad \forall k \in V^H \tag{2}$$

$$x_{i,k} \in \{0,1\} \quad \forall (i,k) \in V^{G+0} \times V^{H+0} \tag{3}$$

Integer Quadratic Program IQP-GED

Binary variables:

- For all $i \in V^{G+0}$, $k \in V^{H+0}$ we introduce variables $x_{i,k} = 1$ $\Leftrightarrow$ node i is mapped to k

$$\min \sum_{i \in V^{G+0}} \sum_{k \in V^{H+0}} c_V(i,k) x_{i,k} + \sum_{i,j \in V^{G+0}} \sum_{k,l \in V^{H+0}} c_E(ij,kl) x_{i,k} x_{j,l}$$

$$\sum_{k \in V^{H+0}} x_{i,k} = 1 \qquad \forall i \in V^G \tag{1}$$

$$\sum_{i \in V^{G+0}} x_{i,k} = 1 \qquad \forall k \in V^H \tag{2}$$

$$x_{i,k} \in \{0,1\} \quad \forall (i,k) \in V^{G+0} \times V^{H+0} \tag{3}$$

Observation

If both graphs have n vertices, then the IQP-GED formulation contains $(n+1)^2$ binary variables and $(n+1)^2 + 2n$ constraints.

Integer Quadratic Program IQP-GED

Lemma

Let G and H be graphs and (x^*) be an optimal solution to the IQP-GED.

1. Then (x^*) corresponds to a mapping of each vertex in G to either a vertex in H or to a deletion (V^{H+0}), and vice versa: A mapping to each vertex in H from either a vertex in G or from an insertion (V^{G+0}) (whereby a mapping from ϵ denotes an insertion, and a mapping to ϵ corresponds to a deletion).

Integer Quadratic Program IQP-GED

Lemma

Let G and H be graphs and (x^*) be an optimal solution to the IQP-GED.

1. Then (x^*) corresponds to a mapping of each vertex in G to either a vertex in H or to a deletion (V^{H+0}), and vice versa: A mapping to each vertex in H from either a vertex in G or from an insertion (V^{G+0}) (whereby a mapping from ϵ denotes an insertion, and a mapping to ϵ corresponds to a deletion).
2. Such a mapping induces a set of feasible edit paths from G to H.

Integer Quadratic Program IQP-GED

Lemma

Let G and H be graphs and (x^*) be an optimal solution to the IQP-GED.

1. Then (x^*) corresponds to a mapping of each vertex in G to either a vertex in H or to a deletion (V^{H+0}), and vice versa: A mapping to each vertex in H from either a vertex in G or from an insertion (V^{G+0}) (whereby a mapping from ϵ denotes an insertion, and a mapping to ϵ corresponds to a deletion).
2. Such a mapping induces a set of feasible edit paths from G to H.
3. The costs of each edit path induced by the mapping (x^*) is equal to the objective value of the IQP-GED.

Integer Quadratic Program IQP-GED

Lemma

Let G and H be graphs and (x^*) be an optimal solution to the IQP-GED.

1. Then (x^*) corresponds to a mapping of each vertex in G to either a vertex in H or to a deletion (V^{H+0}), and vice versa: A mapping to each vertex in H from either a vertex in G or from an insertion (V^{G+0}) (whereby a mapping from ϵ denotes an insertion, and a mapping to ϵ corresponds to a deletion).
2. Such a mapping induces a set of feasible edit paths from G to H.
3. The costs of each edit path induced by the mapping (x^*) is equal to the objective value of the IQP-GED.
4. Every edit path induces a mapping from V^{G+0} to V^{H+0}, satisfies constraints (1) – (3) of IQP-GED, and the costs coincide.

Integer Quadratic Program IQP-GED

Lemma

Let G and H be graphs and (x^*) be an optimal solution to the IQP-GED.

