From tree matching to graph alignment

Luca Ganassali and Laurent Massoulié

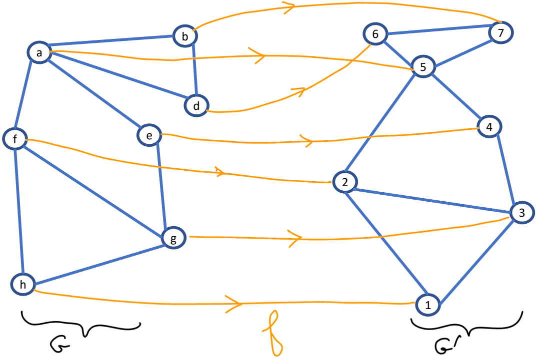
https://arxiv.org/pdf/2002.01258.pdf

Inria

January 26, 2021

The graph isomorphism problem

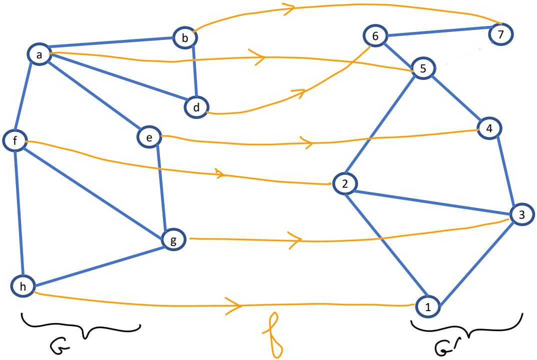
Definition: Given two graphs $G = (V, E)$, $G' = (V', E')$, is there a **graph isomorphism**, i.e. a bijection $f : V \to V'$ such that $(i, j) \in E \Leftrightarrow (f(i), f(j)) \in E'$?



$\to$ Classical problem in NP, thought to be neither in P, nor NP-complete

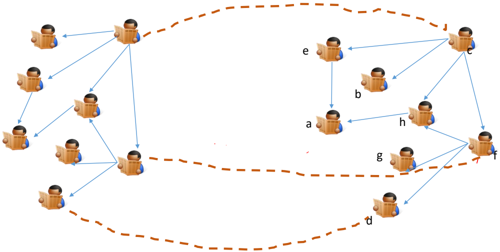
Graph alignment

Relaxed version: bijection f between vertices V of G and V' of G' that preserves **most** edges



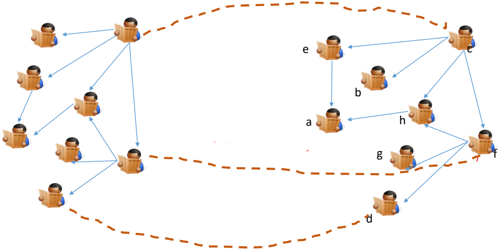
Formally, f minimizes $\sum_{i,j \in V} \left| \mathbb{I}_{(i,j) \in E} - \mathbb{I}_{(f(i),f(j)) \in E'} \right|$
$\rightarrow$ An instance of the NP-hard **quadratic assignment problem:**
$\max_{\Pi} \text{Trace}(A \Pi A' \Pi^{\top})$ where Π runs over permutation matrices

Applications



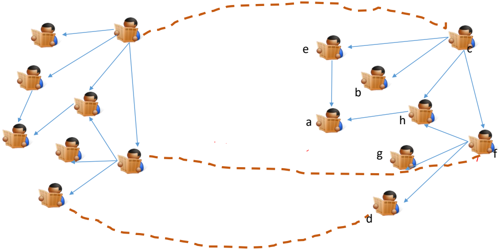
- De-anonymization of users of social network

Applications



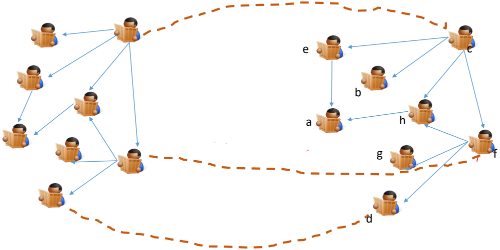
- De-anonymization of users of social network
- Align protein interaction networks for cells of species $\rightarrow$ infer function of proteins in biology of species A from knowledge of protein function in biology of species B

Applications



- De-anonymization of users of social network
- Align protein interaction networks for cells of species → infer function of proteins in biology of species A from knowledge of protein function in biology of species B
- Align meshes of 3D images of hearts to transfer segmentation into distinct parts from image of reference heart

Applications

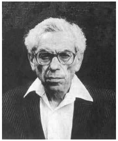


- De-anonymization of users of social network
- Align protein interaction networks for cells of species → infer function of proteins in biology of species A from knowledge of protein function in biology of species B
- Align meshes of 3D images of hearts to transfer segmentation into distinct parts from image of reference heart
- Align graphs between words in languages A and B to construct dictionary between the two languages
- ...

