Graph Edit Distance: Basics and History

May 21, 2017

Recognition implies the definition of similarity or dissimilarity measures.

Different measures between graphs:

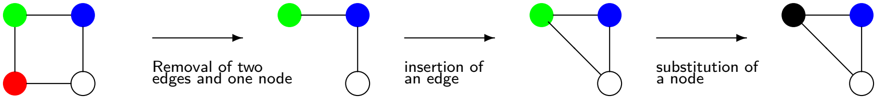
- Distance based on maximum common subgraphs,
- Distance based on spectral characteristics,
- Graph kernels (simmilarities),
- Graph Edit distance.
  - ☹ Not definite negative,
  - ☺Very fine.

Edit Paths

Edit Paths

Definition (Edit path)

Given two graphs G_1 and G_2 an **edit path** between G_1 and G_2 is a sequence of node or edge removal, insertion or substitution which transforms G_1 into G_2.



A substitution is denoted $u \rightarrow v$, an insertion $\epsilon \rightarrow v$ and a removal $u \rightarrow \epsilon$.

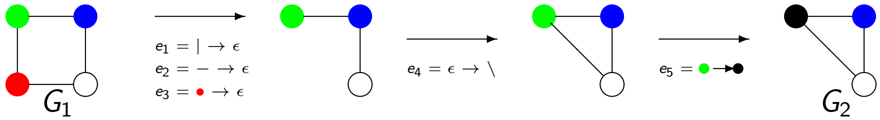
Alternative edit operations such as merge/split have been also proposed[Ambauen et al., 2003].

Costs

All paths go to Roma... However we are usually only interested by the shortest one.

Let $c(.)$ denote the cost of any elementary operation. The cost of an edit path is defined as the sum of the costs of its elementary operations.

- All cost are positive: $c() \geq 0$,
- A node or edge substitution which does not modify a label has a 0 cost: $c(l \rightarrow l) = 0$.



If all costs are equal to 1, the cost of this edit path is equal to 5.

Graph edit distance

Definition (Graph edit distance)

The graph edit distance between G_1 and G_2 is defined as the cost of the less costly path within $\Gamma(G_1, G_2)$. Where $\Gamma(G_1, G_2)$ denotes the set of edit paths between G_1 and G_2.

$$d(G_1, G_2) = \min_{\gamma \in \Gamma(G_1, G_2)} \sum_{e \in \gamma} c(e)$$

Main concepts

Tree search algorithms explore the space $\Gamma(G_1, G_2)$ with some heuristics to avoid to visit unfruitful states. More precisely let us consider a partial edit path p between G_1 and G_2. Let:

- $g(p)$ the cost of the partial edit path.
- $h(p)$ a lower bound of the cost of the remaining part of the path required to reach G_2.

$$\forall \gamma \in \Gamma(G_1, G_2), \gamma = p.q,\ g(p) + h(p) \leq d_\gamma(G_1, G_2)$$

where $d_\gamma(G_1, G_2)$ denotes the cost of the edit path γ.

Main concepts

Let UB denote the best approximation of the GED found so far. If $g(p) + h(p) > UB$ we have:

$$\forall \gamma \in \Gamma(G_1, G_2), \gamma = p.q, d_\gamma(G_1, G_2) \geq g(p) + h(p) > UB$$

In other terms, all the sons of p will provide a greater approximation of the GED and correspond thus to unfruitful nodes.

Choosing a good lower bound

Let us suppose that n_1 and n_2 vertices of respectively V_1 and V_2 remain to be assigned. Different choices are possible for function h[Abu-Aisheh, 2016]:

Null function: $h(p) = 0$,

Bipartite Function: $h(p) = d_{lb}(G_1, G_2)$ (see bellow). Closer bound but requires a $\mathcal{O}(\max\{n_1, n_2\}^3)$ algorithm.

Hausdorff distance $h(p)$ is set to the Hausdroff distance between the sets of n_1 and n_2 vertices (including their incident edges). Requires $\mathcal{O}((n_1 + n_2)^2)$ computation steps.

A^* algorithm

Input: Two graphs G_1 and G_2 with $V_1 = \{u_1, \ldots, u_n\}$ and $V_2 = \{v_1, \ldots, v_m\}$
Output: A minimum edit path between G_1 and G_2
OPEN$=\{u_1 \to \epsilon\} \cup \bigcup_{w \in V_2} \{u_1 \to w\}$
repeat
    $p = \arg\min_{q \in OPEN} \{g(q) + h(q)\}$
    if p is a complete edit path **then**
        return p
    end if
    Complete p by operations on u_{k+1} or on remaining vertices of V_2.
    Add completed paths to OPEN
until end of times

A^* algorithm: discussion

- 🙂 If algorithm A^* terminates it always returns the optimal value of the GED.
- ☹ The set OPEN may be as large as the number of edit paths between G_1 and G_2.
- ☹ The algorithm do not return any result before it finds the optimal solution.