1. Then (x^*) corresponds to a mapping of each vertex in G to either a vertex in H or to a deletion (V^{H+0}), and vice versa: A mapping to each vertex in H from either a vertex in G or from an insertion (V^{G+0}) (whereby a mapping from ϵ denotes an insertion, and a mapping to ϵ corresponds to a deletion).
2. Such a mapping induces a set of feasible edit paths from G to H.
3. The costs of each edit path induced by the mapping (x^*) is equal to the objective value of the IQP-GED.
4. Every edit path induces a mapping from V^{G+0} to V^{H+0}, satisfies constraints (1) – (3) of IQP-GED, and the costs coincide.

Notice: We do not require constraints (1) and (2) for the dummy vertices, since we introduced exactly one dummy vertex for G and one for H.

Linearization of Integer Quadratic Programs

Quadratic programs with linear constraints can be transformed into linear programs. Example:

$$
\begin{aligned}
\min \sum_{u \in V} \sum_{v \in V} z_u z_v + \sum_{u \in V} c_u z_u & & & \qquad (4)\\
\sum_{u \in V} z_u & = 1 & \forall v \in V & \qquad (5)\\
\sum_{v \in V} z_v & = 1 & \forall u \in V & \qquad (6)\\
z_u & \in \{0,1\} & \forall (u,v) \in V \times V & \qquad (7)
\end{aligned}
$$

Linearization of Integer Quadratic Programs

Quadratic programs with linear constraints can be transformed into linear programs. Example:

$$\min \sum_{u \in V} \sum_{v \in V} z_u z_v + \sum_{u \in V} c_u z_u \tag{4}$$

$$\sum_{u \in V} z_u = 1 \quad \forall v \in V \tag{5}$$

$$\sum_{v \in V} z_v = 1 \quad \forall u \in V \tag{6}$$

$$z_u \in \{0,1\} \quad \forall (u,v) \in V \times V \tag{7}$$

Introduce new binary variables and constraints:

- For all $u, v \in V$ we introduce new variables $y_{u,v} = 1$ $\Leftrightarrow z_u = 1$ and $z_v = 1$

Linearization of Integer Quadratic Programs

Quadratic programs with linear constraints can be transformed into linear programs. Example:

$$\min \sum_{u \in V} \sum_{v \in V} z_u z_v + \sum_{u \in V} c_u z_u \tag{4}$$

$$\sum_{u \in V} z_u = 1 \quad \forall v \in V \tag{5}$$

$$\sum_{v \in V} z_v = 1 \quad \forall u \in V \tag{6}$$

$$z_u \in \{0,1\} \quad \forall (u,v) \in V \times V \tag{7}$$

Introduce new binary variables and constraints:

- For all $u, v \in V$ we introduce new variables $y_{u,v} = 1 \Leftrightarrow z_u = 1$ and $z_v = 1$
- We need additional constraints that guarantee the above rule during the optimization process. Here: $y_{u,v} \geq z_u + z_v - 1$.

Back to: Integer Quadratic Program IQP-GED

GED can naturally be written as an integer quadratic program.

Binary variables:

- For all $i \in V^{G+0}, k \in V^{H+0}$ we introduce variables $x_{i,k} = 1$
  $\Leftrightarrow$ node *i* is mapped to *k*

$$\min \sum_{i \in V^{G+0}} \sum_{k \in V^{H+0}} c_V(i,k) x_{i,k} + \sum_{i,j \in V^{G+0}} \sum_{k,l \in V^{H+0}} c_E(ij,kl) x_{i,k} x_{j,l}$$

$$\sum_{k \in V^{H+0}} x_{i,k} = 1 \quad \forall i \in V^G \tag{8}$$

$$\sum_{i \in V^{G+0}} x_{i,k} = 1 \quad \forall k \in V^H \tag{9}$$

$$x_{i,k} \in \{0,1\} \quad \forall (i,k) \in V^{G+0} \times V^{H+0} \tag{10}$$

Standard Linearization of IQP-GED: LIQP-GED

Binary variables:

- For all $i \in V^{G+0}, k \in V^{H+0}$ we introduce variables $x_{i,k} = 1$
  $\Leftrightarrow$ node i is mapped to k

Standard Linearization of IQP-GED: LIQP-GED

Binary variables:

