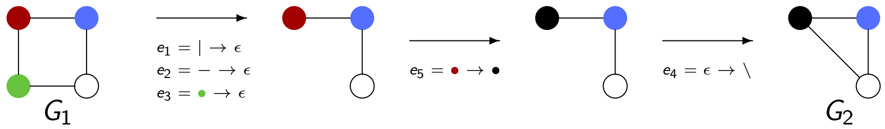
Approximating GED using several stochastically generated solutions and parallelized IPFP

N. Boria, S. Bougleux, and L. Brun

Normandie Univ
ENSICAEN

Agenda

Graph Edit Distance


G_1
$e_1 = | \to \epsilon$
$e_2 = - \to \epsilon$
$e_3 = \bullet \to \epsilon$
$e_5 = \bullet \to \bullet$
$e_4 = \epsilon \to \backslash$
G_2

Example of edit sequence $\gamma \in \Gamma(G_1, G_2)$

Edit costs

Each edit operation e_k is penalized by a cost $c(e_k)$

$$\begin{aligned} \mathrm{GED}(G_1, G_2) &= \min_{\gamma \in \Gamma(G_1, G_2)} \left\{ \sum_{e \in \gamma} c(e) \right\} && (1) \\ &= \min_{\mathbf{x}} \left\{ \frac{1}{2} \mathbf{x}^\top \Delta \mathbf{x} + \mathbf{c}^\top \mathbf{x} \right\} && (2) \\ &= \min_{\mathbf{x}} Q(\mathbf{x}) && (3) \end{aligned}$$

Graph Edit Distance

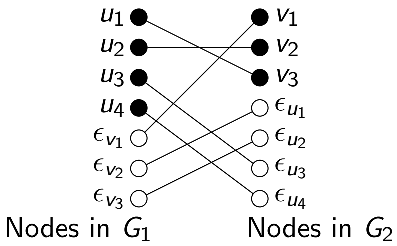
GED

Measures the amount and importance of modifications that are necessary to transform one graph into another

When graphs represent molecules
⇒ GED can provide an accurate measure of similarity or dissimilarity between two molecules
⇒ When all pairwise distances are computed within a set of molecules, some classification or clustering might be performed, and some properties might be predicted
⇒ A median graph of a group of graphs can be computed, hence helping in finding a common ancestor to a set of molecules.

GED as a Quadratic Assignment Problem

Find an assignment **x** between the nodes of the two graphs under the following constraint :



$$\mathrm{GED}(G_1, G_2) = \min_{\mathbf{x}} \left\{ \frac{1}{2}\mathbf{x}^\top \Delta \mathbf{x} + \mathbf{c}^\top \mathbf{x} \right\}$$

with cost matrices : Δ for edge assignement and **c** for node assingment

Complexity

QAP, and thus GED computation is NP-hard.

Agenda

Frank-Wolfe Algorithm

Gradient descent method to find the global minimum of a convex function

Principle

At each iteration k, we dispose of a current continuous solution x_k :

1. Minimize linear approximation of Q around x_k according to its 1st order Taylor expansion
   - ▷ $b_k = \text{argmin}_{b \in \mathbb{R}^{n \times m}} \{ b \nabla Q(x_k) \}$
2. Step size determination by a line search
   - ▷ $\gamma^* = \min_{\gamma \in [0;1]} \{ Q(x_k + \gamma(b_k - x_k)) \}$
3. Update current solution
   - ▷ $x_{k+1} = x_k + \gamma(b_k - x_k)$

Frank-Wolfe Algorithm

A simple procedure to find a local minimum of Q, if not convex

- ▷ Method used in context of Graph Matching (IPFP) [Leordeanu et al. 2009]
- ▷ Also used for GED estimation [Bougleux et al. 2017]
- ▷ Converge generally to a local minimum
- ▷ Strongly impacted by the initialization

IPFP : Impact of the initialization

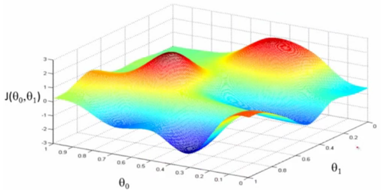


Figure: Example of a non-convex function

Non-convexity of Q

IPFP converges to a local minimum of Q. If Q is not convex, this minimum is not necessarily global.