Depth first search algorithm

Input: Two graphs G_1 and G_2 with $V_1 = \{u_1, \ldots, u_n\}$ and $V_2 = \{v_1, \ldots, v_m\}$
Output: γ_{UB} and UB a minimum edit path and its associated cost

$(\gamma_{UB}, UBCOST) = GoodHeuristic(G_1, G_2)$
initialize $OPEN$
while $OPEN \neq \emptyset$ **do**
    $p = OPEN.popFirst()$
    if p is a leaf (i.e. if all vertices of V_1 are mapped) **then**
        complete p by inserting pending vertices of V_2
        update $(\gamma_{UB}, UBCOST)$ if required
    else
        Stack into OPEN all sons q of p such that $g(q) + h(q) < UBCOST$.
    end if
end while

Depth first search algorithm

Discussion

- 🙂The number of pending edit paths in OPEN is bounded by $|V_1|.|V_2|$,
- 🙂The initialization by an heuristic allows to discard many branches,
- This algorithm quickly find a first edit path. It may be tuned into [Abu-Aisheh, 2016]:
  - An any time algorithm,
  - a Parallel or distributed algorithm.
- ☹ The computation of the optimal value of the Graph Edit distance may anyway require long processing times.

Independent Edit paths

An element $\gamma \in \Gamma(G_1, G_2)$ is potentially infinite by just doing and undoing a given operation (e.g. insert and then delete a node). All cost being positive such an edit path can not correspond to a minimum:

Definition (Independent Edit path)

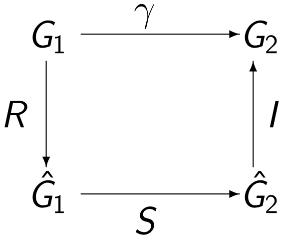
An independent edit path between two labeled graphs G_1 and G_2 is an edit path such that:

1. No node nor edge is both substituted and removed,
2. No node nor edge is simultaneously substituted and inserted,
3. Any inserted element is never removed,
4. Any node or edge is substituted at most once,

Decomposition of an Edit Path

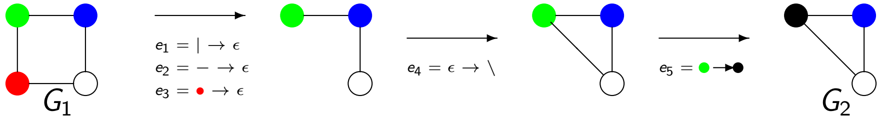
Proposition

The elementary operations of an independent edit path between two graphs G_1 and G_2 may be ordered into a sequence of removals, followed by a sequence of substitutions and terminated by a sequence of insertions.

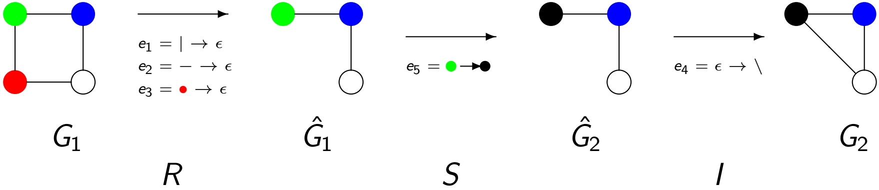


Note that $\hat{G}_1 \underset{s}{\cong} \hat{G}_2$

Decomposition of an Edit Path



May be reordered into:



GED Formulation based on $\hat{G}_1$

Let $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$. We have:

$$\begin{aligned} d(G_1, G_2) &= \sum_{v \in V_1 \setminus \hat{V}_1} c_{vd}(v) + \sum_{e \in E_1 \setminus \hat{E}_1} c_{ed}(e) + \sum_{v \in \hat{V}_1} c_{vs}(v) + \sum_{e \in \hat{E}_1} c_{es}(e) \\ &+ \sum_{v \in V_2 \setminus \hat{V}_2} c_{vi}(v) + \sum_{e \in E_2 \setminus \hat{E}_2} c_{ei}(e) \end{aligned}$$

If all costs are constants we have:

$$\begin{aligned} d(G_1, G_2) &= (|V_1| - |\hat{V}_1|) c_{vd} + (|E_1| - |\hat{E}_1|) c_{ed} + V_f c_{vs} + E_f c_{es} \\ &+ (|V_2| - |\hat{V}_2|) c_{vi} + (|E_2| - |\hat{E}_2|) c_{ei} \end{aligned}$$

where V_f (resp. E_f) denotes the number of vertices (resp. edges) substituted with a non zero cost and c_{vd},

GED Formulation based on $\hat{G}_1$

By grouping constant terms minimize:

$$\begin{aligned} d(G_1, G_2) &= (|V_1| - |\hat{V}_1|)c_{vd} + (|E_1| - |\hat{E}_1|)c_{ed} + V_f c_{vs} + E_f c_{es} \\ &\quad + (|V_2| - |\hat{V}_2|)c_{vi} + (|E_2| - |\hat{E}_2|)c_{ei} \end{aligned}$$

is equivalent to maximize:

$$M(P) \overset{not.}{=} |\hat{V}_1|(c_{vd} + c_{vi}) + |\hat{E}_1|(c_{ed} + c_{ei}) - V_f c_{vs} - E_f c_{es}$$

We should thus maximize $\hat{G}_1$ while minimizing V_f and E_f.