- For all $i \in V^{G+0}, k \in V^{H+0}$ we introduce variables $x_{i,k} = 1$
  $\Leftrightarrow$ node i is mapped to k
- For all $i, j \in V^{G+0}, k, l \in V^{H+0}$ we introduce variables $y_{i,k,j,l} = 1$
  $\Leftrightarrow$ $x_{i,k} = 1$ and $x_{j,l} = 1$

Standard Linearization of IQP-GED: LIQP-GED

Binary variables:

- For all $i \in V^{G+0}$, $k \in V^{H+0}$ we introduce variables $x_{i,k} = 1$
  $\Leftrightarrow$ node i is mapped to k
- For all $i, j \in V^{G+0}$, $k, l \in V^{H+0}$ we introduce variables $y_{i,k,j,l} = 1$
  $\Leftrightarrow$ $x_{i,k} = 1$ and $x_{j,l} = 1$

$$\min \sum_{i \in V^{G+0}} \sum_{k \in V^{H+0}} c_V(i,k) x_{i,k} + \sum_{i,j \in V^{G+0}} \sum_{k,l \in V^{H+0}} c_E(ij, kl) y_{i,k,j,l}$$

$$\sum_{k \in V^{H+0}} x_{i,k} = 1 \qquad \forall i \in V^G \tag{11}$$

$$\sum_{i \in V^{G+0}} x_{i,k} = 1 \qquad \forall k \in V^H \tag{12}$$

$$x_{i,k} + x_{j,l} - y_{i,k,j,l} \leq 1 \qquad \forall i,j \in V^{G+0}, \forall k,l \in V^{H+0} \tag{13}$$

$$x_{i,k} \in \{0,1\} \qquad \forall (i,k) \in V^{G+0} \times V^{H+0} \tag{14}$$

$$y_{i,k,j,l} \in \{0,1\} \qquad \forall i,j \in V^{G+0}, \forall k,l \in V^{H+0} \tag{15}$$

Standard Linearization of IQP-GED: LIQP-GED

Lemma

The formulation LIQP-GED is equivalent to IQP-GED, in particular, a feasible solution of LIQP-GED corresponds to a feasible solution IQP-GED, and vice versa. The cost of the optimal solution is the same in both cases.

Proof: $\Rightarrow$:

- Let (x', y') be a feasible solution to LIQP-GED. From this we take the first part and claim that x' is also a feasible solution of IQP-GED, since it satisfies constraints (8) to (10).

Standard Linearization of IQP-GED: LIQP-GED

Lemma

The formulation LIQP-GED is equivalent to IQP-GED, in particular, a feasible solution of LIQP-GED corresponds to a feasible solution IQP-GED, and vice versa. The cost of the optimal solution is the same in both cases.

Proof: $\Rightarrow$:

- Let (x', y') be a feasible solution to LIQP-GED. From this we take the first part and claim that x' is also a feasible solution of IQP-GED, since it satisfies constraints (8) to (10).
- The first part of the objective functions is the same in both formulations, we will concentrate on the second part.

Standard Linearization of IQP-GED: LIQP-GED

Lemma

The formulation LIQP-GED is equivalent to IQP-GED, in particular, a feasible solution of LIQP-GED corresponds to a feasible solution IQP-GED, and vice versa. The cost of the optimal solution is the same in both cases.

Proof: $\Rightarrow$:

- Let (x', y') be a feasible solution to LIQP-GED. From this we take the first part and claim that x' is also a feasible solution of IQP-GED, since it satisfies constraints (8) to (10).
- The first part of the objective functions is the same in both formulations, we will concentrate on the second part.
- In the case that $y'_{i,k,j,l} = 0$, then because of (13) we have that either $x'_{i,k} = 0$ or $x'_{j,l} = 0$. But then this leads to a contribution of 0 in both objective functions.

Standard Linearization of IQP-GED: LIQP-GED

Proof: $\Leftarrow$:

- Let (x') be a feasible solution to IQP-GED. From this we assign a vector (x', y') with $y'_{i,k,j,l} = x'_{i,k} x'_{j,l}$ and claim that it is feasible for LIQP-GED. Since $0 \leq y'_{i,k,j,l} \leq 1$, constraint (13) is valid.