- ▷ **Impact on time complexity** : if x_1 is far from a local minimum of Q, there may be more iterations to converge
- ▷ **Impact on accuracy** : if Q is non-convex, the returned local minimum depends on x_1

Multiple initializations : mIPFP

A parallel multistart approach based on IPFP.

Procedure

1. Generate a set of k assignments S_k
2. Compute the set of refinements $\{IPFP(\mathbf{x}) \mid \mathbf{x} \in S_k\}$
3. Return $\text{mIPFP}(S_k) = \min_{\mathbf{x} \in S_k} \{Q(\mathbf{y}) : \mathbf{y} = \text{IPFP}(\mathbf{x})\}$

Multiple initializations : mIPFP

A parallel multistart approach based on IPFP.

Procedure

1. Generate a set of k assignments S_k
2. Compute the set of refinements $\{IPFP(\mathbf{x}) \mid \mathbf{x} \in S_k\}$
3. Return $\text{mIPFP}(S_k) = \min_{\mathbf{x} \in S_k} \{Q(\mathbf{y}) : \mathbf{y} = \text{IPFP}(\mathbf{x})\}$

- ▷ Simple procedure
- ▷ Can be easily parallelized, as each IPFP is independent
- ▷ Several kinds of initializations are possible
- ▷ Significant improvement w.r.t. to "single start" versions in terms of distance for little to no computing time cost.

Agenda

Two conflicting criteria for a better generator

Quality of IPFP relies mostly on the quality of the initial solution.
How can we produce better initial solutions for a parallelized algorithm ?

Two conflicting criteria for a better generator

Quality of IPFP relies mostly on the quality of the initial solution.
How can we produce better initial solutions for a parallelized algorithm ?

A good solution generator should follow the two conflicting objectives:

- ▷ Producing solutions that are "far" from one another : **exploration criterion**
- ▷ Producing solutions that are already good solution in terms of GED : **quality criterion**

Our proposition for a better generator: RANDPOST(k,l)

The Algorithm we propose is a refinement of mIPFP : it consists in several iterations of mIPFP where each new iteration generates k new solutions in a stochastic fashion such that each assignment $(i \rightarrow j)$ is picked with a probability roughly equal to:

$$\Psi_{ij} = \frac{\#\text{refined solutions that include}(i \rightarrow j)}{\#\text{refined solutions}}$$

Our proposition for a better generator: RANDPOST(k,l)

The Algorithm we propose is a refinement of mIPFP : it consists in several iterations of mIPFP where each new iteration generates k new solutions in a stochastic fashion such that each assignment $(i \to j)$ is picked with a probability roughly equal to:

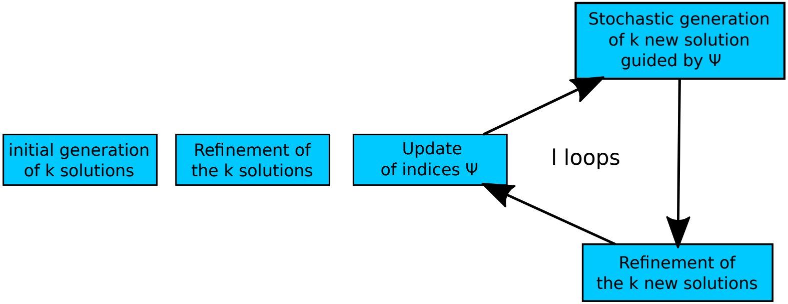
$$\Psi_{ij} = \frac{\#\text{refined solutions that include}(i \to j)}{\#\text{refined solutions}}$$

The randomness of the procedure answers the **exploration criterion**.