If $c(u \to v) \geq c(u \to \epsilon) + c(\epsilon \to v)$, $M(P)$ is maximum for $V_f = E_f = \emptyset$ and $\hat{G}_1$ is a maximum common sub graph of G_1 and G_2[Bunke, 1997, Bunke and Kandel, 2000].

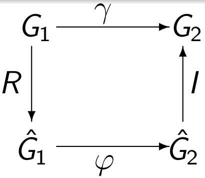
Further restriction

Definition (Restricted edit path)

A restricted edit path is an independent edit path in which an edge cannot be removed and then inserted.

Proposition

If G_1 and G_2 are simple graphs, there is a one-to-one mapping between the set of restricted edit paths between G_1 and G_2 and the set of injective functions from a subset of V_1 to V_2 . We denote by φ_0 , the special function from the empty set onto the empty set.



ϵ-assignments

- Let v_1 and V_2 be two sets, with $n = |v_1|$ and $m = |V_2|$.
- Consider $V_1^\epsilon = V_1 \cup \{\epsilon\}$ and $V_2^\epsilon = V_2 \cup \{\epsilon\}$.

Definition

An ϵ-assignment from V_1 to V_2 is a mapping $\varphi : V_1^\epsilon \to \mathcal{P}(V_2^\epsilon)$, satisfying the following constraints:

$$\begin{array}{ll} \forall i \in V_1 & |\varphi(i)| = 1 \\ \forall j \in V_2 & |\varphi^{-1}(j)| = 1 \\ \epsilon \in \varphi(\epsilon) & \end{array}$$

An ϵ assignment encodes thus:

1. Substitutions: $\varphi(i) = j$ with $(i, j) \in V_1 \times V_2$.
2. Removals: $\varphi(i) = \epsilon$ with $i \in V_1$.
3. Insertions: $j \in \varphi^{-1}(\epsilon)$ with $j \in V_2$.

From assignments to matrices

- An $\epsilon-$assignment can be encoded in matrix form: $\mathbf{X} = (x_{i,k})_{(i,k)\in\{1,\dots,n+1\}\times\{1,\dots,m+1\}}$ with

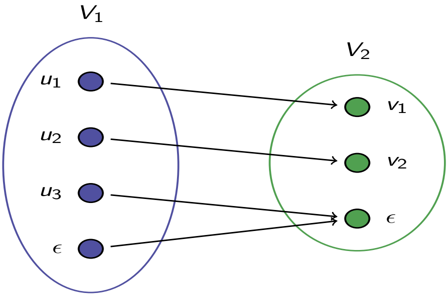
$$\forall (i,k) \in \{1,\dots,n+1\}\times\{1,\dots,m+1\} \quad x_{i,k} = \begin{cases} 1 & \text{if } k \in \varphi(i) \\ 0 & \text{else} \end{cases}$$

- We have:

$$\begin{cases} \forall i = 1,\dots,n, & \sum_{k=1}^{m+1} x_{i,k} = 1 & (|\varphi(i)| = 1) \\ \forall k = 1,\dots,m, & \sum_{j=1}^{n+1} x_{j,k} = 1 & (|\varphi^{-1}(k)| = 1) \\ & x_{n+1,m+1} = 1 & (\epsilon \in \varphi(\epsilon)) \\ \forall (i,j) & x_{i,j} \in \{0,1\} & \end{cases}$$

From functions to matrices

Example



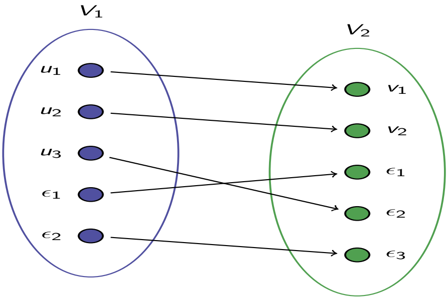
$$\mathbf{x} = \begin{array}{c} \\ u_1 \\ u_2 \\ u_3 \\ \\ \epsilon \end{array} \begin{array}{c} \begin{array}{cc|c} v_1 & v_2 & \epsilon \end{array} \\ \left(\begin{array}{cc|c} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ \hline 0 & 0 & 1 \end{array} \right) \end{array}$$

- The set of permutation matrices encoding $\epsilon-$assignments is called the set of ϵ-assignment matrices denoted by $\mathcal{A}_{n,m}$.

An alternative encoding

Example

- Usual assignments are encoded by larger $(n+m) \times (n+m)$ matrices[Riesen, 2015].



$$\mathbf{x} = \begin{array}{c} \\ u_1 \\ u_2 \\ u_3 \\ \\ \epsilon_1 \\ \epsilon_2 \end{array} \begin{array}{c} \begin{array}{ccccc} v_1 & v_2 & \epsilon_1 & \epsilon_2 & \epsilon_3 \end{array} \\ \left(\begin{array}{cc|ccc} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ \hline 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{array}\right) \end{array}$$

Back to edit paths

- Let us consider two simple graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ with $|V_1| = n$ and $|V_2| = m$.