Generative, probabilistic models of graphs

Erdős-Rényi random graph $\mathcal{G}(n, p)$:

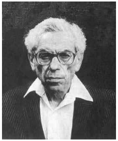
n vertices. Each edge (i, j) present with probability p independently of other edges.



Generative, probabilistic models of graphs

Erdős-Rényi random graph $\mathcal{G}(n, p)$:

n vertices. Each edge (i, j) present with probability p independently of other edges.

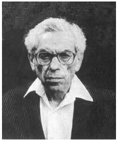


Correlated Erdős-Rényi graphs $(G_1, G_2) \sim \mathrm{ERC}(n, p, s)$:
Start from "master graph" $G_0 \sim \mathcal{G}(n, p/s)$

Generative, probabilistic models of graphs

Erdős-Rényi random graph $\mathcal{G}(n, p)$:

n vertices. Each edge (i, j) present with probability p independently of other edges.



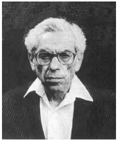
Correlated Erdős-Rényi graphs $(G_1, G_2) \sim \mathrm{ERC}(n, p, s)$:
Start from "master graph" $G_0 \sim \mathcal{G}(n, p/s)$

- Keep each edge with prob. s to form $G_1 \sim \mathcal{G}(n, p)$ and independently, $G_2' \sim \mathcal{G}(n, p)$

Generative, probabilistic models of graphs

Erdős-Rényi random graph $\mathcal{G}(n, p)$:

n vertices. Each edge (i, j) present with probability p independently of other edges.



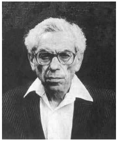
Correlated Erdős-Rényi graphs $(G_1, G_2) \sim \mathrm{ERC}(n, p, s)$:
Start from "master graph" $G_0 \sim \mathcal{G}(n, p/s)$

- Keep each edge with prob. s to form $G_1 \sim \mathcal{G}(n, p)$ and independently, $G_2' \sim \mathcal{G}(n, p)$
  $\rightarrow \mathbb{P}((i, j) \in E_1 \cap E_2') = p * s$,
  $\rightarrow \mathbb{P}((i, j) \in E_1, (i, j) \notin E_2') = p(1 - s)$

Generative, probabilistic models of graphs

Erdős-Rényi random graph $\mathcal{G}(n, p)$:

n vertices. Each edge (i, j) present with probability p independently of other edges.




Correlated Erdős-Rényi graphs $(G_1, G_2) \sim \mathrm{ERC}(n, p, s)$:
Start from "master graph" $G_0 \sim \mathcal{G}(n, p/s)$

- Keep each edge with prob. s to form $G_1 \sim \mathcal{G}(n, p)$ and independently, $G_2' \sim \mathcal{G}(n, p)$
  $\rightarrow \mathbb{P}((i, j) \in E_1 \cap E_2') = p * s$,
  $\rightarrow \mathbb{P}((i, j) \in E_1, (i, j) \notin E_2') = p(1 - s)$
- Shuffle labels of nodes of G_2' uniformly at random to form G_2
  Formally: random permutation σ;
  Adjacency matrix $A_2 = \Pi_\sigma A_2' \Pi_\sigma^\top$

Goal: recover permutation σ from graphs G_1 and G_2

Exact recovery of permutation σ:

- Information-theoretically feasible iff $nps = \log n + \omega(1)$ [Cullina-Kyavash'16]
- Polynomial-time feasible if $np \geq \log^{\alpha}(n)$ and $1 - s \leq \log^{-\beta}(n)$ [Ding et al.'18]

$\rightarrow$ Recovery of σ only feasible for random graphs with average degree $np = \Omega(\log n)$

Goal: recover permutation σ from graphs G_1 and G_2

Exact recovery of permutation σ:

- Information-theoretically feasible iff $nps = \log n + \omega(1)$ [Cullina-Kyavash'16]
- Polynomial-time feasible if $np \geq \log^{\alpha}(n)$ and $1 - s \leq \log^{-\beta}(n)$ [Ding et al.'18]

$\rightarrow$ Recovery of σ only feasible for random graphs with average degree $np = \Omega(\log n)$

This work: polynomial-time recovery, in sparse regime $np = O(1)$.

