Graph Edit Distance: Basics and History

Luc Brun
Special Thanks to:
Benoit Gaüzère,
Sébastien Bougleux,
Evariste Daller,
Nicolas Boria.

12 Octobre 2018

Recognition implies the definition of similarity or dissimilarity measures.

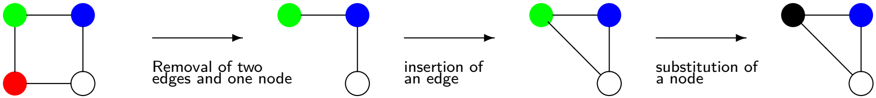
Different measures between graphs:

- Distance based on maximum common subgraphs,
- Distance based on spectral characteristics,
- Graph kernels (simmilarities),
  - 🙂 Definite positive,
  - 🙁 Not so fine.
- Graph Edit distance.
  - 🙁 Not definite negative,
  - 🙂 Very fine.

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.



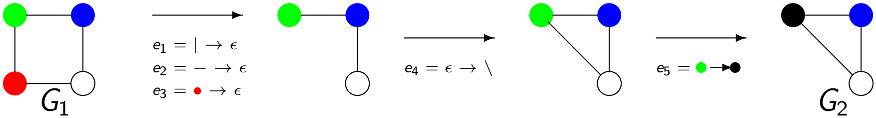
A substitution is denoted $u \to v$, an insertion $\epsilon \to v$ and a removal $u \to \epsilon$.

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

Costs

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: $c(l \to l) = 0$.



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

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.

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

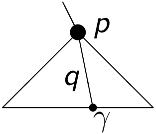
- $\mathcal{NP}$-hard.
- Suggests an exploration of $\Gamma(G_1, G_2)$,
  - Tree based algorithms.

Tree search algorithms

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.
- 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.

Depth first search algorithm

[Abu-Aisheh, 2016, Abu-Aisheh et al., 2018]

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

$(\gamma_{UB}, UBCOST) = GoodHeuristic(G_1, G_2)$
initialize $OPEN = \{u_1 \to \epsilon\} \cup \bigcup_{w \in V_2} \{u_1 \to w\}$
while $OPEN \neq \emptyset$ **do**
    $p = OPEN.popFirst()$
    if p is a leaf **then**
        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

- Bounded number of edit paths

Depth first search algorithm

[Abu-Aisheh, 2016, Abu-Aisheh et al., 2018]

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 $OPEN = \{u_1 \to \epsilon\} \cup \bigcup_{w \in V_2} \{u_1 \to w\}$
while $OPEN \neq \emptyset$ **do**
    $p = OPEN.popFirst()$
    if p is a leaf **then**
        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

- Bounded number of edit paths
- Anytime, Parallel.

Depth first search algorithm [Abu-Aisheh, 2016, Abu-Aisheh et al., 2018]

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 $OPEN = \{u_1 \to \epsilon\} \cup \bigcup_{w \in V_2} \{u_1 \to w\}$
while $OPEN \neq \emptyset$ **do**
    $p = OPEN.popFirst()$
    if p is a leaf **then**
        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

- Bounded number of edit paths
- Anytime, Parallel.
- Compution of the global optimum may be long

Restricted 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 (Restricted 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,
5. Any edge cannot be removed and then inserted.

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

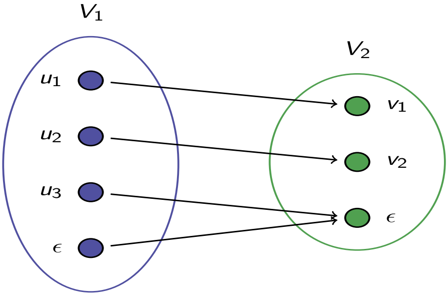
- An $\epsilon-$assignment can be encoded in matrix form: $\mathbf{X} = (x_{i,k})_{(i,k)\in\{1,\ldots,n+1\}\times\{1,\ldots,m+1\}}$ with

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

- We have:

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

Example



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

- The set of permutation matrices encoding $\epsilon-$assignments is called the set of ϵ-assignment matrices denoted by $\mathcal{A}_{n,m}$.
- Usual assignments are encoded by larger $(n+m) \times (n+m)$ matrices[Riesen, 2015].