Proposition

There is a one-to-one relation between the set of restricted edit paths from G_1 to G_2 and $\mathcal{A}_{n,m}$.

Theorem

Any non-infinite value of $\frac{1}{2}\mathbf{x}^T\Delta\mathbf{x} + \mathbf{c}^T\mathbf{x}$ corresponds to the cost of a restricted edit path. Conversely the cost of any restricted edit path may be written as $\frac{1}{2}\mathbf{x}^T\Delta\mathbf{x} + \mathbf{c}^T\mathbf{x}$ with the appropriate $\mathbf{x}$.

Costs of Node assignments

$$\mathbf{C} = \begin{pmatrix} c(u_1 \to v_1) & \cdots & c(u_1 \to v_m) & c(u_1 \to \varepsilon) \\ & \ddots & & \vdots \\ \vdots & c(u_i \to v_j) & \vdots & c(u_i \to \varepsilon) \\ & & \ddots & \vdots \\ c(u_n \to v_1) & \ldots & c(u_n \to v_m) & c(u_n \to \varepsilon) \\ c(\varepsilon \to v_1) & c(\epsilon \to v_i) & c(\epsilon \to v_m) & 0 \end{pmatrix}$$

- $c = vect(\mathbf{C})$
- Alternative factorizations of cost matrices exist[Serratosa, 2014, Serratosa, 2015].

Cost of edges assignments

- Let us consider a $(n+1)(m+1) \times (n+1)(m+1)$ matrix D such that:

$$d_{ik,jl} = c_e(i \to k, j \to l)$$

with:

(i,j)	(k,l)	edit operation	cost $c_e(i \to k, j \to l)$
$\in E_1$	$\in E_2$	substitution of (i,j) by (k,l)	$c((i,j) \to (k,l))$
$\in E_1$	$\notin E_2$	removal of (i,j)	$c((i,j) \to \epsilon)$
$\notin E_1$	$\in E_2$	insertion of (k,l) into E_1	$c(\epsilon \to (k,l))$
$\notin E_1$	$\notin E_2$	do nothing	0

- $\Delta = \mathbf{D}$ if both G_1 and G_2 are undirected and
- $\Delta = \mathbf{D} + \mathbf{D}^T$ else.
- Matrix Δ is symmetric in both cases.

Let us approximate

- One solution to solve this quadratic problem consists in dropping the quadratic term. We hence get:

$$d(G_1, G_2) \approx \min \left\{ \mathbf{c}^T \mathbf{x} \mid \mathbf{x} \in \text{vec}[\mathcal{A}^{\sim}_{n,m}] \right\} = \min \sum_{i=1}^{n+m} \sum_{j=1}^{n+m} c_{i,j} x_{i,j}$$

- This problem is an instance of a bipartite graph matching problem also called linear sum assignment problem.

Bipartite Graph matching

- Approximations of the Graph edit distance defined using bipartite Graph matching methods are called bipartite Graph edit distances (BP-GED).
- Different methods, such as Munkres or Jonker-Volgerand[Burkard et al., 2012] allow to solve this problem in cubic time complexity.
- The general procedure is as follows:
  1. compute:
  $$x = \arg\min c^T x$$
  2. Deduce the edge operations from the node operations encoded by x.
  3. Return the cost of the edit path encoded by x.

Matching Neighborhoods

- The matrix **C** defined previoulsy only encodes node information.
- Idea: Inject edge information into matrix C. Let

$$d_{i,j} = min_x \sum_{k=1}^{n+m} c(ik \to jl) x_{k,l}$$

  the cost of the optimal mapping of the edges incident to i onto the edge incident to j.

- Let :

$$c^*_{i,j} = c(u_i \to v_j) + d_{i,j} \text{ and } x = \arg\min (c^*)^T x$$

- The edit path deduced from x defines an edit cost $d_{ub}(G_1, G_2)$ between G_1 and G_2.

Matching Neighborhoods

- We may alternatively consider:

$$d_{lb}(G_1, G_2) = \sum_{i=1}^{n+m} \sum_{j=1}^{n+m} \left(c(u_i \to v_j) + \frac{1}{2} d_{i,j} \right) x_{i,j}$$

- The resulting distances satisfy[Riesen, 2015]:

$$d_{lb}(G_1, G_2) \leq d(G_1, G_2) \leq d_{ub}(G_1, G_2)$$

Matching larger structure

- The upper bound provided by $d_{ub}(G_1, G_2)$ may be large, especially for large graphs. So a basic idea consists in enlarging the considered substructures:
  1. Consider both the incident edges and the adjacent nodes.
  2. Associate a bag of walks to each vertex and define **C** as the cost of matching these bags [Gaüzère et al., 2014].
  3. Associate to each vertex a subgraph centered around it and define **C** as the optimal edit distance between these subgraphs [Carletti et al., 2015].
  4. ...
- All these heuristics provide an upper bound for the Graph edit distance.
- But: up to a certain point the linear approximation of a quadratic problem reach its limits.