Standard Linearization of IQP-GED: LIQP-GED

Proof: $\Leftarrow$:

- Let (x') be a feasible solution to IQP-GED. From this we assign a vector (x', y') with $y'_{i,k,j,l} = x'_{i,k}x'_{j,l}$ and claim that it is feasible for LIQP-GED. Since $0 \leq y'_{i,k,j,l} \leq 1$, constraint (13) is valid.
- The first part of the objective functions is the same in both formulations, we will concentrate on the second part.

Standard Linearization of IQP-GED: LIQP-GED

Proof: $\Leftarrow$:

- Let (x') be a feasible solution to IQP-GED. From this we assign a vector (x', y') with $y'_{i,k,j,l} = x'_{i,k}x'_{j,l}$ and claim that it is feasible for LIQP-GED. Since $0 \leq y'_{i,k,j,l} \leq 1$, constraint (13) is valid.
- The first part of the objective functions is the same in both formulations, we will concentrate on the second part.
- In the case that $y'_{i,k,j,l} = 0$, the contribution to both objective functions is 0.

Standard Linearization of IQP-GED: LIQP-GED

Proof: $\Leftarrow$:

- Let (x') be a feasible solution to IQP-GED. From this we assign a vector (x', y') with $y'_{i,k,j,l} = x'_{i,k}x'_{j,l}$ and claim that it is feasible for LIQP-GED. Since $0 \leq y'_{i,k,j,l} \leq 1$, constraint (13) is valid.
- The first part of the objective functions is the same in both formulations, we will concentrate on the second part.
- In the case that $y'_{i,k,j,l} = 0$, the contribution to both objective functions is 0.
- Otherwise, $y'_{i,k,j,l} = 1$, and we have $x'_{i,k} = x'_{j,l} = 1$. In both objective functions, we get the same contribution of $c_E(ij, kl)$.

Standard Linearization of IQP-GED: LIQP-GED

Proof: $\Leftarrow$:

- Let (x') be a feasible solution to IQP-GED. From this we assign a vector (x', y') with $y'_{i,k,j,l} = x'_{i,k} x'_{j,l}$ and claim that it is feasible for LIQP-GED. Since $0 \leq y'_{i,k,j,l} \leq 1$, constraint (13) is valid.
- The first part of the objective functions is the same in both formulations, we will concentrate on the second part.
- In the case that $y'_{i,k,j,l} = 0$, the contribution to both objective functions is 0.
- Otherwise, $y'_{i,k,j,l} = 1$, and we have $x'_{i,k} = x'_{j,l} = 1$. In both objective functions, we get the same contribution of $c_E(ij, kl)$.

Observation

If both graphs have n vertices, then the LIQP-GED formulation contains $(n+1)^2 + (n+1)^4$ binary variables and $(n+1)^2 + 2(n+1)^4 + 2n$ constraints.

Alternative Binary Integer Linear Program: BIP-GED

Formulation by Lerouge et al. 2017

Binary variables:

- For all $i \in V^G$, $k \in V^H$ we introduce variables $x_{i,k} = 1$ $\Leftrightarrow$ node i is mapped to k
- For all edges $(i, j) \in E^G$ and $(k, l) \in E^H$ we introduce variables $w_{ij,kl} = 1 \Leftrightarrow$ edge (i, j) is mapped to edge (k, l)

Binary Integer Linear Program BIP-GED

$$\min \sum_{i \in V^G} \sum_{k \in V^H} (c_{i,k} - c_{i,\epsilon} - c_{\epsilon,k}) x_{i,k} + \sum_{ij \in E^G} \sum_{kl \in E^H} (c_{ij,kl} - c_{ij,\epsilon} - c_{\epsilon,kl}) w_{ij,kl} + C$$

.