Pairwise assignments that are thought to appear in many good solutions are made more likely to be picked by the algorithm in order to answer the **quality criterion**

Our proposition for a better generator: RANDPOST(k,l)

General architecture of algorithm RANDPOST(k,l)


initial generation of k solutions
Refinement of the k solutions
Update of indices Ψ
Stochastic generation of k new solution guided by Ψ
l loops
Refinement of the k new solutions

Agenda

Datasets

Dataset	*Number of graphs*	*Avg Size*
MAO	68	18.4
PAH	94	20.7
MUTA 10-70	10	10-70
ClinTox	25	115.7

▷ Monoamine Oxydase (MAO) dataset .This dataset is composed of 68 molecules divided into two classes: 38 molecules inhibit the monoamine oxidase (antidepressant drugs) and 30 do not.

▷ Polyciclic Aromatic Hydrocarbons dataset (PAH) This dataset is composed cyclic unlabeled graphs. All atoms are carbons, all bounds are aromatics. This is a classification problem (cancerous or not cancerous molecules).

▷ Mutagenicity Graphs (MUTA). Mutagen and non-mutagen molecules.

▷ ClinTox Dataset. Drugs approved by the FDA and those that have failed clinical trials for toxicity reasons.

Benchmark

Save for results on ClinTox dataset, our results were compared to the results of all 9 algorithms that participated to the Graph Distance Contest (ICPR 2016).

- absolute errors were computed w.r.t. to the best solutions found among all 13 algorithms (9 of contest + 4 versions of RANDPOST).

- For a given algorithm, "% best" represents the proportion of pairs of instance where the best GED among all 13 computed GEDs was found.

Experiments - MAO,PAH & ClinTox

Metric Costs

Algorithms	MAO				PAH				ClinTox			
	time	GED	err.	%best	time	GED	err.	%best	time	GED	err.	%best
RANDPOST(1,0)	0.021	30.97	6.84	16	0.025	26.73	14.64	1	6.474	178.73	26.80	0
RANDPOST(40,0)	0.089	24.16	0.03	98	0.117	21.23	9.13	19	18.453	161.77	9.84	4
RANDPOST(20,1)	0.144	24.14	0.01	99	0.182	21.09	8.99	21	30.881	157.35	5.421	15
RANDPOST(10,3)	0.172	24.17	0.04	99	0.263	20.90	8.80	24	51.467	154.72	2.79	39
RANDPOST(5,7)	0.237	24.36	0.23	94	0.439	20.84	8.75	25	76.794	153.06	1.135	70

Non Metric Costs

Algorithms	MAO				PAH				ClinTox			
	time	GED	err.	%best	time	GED	err.	%best	time	GED	err.	%best
RANDPOST(1,0)	0.027	22.73	6.62	17	0.027	20.62	11.84	1	6.245	184.50	23.02	1
RANDPOST(40,0)	0.090	16.14	0.03	98	0.136	15.11	6.33	20	18.846	169.28	7.80	10
RANDPOST(20,1)	0.151	16.12	0.02	99	0.210	15.03	6.25	21	30.019	165.86	4.37	24
RANDPOST(10,3)	0.225	16.14	0.04	99	0.324	14.85	6.07	23	49.901	164.25	2.76	40
RANDPOST(5,7)	0.340	16.30	0.20	95	0.527	14.83	6.05	24	88.499	162.29	0.809	74

Experiments - MUTA Subsets - metric cost function

Algorithms	MUTA 10				MUTA 20				MUTA 30			
	time	GED	err.	%best	time	GED	err.	%best	time	GED	err.	%best
RANDPOST(1,0)	0.013	13.27	1.29	71	0.026	22.01	3.15	23	0.075	32.45	8.19	0
RANDPOST(40,0)	0.024	11.98	0.00	100	0.089	19.00	0.14	86	0.254	25.68	1.42	39
RANDPOST(20,1)	0.040	11.98	0.00	100	0.150	18.93	0.07	93	0.452	25.40	1.14	39
RANDPOST(10,3)	0.069	11.98	0.00	100	0.223	19.01	0.15	88	0.714	25.17	0.91	50
RANDPOST(5,7)	0.123	11.98	0.00	100	0.364	19.11	0.25	81	1.103	25.35	1.09	53