Improving the initial edit path

- One idea to improve the results of the linear approximation of the GED consists in applying a post processing stage.
- Two ideas have been proposed:
  1. By modifying the initial cost matrix and recomputing a new assignment.
  2. By swapping elements in the assignment returned by the linear algorithm.

BP-Iterative

- q consecutive trials to improve the initial guess provided by a bipartite algorithm.

```
1: Build a Cost matrix C*
2: Compute x = arg min(c*)^T x
3: Compute d_best = d_x(G_1, G_2)
4: for i=1 to q do
5:     determine node operation u_i → v_j with highest cost
6:     set c*_{i,j} = +∞                    ▷ prevent assignment u_i → v_j
7:     compute a new edit path γ on modified matrix C*.
8:     if d_γ(G_1, G_2) < d_best then
9:         d_best = d_γ(G_1, G_2)
10:    end if
11: end for
```

Alternative strategies

- BP-Iterative never reconsiders an assignment.
- Riesen[Riesen, 2015] proposed an alternative procedure called $BP - Float$ which after each new forbidden assignment:
  1. reconsider the entries previously set to $+\infty$,
  2. Restore them if the associated edit cost decreases
- An alternative search procedure is based on Genetic algorithms[Riesen, 2015]:
  1. Build a population by randomly forbidding some entries of $\mathbf{C}^*$,
  2. Select a given percentation of the population whose cost matrix is associated to the lowest edit distance,
  3. Cross the population by forbidding an entry if this entry is forbidden by one of the two parent.
  4. Go back to step 2.

Swapping assignments

- Forbidding entries of matrix $\mathbf{C}^*$ requires to recompute a new assignment from scrach.
- The basic idea of the swapping strategy consits to replace assignments:

$$\left.\begin{array}{ccc} u_i & \rightarrow & v_k \\ u_j & \rightarrow & v_l \end{array}\right) \text{ by } \left(\begin{array}{ccc} u_i & \rightarrow & v_l \\ u_j & \rightarrow & v_k \end{array}\right.$$

- Or in matrix terms, replacing

$$\left.\begin{array}{ccccc} x_{i,k} & = & x_{j,l} & = & 1; \\ x_{i,l} & = & x_{j,k} & = & 0 \end{array}\right) \text{ by } \left(\begin{array}{ccccc} x_{i,l} & = & x_{j,k} & = & 1; \\ x_{i,k} & = & x_{j,l} & = & 0. \end{array}\right.$$

BP-Greedy-Swap

swapped=true
while swapped **do**
  Search for indices (i, j, k, l) such that $x_{i,k} = x_{j,l} = 1$
  $cost_{orig} = c^*_{i,k} + c^*_{j,l}$
  $cost_{swap} = c^*_{i,l} + c^*_{j,k}$
  if $|cost_{orig} - cost_{swap}| < \theta cost_{orig}$ **then**
    $x' = swap(x)$
    if $d_{x'}(G_1, G_2) < d_{best}$ **then**
      $d_{best} = d_{x'}(G_1, G_2)$
      best swap=x'; swapped=true
    end if
  end if
  if we swapped for at least one $(i, j) \in V_1^2$ **then** update x according to best swap.
end while

Discussion

- $BP - Greedy - Swap$ as $BP - Iterative$ never reconsiders a swap,
- An alternative exploration of the space of solutions may be performed using Genetic algorithms[Riesen, 2015].
- Or by a tree search:
  - This procedure is called **Beam search**
  - Each node of the tree is defined by $(x, q, d_x(G_1, G_2))$ where x is an assignment and q the depth of the node.
  - Nodes are sorted according to:
    1. their depth,
    2. their cost.

BP-Beam

Build a Cost matrix $\mathbf{C}^*$
Compute $x = \arg\min (c^*)^T x$ and $d_x(G_1, G_2)$
$OPEN = \{(x, 0, d_x(G_1, G_2))\}$
while $OPEN \neq \emptyset$ **do**
  remove first tree node in OPEN: $(x, q, d_x(G_1, G_2))$
  for j=q+1 to n+m **do**
    x'=swap(x,q+1,j)
    $OPEN = OPEN \cup \{(x', q+1, d_{x'}(G_1, G_2))\}$
    if $d_{x'}(G_1, G_2) < d_{best}$ **then**
      $d_{best} = d_{x'}(G_1, G_2)$
    end if
  end for
  while size of $OPEN > b$ **do**
    Remove tree nodes with highest value d_x from $OPEN$
  end while
end while

From linear to quadratic optimization

- Improvements of Bipartite matching explore the space of edit paths around an initial solution.
- In order to go beyond these results, this search must be guided by considering the real quadratic problem.

$$d(G_1, G_2) = \frac{1}{2}\mathbf{x}^{\mathbf{T}}\mathbf{\Delta x} + c^T x$$

Justice and Hero

This work[Justice and Hero, 2006]:

- Proposes a quadratic formulation of the Graph Edit distance,
- Designed for Graphs with unlabeled edges by augmenting graphs with ϵ nodes,
- Solved by relaxing the integer constraint $x_{i,j} \in \{0, 1\}$ to $x_{i,j} \in [0, 1]$ and then solving it using interior point method.
- Propose a linear approximation corresponding to bipartite matching.