$$\sum_{k \in V^H} x_{i,k} \leq 1 \qquad \forall i \in V^G \tag{16}$$

$$\sum_{i \in V^G} x_{i,k} \leq 1 \qquad \forall k \in V^H \tag{17}$$

$$\sum_{l:(k,l) \in E^H} w_{ij,kl} - x_{i,k} - x_{j,k} \leq 0 \qquad \forall k \in V^H, \forall (i,j) \in E^G \tag{18}$$

$$x_{i,k} \in \{0,1\} \qquad \forall (i,k) \in V^G \times V^H \tag{19}$$

$$w_{ij,kl} \in \{0,1\} \qquad \forall (i,j) \in E^G, \forall (k,l) \in E^H \tag{20}$$

with constant

$$C = \sum_{i \in V^G} c_{i,\epsilon} + \sum_{k \in V^H} c_{\epsilon,k} + \sum_{ij \in E^G} c_{ij,\epsilon} + \sum_{kl \in E^H} c_{\epsilon,kl}$$

Binary Integer Linear Program BIP-GED

Lemma

Let G and H be graphs and (x^*, w^*) be an optimal solution to the BIP-GED with value z^*. Then we have

$$GED(G, H) = z^*.$$

Binary Integer Linear Program BIP-GED

Lemma

Let G and H be graphs and (x^*, w^*) be an optimal solution to the BIP-GED with value z^*. Then we have

$$GED(G, H) = z^*.$$

Observation

If both graphs have n vertices and m edges, then the BIP-GED formulation has $n^2 + m^2$ variables and $n^2 + m^2 + nm + 2n$ constraints.

Mixed Integer Linear Program MIP-GED

Binary variables x and continuous variables z:

- For all $i \in V^{G+0}, k \in V^{H+0}$ we introduce variables $x_{i,k} = 1$ $\Leftrightarrow$ node (or dummy node) i is mapped to k
- For all $i \in V^{G+0}, k \in V^{H+0}$ we introduce variables $z_{i,k}$ that contain the edit cost that is induced by mapping i to k, given all other node assignments.

Mixed Integer Linear Program MIP-GED

Binary variables x and continuous variables z:

- For all $i \in V^{G+0}, k \in V^{H+0}$ we introduce variables $x_{i,k} = 1$ $\Leftrightarrow$ node (or dummy node) i is mapped to k
- For all $i \in V^{G+0}, k \in V^{H+0}$ we introduce variables $z_{i,k}$ that contain the edit cost that is induced by mapping i to k, given all other node assignments.

The constants $u_{i,k}$ are defined as

$$u_{i,k} = c_V(i,k) + \sum_{j \in V^{G+0}} \sum_{l \in V^{H+0}} \frac{c_E(ij, kl)}{2}$$

Mixed Integer Linear Program MIP-GED

$$\min \sum_{i \in V^{G+0}} \sum_{k \in V^{H+0}} z_{i,k}$$

$$\begin{aligned}
\sum_{k \in V^{H+0}} x_{i,k} &= 1 && \forall i \in V^G && (21)\\
\sum_{i \in V^{G+0}} x_{i,k} &= 1 && \forall k \in V^H && (22)\\
\sum_{j \in V^{G+0}} \sum_{l \in V^{H+0}} \frac{c_E(ij, kl)}{2} x_{j,l} & && && (23)\\
+ c_V(i,k) - (1 - x_{i,k}) u_{i,k} &\leq z_{i,k} && \forall (i,k) \in V^{G+0} \times V^{H+0} && (24)\\
x_{i,k} &\in \{0,1\} && \forall (i,k) \in V^{G+0} \times V^{H+0} && (25)\\
z_{i,k} &\geq 0 && \forall (i,k) \in V^{G+0} \times V^{H+0} && (26)
\end{aligned}$$

Mixed Integer Linear Program MIP-GED

Lemma

Let G and H be graphs and (x^*, z^*) be an optimal solution to the MIP-GED. Then we have

$$GED(G, H) = \sum_{i \in V^{G+0}} \sum_{k \in V^{H+0}} z^*_{i,k}.$$

Mixed Integer Linear Program MIP-GED

Lemma

Let G and H be graphs and (x^*, z^*) be an optimal solution to the MIP-GED. Then we have

$$GED(G,H) = \sum_{i \in V^{G+0}} \sum_{k \in V^{H+0}} z^*_{i,k}.$$

Proof: Case 1: $x^*_{i,k} = 0$ for $i \in V^{G+0}$ and $k \in V^{H+0}$

- Constraint (24) gives:

$$z_{i,k} \geq \sum_{j \in V^{G+0}} \sum_{l \in V^{H+0}} \frac{c_E(ij, kl)}{2} x_{j,l} + c_V(i,k) - u_{i,k} = 0$$

- Since the objective function minimizes the (sum of the) z-values, it will end up with $z_{i,k} = 0$. The contribution to the objective function is 0 as in the formulation IQP-GED.