Algorithms	MUTA 40				MUTA 50				MUTA 60			
	time	GED	err.	%best	time	GED	err.	%best	time	GED	err.	%best
RANDPOST(1,0)	0.170	44.83	11.04	3	0.326	48.55	11.21	7	0.609	60.81	13.89	1
RANDPOST(40,0)	0.602	36.07	2.28	24	1.181	40.10	2.76	20	2.898	50.64	3.72	13
RANDPOST(20,1)	1.089	35.08	1.29	43	2.075	39.06	1.72	34	4.914	49.39	2.47	26
RANDPOST(10,3)	1.819	34.87	1.08	46	3.621	38.55	1.21	49	6.545	48.25	1.33	48
RANDPOST(5,7)	2.820	34.57	0.78	60	6.059	38.06	0.72	58	11.091	47.66	0.74	65

Algorithms	MUTA 70				MUTAmix			
	time	GED	err.	%best	time	GED	err.	%best
RANDPOST(1,0)	1.378	75.28	16.22	1	4.972	140.16	5.98	19
RANDPOST(40,0)	4.297	63.90	4.84	15	0.876	136.32	2.14	36
RANDPOST(20,1)	7.665	62.13	3.07	22	1.444	135.54	1.36	50
RANDPOST(10,3)	12.678	60.52	1.46	48	2.434	135.35	1.17	53
RANDPOST(5,7)	18.815	60.29	1.23	59	3.495	134.69	0.51	70

Experiments - MUTA Subsets - anti-metric cost function

Algorithms	MUTA 10				MUTA 20				MUTA 30			
	time	GED	err.	%best	time	GED	err.	%best	time	GED	err.	%best
RANDPOST(1,0)	0.013	23.34	0.88	77	0.019	36.30	2.46	42	0.054	48.62	7.80	4
RANDPOST(40,0)	0.035	22.46	0.00	100	0.089	33.90	0.06	97	0.247	42.31	1.49	45
RANDPOST(20,1)	0.062	22.46	0.00	100	0.144	33.94	0.10	95	0.428	41.93	1.11	54
RANDPOST(10,3)	0.125	22.46	0.00	100	0.247	33.94	0.10	95	0.546	41.59	0.77	65
RANDPOST(5,7)	0.229	22.46	0.00	100	0.386	34.06	0.22	90	0.769	41.86	1.04	61

Algorithms	MUTA 40				MUTA 50				MUTA 60			
	time	GED	err.	%best	time	GED	err.	%best	time	GED	err.	%best
RANDPOST(1,0)	0.123	66.78	9.08	11	0.284	67.67	11.53	5	0.457	83.39	13.05	2
RANDPOST(40,0)	0.570	59.45	1.75	33	1.288	59.23	3.09	23	2.385	73.58	3.24	21
RANDPOST(20,1)	0.898	58.97	1.27	50	2.093	58.14	2.00	33	4.481	72.96	2.62	27
RANDPOST(10,3)	1.461	58.76	1.06	57	3.819	57.82	1.68	47	6.798	71.93	1.59	44
RANDPOST(5,7)	2.119	59.14	1.44	48	5.848	57.49	1.35	54	9.580	71.48	1.14	63

Algorithms	MUTA 70				MUTAmix			
	time	GED	err.	%best	time	GED	err.	%best
RANDPOST(1,0)	1.149	102.91	16.25	0	4.973	106.43	4.95	27
RANDPOST(40,0)	4.228	91.68	5.02	10	1.046	103.65	2.17	43
RANDPOST(20,1)	6.854	89.73	3.07	28	1.786	102.78	1.30	58
RANDPOST(10,3)	10.300	88.66	2.00	46	2.628	102.39	0.91	70
RANDPOST(5,7)	15.548	88.01	1.35	58	3.509	102.35	0.87	65

Agenda

Conclusions and future work

Summary

- ▷ Allows to generate a great number of initial solutions in little time.
- ▷ Improvement w.r.t. simple multistart method, especially on graphs with 30+ nodes
- ▷ By design, less parallelizable that simple multistart.

Future work

1. Test the method with different kinds of initialization methods
2. Test different kinds of Ψ-based probability distributions
3. Make the algorithm choose which criterion (exploration or quality) to favor based on the Ψ indices.
4. Make the method more parallelizable