Neuhauss and Bunke

In this work[Neuhaus and Bunke, 2007]

- Also a quadratic formulation but with node operations restricted to substitution with the induced operations on edges.
- Works well mainly for graphs having close sizes with a cost of substitution lower than a cost of removal plus a cost of insertion.

From quadratic to linear problems

- Let us consider a quadratic problem:

$$x^T Q x = \sum_{i=1}^{n} \sum_{j=1}^{n} q_{i,j} x_i x_j$$

- Let us introduce $y_{n*i+j} = x_i x_j$ we get:

$$x^T Q x = \sum_{k=1}^{n^2} q_k y_k$$

  with an appropriate renumbering of Q's elements. Hence a Linear Sum Assignment Problem with additional constraints.
- Note that the Hungarian algorithm can not be applied due to additional constraints. We come back to tree based algorithms.
- This approach has been applied to the computation of the exact Graph Edit Distance by[Lerouge et al., 2016]

Integer Projected Fixed Point (IPFP)

[Leordeanu et al., 2009]

- Start with an good Guess x_0:

  $x = x_0$
  while a fixed point is not reached **do**
  $\quad b^* = \arg\min\{x^T\Delta + c^T)b, b \in \{0,1\}^{(n+m)^2}\}$
  $\quad t^* =$ line search between x and b^*
  $\quad x = x + t^*(b^* - x)$
  end while

- b^* minimizes a sum of:

$$(x^T\Delta + c^T)_{i,j} = c_{i,j} + \sum_{k,l}^{n+m} d_{i,j,k,l} x_{k,l}$$

  which may be understood as the cost of mapping i to j given the previous assignment x.

- The distance decreases at each iteration. Moreover,
- at each iteration we come back to an integer solution,

Graduated NonConvexity and Concativity Procedure (GNCCP)

- Consider [Liu and Qiao, 2014]:

$$S_\eta(x) = (1 - |\zeta|)S(x) + \zeta x^T x \text{ with } S(x) = \frac{1}{2}x^T \Delta x + c^T x$$

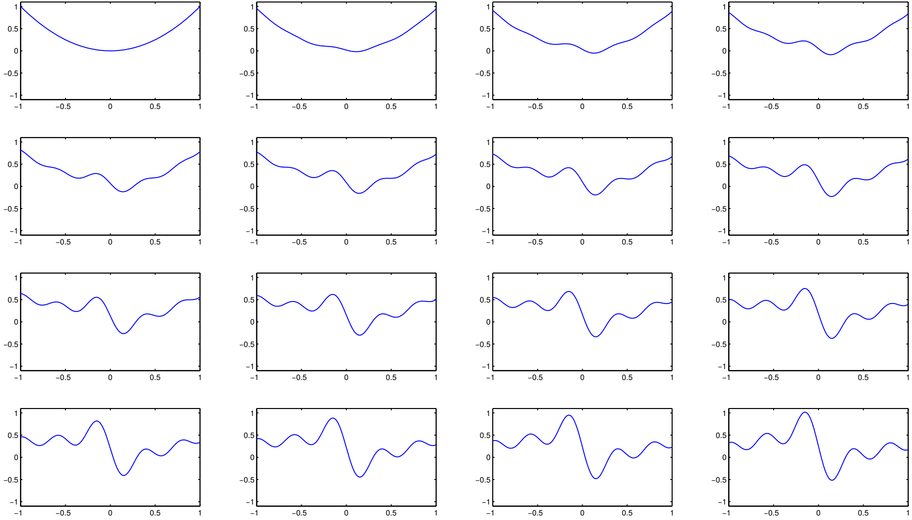
where $\zeta \in [-1, 1]$.

$$\begin{cases} \zeta = 1 : \text{ Convex objective function} \\ \zeta = -1 : \text{ Concave objective function} \end{cases}$$

- The algorithm tracks the optimal solution from a convex to a concave relaxation of the problem.

From $\zeta = 1$ to $\zeta = 0$

$\zeta = 1$



$\zeta = 0$

GNCCP Algorithm

$\zeta = 1, d = 0.1, x = 0$
while $(\zeta > -1)$ and $(x \notin \mathcal{A}_{n,m})$ **do**
  $Q = \frac{1}{2}(1 - |\zeta|)\Delta + \zeta I$
  $L = (1 - |\zeta|)c$
  $x = IPFP(x, Q, L)$
  $\zeta = \zeta - d$
end while

Datasets

Dataset	*Number of graphs*	*Avg Size*	*Avg Degree*	*Properties*
Alkane	150	8.9	1.8	acyclic, unlabeled
Acyclic	183	8.2	1.8	acyclic
MAO	68	18.4	2.1	
PAH	94	20.7	2.4	unlabeled, cycles
MUTAG	8×10	40	2	