Proof of Lemma ff

Case 2: $x^*_{i,k} = 1$ for $i \in V^{G+0}$ and $k \in V^{H+0}$

- Constraint (24) gives:

$$z_{i,k} \geq \sum_{j \in V^{G+0}} \sum_{l \in V^{H+0}} \frac{c_E(ij, kl)}{2} x_{j,l} + c_V(i, k)$$

- For all $x_{jl} = 0$ the contribution to the sum is 0, which is true also for the objective function of IQP-GED.
- For all $x_{jl} = 1$ the contribution to the sum is $\frac{c_E(ij,kl)}{2}$
- $c_E(ij, kl)$ is the edit cost for changing edge (i, j) to (k, l).
- Constraint (24) gives half of that to $z_{i,k}$ and half of it to $z_{j,l}$.
- The sum of the contribution to the objective function for $z_{i,k}$ and for $z_{j,l}$ is exactly the same as in the formulation IQP-GED.

Proof of Lemma ff

Case 2: $x^*_{i,k} = 1$ for $i \in V^{G+0}$ and $k \in V^{H+0}$

- Constraint (24) gives:

$$z_{i,k} \geq \sum_{j \in V^{G+0}} \sum_{l \in V^{H+0}} \frac{c_E(ij,kl)}{2} x_{j,l} + c_V(i,k)$$

- For all $x_{jl} = 0$ the contribution to the sum is 0, which is true also for the objective function of IQP-GED.
- For all $x_{jl} = 1$ the contribution to the sum is $\frac{c_E(ij,kl)}{2}$
- $c_E(ij,kl)$ is the edit cost for changing edge (i,j) to (k,l).
- Constraint (24) gives half of that to $z_{i,k}$ and half of it to $z_{j,l}$.
- The sum of the contribution to the objective function for $z_{i,k}$ and for $z_{j,l}$ is exactly the same as in the formulation IQP-GED.

Observation

If both graphs have n vertices, then the MIP-GED formulation contains $2(n+1)^2$ variables and $3(n+1)^2 + 2n$ constraints.

Computational Study by Blumenthal and Gamper

Test Set Up

- Comparison of performance of A^*-based approaches
  - A^*-GED (based on best-first search)
  - DF-GED (basically A^* with depth-first search)
  - CSI-GED (basically edge-based A^*)

  with ILP-based approaches
  - BIP-GED
  - MIP-GED

Source: Blumenthal and Gamper 2020

Computational Study by Blumenthal and Gamper

Test Set Up

- Comparison of performance of A^*-based approaches
  - A^*-GED (based on best-first search)
  - DF-GED (basically A^* with depth-first search)
  - CSI-GED (basically edge-based A^*)

  with ILP-based approaches
  - BIP-GED
  - MIP-GED
- data sets PROTEIN, GREC, LETTERS from the IAM graph database