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

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

- $c = vect(\mathbf{C})$
- 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

$$GED(G_1, G_2) = \min_{x \in vect(\mathcal{A}_{n,m})} \frac{1}{2} x^T \boldsymbol{\Delta} x + c^T x$$

- Matrix Δ is non convex with several local minima. The problem is thus $\mathcal{NP}$-hard.
- One solution to solve this quadratic problem consists in dropping the quadratic term. We hence get:

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

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

Bipartite Graph matching

- Such an approximation of the GED is called a bipartite Graph edit distances (BP-GED).
- Munkres or Jonker-Volgenant[Burkard et al., 2012] allow to solve this problem in cubic time complexity.

- The general procedure is as follows:
  1. compute:
  $$x = \underset{x \in vec(\mathcal{A}_{n,m})}{\arg\min} \; c^T x$$
  2. Return the cost of the edit path γ_x encoded by x.

Matching Neighborhoods

- The matrix **C** defined previoulsy only encodes node information.
- Idea: Inject edge information into matrix C[Riesen and Bunke, 2009]. Let

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

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

- Let :

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

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

Matching 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. Incident edges and adjacent nodes.
  2. Bags of walks [Gaüzère et al., 2014].
  3. Centered subgraphs [Carletti et al., 2015].
  4. Concentric rings [Blumenthal et al., 2018],
  5. ...
- 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.

From linear to quadratic optimization

- 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 [Riesen, 2015]:
  1. By modifying the initial cost matrix and recomputing a new assignment.
  2. By swapping elements in the assignment returned by the linear algorithm.
- In order to go beyond these results, the search must be guided by considering the real quadratic problem.

$$GED(G_1, G_2) = \min_{x \in vect(\mathcal{A}_{n,m})} \frac{1}{2} x^T \mathbf{\Delta} x + c^T x$$

Previous Quadratic methods

- [Justice and Hero, 2006]
- [Neuhaus and Bunke, 2007]
- ...

From quadratic to linear problems

- Let us consider a quadratic problem:

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

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

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

  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]

Frank-Wolfe

[Frank and Wolfe, 1956, Leordeanu et al., 2009]

- Start with an good Guess x_0:

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

- b^* minimizes a sum of:

$$(x^T\Delta + c^T)_{i,k} = c_{i,k} + \sum_{j}^{n+1}\sum_{l}^{m+1} d_{i,k,j,l}x_{j,l}$$

  which may be understood as the cost of mapping i to j given the previous assignment x.
- The edit cost decreases at each iteration.
- 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.

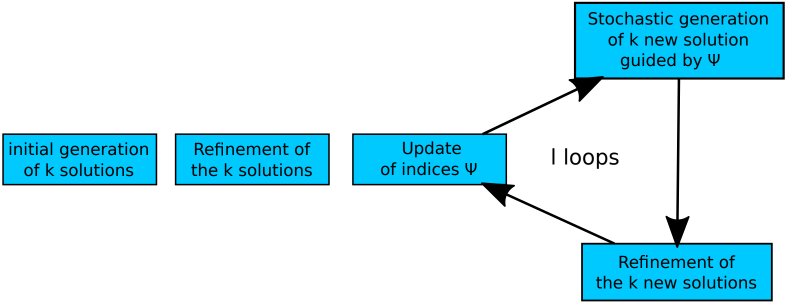
GNCCP Algorithm

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

Exploration of $\mathcal{A}_{n,m}$

[Boria et al., 2018]

- The problem is non convex: Good starting points induce good solutions.
- Proposition [Boria et al., 2018] : Explore randomly the space of solutions while privileging " good " assignments.