Experiments

Algorithm	Alkane		Acyclic	
	d	t	d	t
A^*	15.5	1.29	17.33	6.02
[Riesen and Bunke, 2009]	35.2	0.0013	35.4	0.0011
[Gaüzère et al., 2014]	34.5	0.0020	32.6	0.0018
[Carletti et al., 2015]	26	2.27	28	0.73
$\text{IPFP}_{\text{Random init}}$	22.6	0.007	23.4	0.006
$\text{IPFP}_{\text{Init [Riesen and Bunke, 2009]}}$	22.4	0.007	22.6	0.006
$\text{IPFP}_{\text{Init [Gaüzère et al., 2014]}}$	19.3	0.005	20.4	0.004
[Neuhaus and Bunke, 2007]	20.5	0.07	25.7	0.42
GNCCP	16.8	0.12	19.1	0.07

Experiments

Algorithm	MAO		PAH	
	d	t	d	t
[Riesen and Bunke, 2009]	105	$5\ 10^{-3}$	138	$7\ 10^{-3}$
[Gaüzère et al., 2014]	56.9	$2\ 10^{-2}$	123.8	$3\ 10^{-2}$
[Carletti et al., 2015]	44	6.16	129	2.01
$\text{IPFP}_{\text{Random init}}$	65.2	0.031	63	0.04
$\text{IPFP}_{\text{Init [Riesen and Bunke, 2009]}}$	59	0.031	62.2	0.04
$\text{IPFP}_{\text{Init [Gaüzère et al., 2014]}}$	32.9	0.030	48.9	0.048
[Neuhaus and Bunke, 2007]	59.1	7	52.9	8.20
GNCCP	32.9	0.46	38.7	0.86

- Two characteristics of a Graph edit distance algorithm:
  - Mean Deviation:

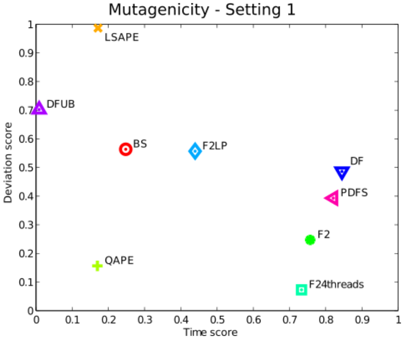
$$\overline{deviation\_score^m} = \frac{1}{\#subsets} \sum_{\mathcal{S} \in subsets} \frac{\overline{dev^m_{\mathcal{S}}}}{max\_dev_{\mathcal{S}}}$$

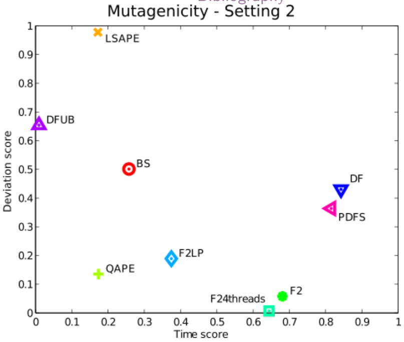
  - Mean execution time:

$$\overline{time\_score^m} = \frac{1}{\#subsets} \sum_{\mathcal{S} \in subsets} \frac{\overline{time^m_{\mathcal{S}}}}{max\_time_{\mathcal{S}}}$$

- Several Algorithms (all limited to 30 seconds):
  - Algorithms based on a linear transformation of the quadratic problem solved by integer programming:
    - F2 (●),
    - F24threads(□),
    - F2LP ((◊, relaxed problem)
  - Algorithms based on Depth first search methods:
    - DF(▽),
    - PDFS(◁),
    - DFUP(△).
  - Beam Search: BS (⊙)
  - *IPFP*[Gaüzère et al., 2014] : QAPE (+)
  - Bipartite matching[Gaüzère et al., 2014]: LSAPE(×)

Average speed-deviation scores on MUTA subsets

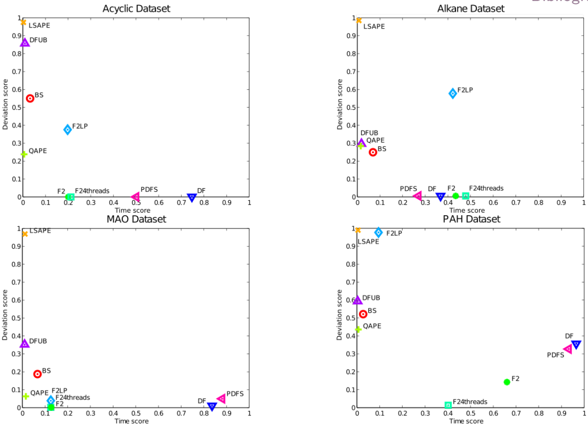




- Two settings of editing costs

	vertices			edges		
	c_s	c_d	c_i	c_s	c_d	c_i
Setting 1	2	4	4	1	2	2
Setting 2	6	2	2	3	1	1

Average speed-deviation scores on GREYC's subsets



- Setting:

	vertices			edges		
	c_s	c_d	c_i	c_s	c_d	c_i
Setting	2	4	4	1	1	1

Conclusion

- Bipartite Graph edit distance has re initiated a strong interest on Graph edit Distance from the research community
  1. It is fast enough to process large graph databases,
  2. It provides a reasonable approximation of the GED.
- More recently new quadratic solvers for the GED have emerged.
  1. They remain fast (while slower than BP-GED),
  2. They strongly improve the approximation or provide exact solutions.