Source: Blumenthal and Gamper 2020

Computational Study by Blumenthal and Gamper

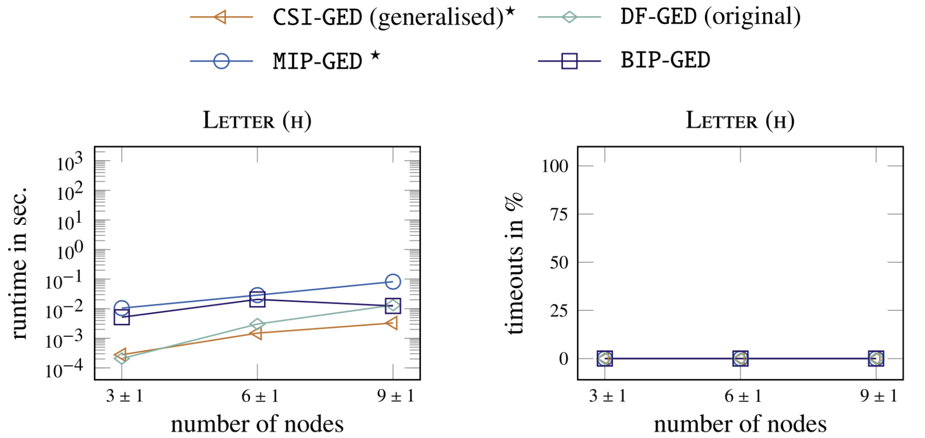
Test Set Up

- Comparison of performance of A^*-based approaches
  - A^*-GED (based on best-first search)
  - DF-GED (basically A^* with depth-first search)
  - CSI-GED (basically edge-based A^*)

  with ILP-based approaches
  - BIP-GED
  - MIP-GED
- data sets PROTEIN, GREC, LETTERS from the IAM graph database
- timeouts: percentage of graph comparisons where the algorithm has not finished within 1000 seconds
- runtime: average runtime across pairwise comparisons

Source: Blumenthal and Gamper 2020

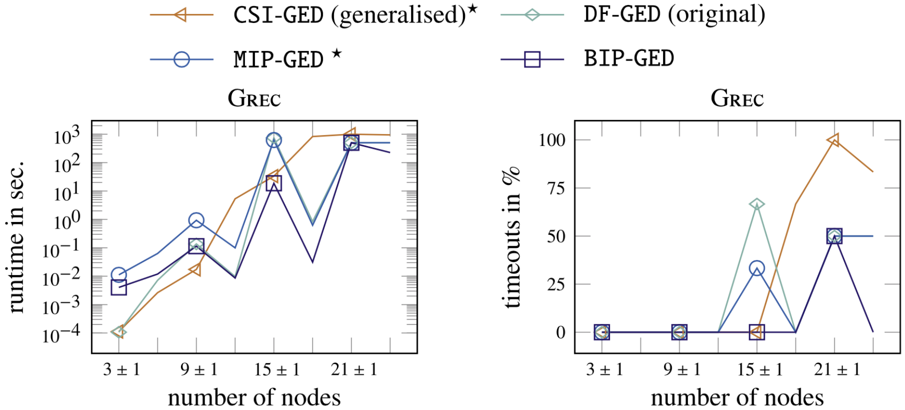
Experimental Results



A^*-GED algorithm often failed and needed much more storage, therefore it is omitted from the plots

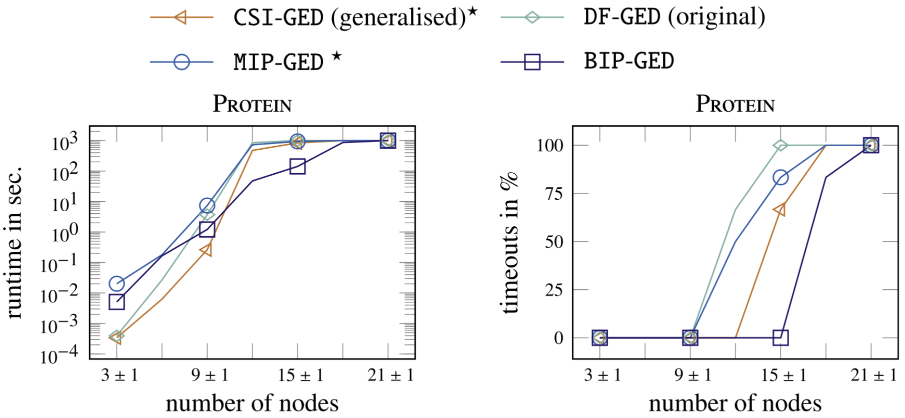
Source: Blumenthal and Gamper 2020

Experimental Results



Source: Blumenthal and Gamper 2020

Experimental Results



Source: Blumenthal and Gamper 2020

Conclusion and Open Problems

- Graph Edit Distance is widely used in practice
- However, exact approaches seem to work for graphs with up to 20 vertices
- Practitioners use heuristics
- new exact approaches necessary

Source: Blumenthal and Gamper 2020