Experiments

Datasets

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

Experiments

Algorithm	Alkane		Acyclic	
	d	t	d	t
A^*	15.3	1.29	16.7	6.02
[Riesen and Bunke, 2009]	37.8	10^{-4}	33.3	10^{-4}
[Gaüzère et al., 2014]	36.0	0.02	31.8	0.02
$\mathrm{FW}_{\text{Random init}}$	19.9	0.02	22.2	0.01
$\mathrm{mFW}_{40\ \text{Random Init}}$	15.3	0.1	16.74	0.07
[Neuhaus and Bunke, 2007]	20.5	0.07	25.7	0.042
GNCCP	16.6	0.12	18.7	0.32

Experiments

Algorithm	MAO		PAH	
	d	t	d	t
[Riesen and Bunke, 2009]	95.7	10^{-3}	135.2	10^{-3}
[Gaüzère et al., 2014]	85.1	1.48	125.8	2.6
$\text{FW}_{\text{Init [Riesen and Bunke, 2009]}}$	38.4	0.04	50.0	0.09
$\text{FW}_{\text{Init [Gaüzère et al., 2014]}}$	38.7	1.53	46.8	2.68
$\text{mFW}_{\text{Init [Riesen and Bunke, 2009]}}$	31.4	0.4	31.5	0.7
$\text{mFW}_{\text{Init [Gaüzère et al., 2014]}}$	32.4	1.75	29.8	2.96
[Neuhaus and Bunke, 2007]	59.1	7	52.9	8.20
GNCCP	34.3	9.23	34.2	14.44

Graph Edit distance Contest

[Abu-Aisheh et al., 2017]

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

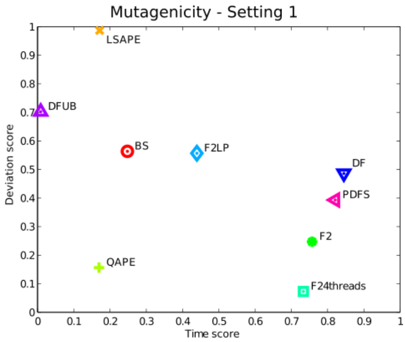
$$\overline{deviation\_score^m} = \frac{1}{\#subsets} \sum_{\mathcal{S} \in subsets} \frac{\overline{dev^m_{\mathcal{S}}}}{max\_dev_{\mathcal{S}}}$$

  - 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(◁),
    - DFUB(△).
  - Beam Search: BS (⊙)
  - $FW_{\text{[Gaüzère et al., 2014]}}$: QAPE (+)
  - Bipartite matching[Gaüzère et al., 2014]: LSAPE(×)

Average speed-deviation scores on MUTA subsets



- Setting 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

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 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

- 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:
  - Median/mean computation,
  - Learning costs for GED,
  - ...

Questions

Bibliography

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

Abu-Aisheh, Z., Gauzere, B., Bougleux, S., Ramel, J.-Y., Brun, L., Raveaux, R., H'eroux, P., and Adam, S. (2017). Graph edit distance contest: Results and future challenges. *Pattern Recognition Letters*, 100(Supplement C):96 – 103.

Abu-Aisheh, Z., Raveaux, R., Ramel, J.-Y., and Martineau, P. (2018). A parallel graph edit distance algorithm. *Expert Systems with Applications*, 94:41 – 57.

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.

Blumenthal, D., Bougleux, S., Gamper, J., and Brun, L. (2018). Ring based approximation of graph edit distance. In *Proceedins of SSPR'2018*, Beijing. IAPR, Springer.

Boria, N., Bougleux, S., and Brun, L. (2018). Approximating ged using a stochastic generator and multistart IPFP. In *Proceedings of SSPR*. IAPR, Springer.

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.

Frank, M. and Wolfe, P. (1956). An algorithm for quadratic programming. *Naval Research Logistics Quarterly*, 3(1-2):95–110.

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).