Future Work

- The Bipartite Matching remains a core element of several quadratic algorithms. So any improvement of such algorithms has many consequences.
- Quadratic algorithms for GED are still immature, a lot of job remains to be done.
- Distances not directly related to GED should also be investigated (Kernel Based, Hausdorff,. . .)
- Related applications are also quite fascinating:
  - Media/mean computation,
  - Learning costs for GED,
  - . . .

Bibliography

Abu-Aisheh, Z. (2016). *Anytime and distributed Approaches for Graph Matching*. PhD thesis, Université François Rabelais, Tours.

Ambauen, R., Fischer, S., and Bunke, H. (2003). Graph edit distance with node splitting and merging, and its application to diatom identification. In *Graph Based Representations in Pattern Recognition: 4th IAPR International Workshop, GbRPR 2003 York, UK, June 30 – July 2, 2003 Proceedings*, pages 95–106, Berlin, Heidelberg. Springer Berlin Heidelberg.

Bunke, H. (1997). On a relation between graph edit distance and maximum common subgraph. *Pattern Recogn. Lett.*, 18(9):689–694.

Bunke, H. and Kandel, A. (2000). Mean and maximum common subgraph of two graphs. *Pattern Recogn. Lett.*, 21(2):163–168.

Burkard, R., Dell'Amico, M., and Martello, S. (2012). *Assignment Problems: Revised Reprint*. Society for Industrial and Applied Mathematics.

Carletti, V., Gaüzère, B., Brun, L., and Vento, M. (2015). *Graph-Based Representations in Pattern Recognition: 10th IAPR-TC-15 International Workshop, GbRPR 2015, Beijing, China, May 13-15, 2015. Proceedings*, chapter Approximate Graph Edit Distance Computation Combining Bipartite

Matching and Exact Neighborhood Substructure Distance, pages 188–197. Springer International Publishing, Cham.

Gaüzère, B., Bougleux, S., Riesen, K., and Brun, L. (2014). *Structural, Syntactic, and Statistical Pattern Recognition: Joint IAPR International Workshop, S+SSPR 2014, Joensuu, Finland, August 20-22, 2014. Proceedings*, chapter Approximate Graph Edit Distance Guided by Bipartite Matching of Bags of Walks, pages 73–82. Springer Berlin Heidelberg, Berlin, Heidelberg.

Justice, D. and Hero, A. (2006). A binary linear programming formulation of the graph edit distance. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(8):1200–1214.

Leordeanu, M., Hebert, M., and Sukthankar, R. (2009). An integer projected fixed point method for graph matching and MAP inference. In *Advances in Neural Information Processing Systems 22: 23rd Annual Conference on Neural Information Processing Systems 2009. Proceedings of a meeting held 7-10 December 2009, Vancouver, British Columbia, Canada.*, pages 1114–1122.

Lerouge, J., Abu-Aisheh, Z., Raveaux, R., Héroux, P., and Adam, S. (2016). Exact graph edit distance computation using a binary linear program. In Robles-Kelly, A., Loog, M., Biggio, B., Escolano, F., and Wilson, R., editors, *Structural, Syntactic, and Statistical Pattern Recognition: Joint IAPR*

International Workshop, S+SSPR 2016, Mérida, Mexico, November 29 – December 2, 2016, Proceedings, pages 485–495, Cham. Springer International Publishing.

Liu, Z. and Qiao, H. (2014). GNCCP - graduated nonconvexityand concavity procedure. *IEEE Trans. Pattern Anal. Mach. Intell.*, 36(6):1258–1267.

Neuhaus, M. and Bunke, H. (2007). A quadratic programming approach to the graph edit distance problem. In Escolano, F. and Vento, M., editors, *Graph-Based Representations in Pattern Recognition: 6th IAPR-TC-15 International Workshop, GbRPR 2007, Alicante, Spain, June 11-13, 2007. Proceedings*, pages 92–102, Berlin, Heidelberg. Springer Berlin Heidelberg.

Riesen, K. (2015). *Structural Pattern Recognition with Graph Edit Distance*. Springer.

Riesen, K. and Bunke, H. (2009). Approximate graph edit distance computation by means of bipartite graph matching. *Image and Vision Computing*, 27(7):950 – 959. 7th IAPR-TC15 Workshop on Graph-based Representations (GbR 2007).

Serratosa, F. (2014). Fast computation of bipartite graph matching. *Pattern Recognition Letters*, 45:244–250.

Serratosa, F. (2015). Speeding up fast bipartite graph matching through a new cost matrix. *Int. Journal of Pattern Recognition*, 29(2).