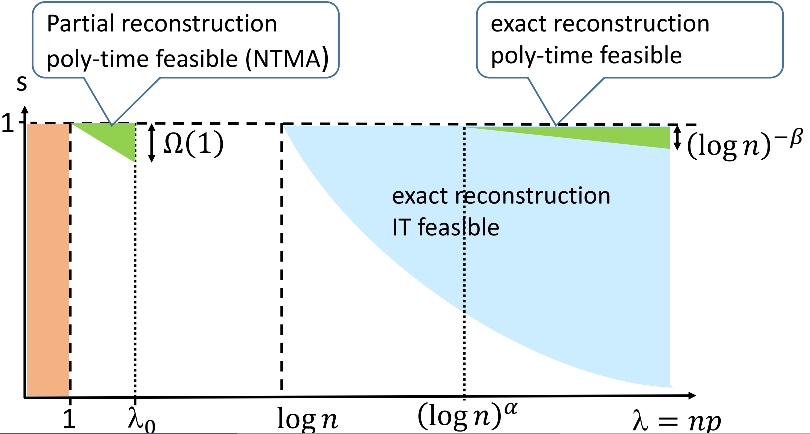
$\rightarrow$ We relax objective from exact recovery to partial recovery:

Construct permutation $\hat{\sigma}$ from G_1, G_2 such that
$\text{overlap}(\hat{\sigma}) := \frac{1}{n} \sum_{i=1}^{n} \mathbb{I}_{\sigma_i = \hat{\sigma}_i} = \Omega(1)$

Main result

Theorem

Let $p = \lambda/n$ for fixed $\lambda \in (1, \lambda_0]$. There exists $s^*(\lambda) < 1$ such that for all $s \in (s^*(\lambda), 1]$, the Neighborhood Tree Matching Algorithm (NTMA) returns a permutation $\hat{\sigma}$ achieving positive overlap with high probability.



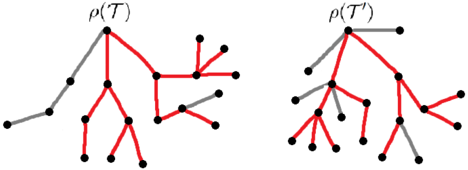
Outline

- Tree matching weights
- NTMA algorithm
- Matching weights for pairs of random trees
- Proof outline and experiments

Tree matching weights

Definition

Given two rooted trees $\mathcal{T}$, $\mathcal{T}'$ and integer $d \geq 0$, **matching weight** $\mathcal{W}_d(\mathcal{T}, \mathcal{T}')$: largest number of leaves of all rooted sub-trees $\mathcal{T}''$ of both $\mathcal{T}$, $\mathcal{T}'$ of depth d.

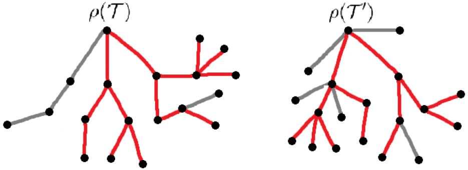


Example of two trees $\mathcal{T}$, $\mathcal{T}'$ with $\mathcal{W}_3(\mathcal{T}, \mathcal{T}') = 7$, where an optimal $t \in \mathcal{A}_3$ is drawn in red.

Tree matching weights

Definition

Given two rooted trees $\mathcal{T}$, $\mathcal{T}'$ and integer $d \geq 0$, **matching weight** $\mathcal{W}_d(\mathcal{T}, \mathcal{T}')$: largest number of leaves of all rooted sub-trees $\mathcal{T}''$ of both $\mathcal{T}$, $\mathcal{T}'$ of depth d.



Example of two trees $\mathcal{T}$, $\mathcal{T}'$ with $\mathcal{W}_3(\mathcal{T}, \mathcal{T}') = 7$, where an optimal $t \in \mathcal{A}_3$ is drawn in red.

Recursive computation: $\mathcal{W}_d(\mathcal{T}, \mathcal{T}') = \max \sum_{(i,u) \in m} \mathcal{W}_{d-1}(\mathcal{T}_i, \mathcal{T}'_u)$

where max over matchings m between neighbors i of $\rho(\mathcal{T})$ and u of $\rho(\mathcal{T}')$, and $\mathcal{T}_i$: tree rooted at i obtained from $\mathcal{T}$ by removing edge $(\rho(\mathcal{T}), i)$

A first attempt

Match vertices i of G_1 and u of G_2 whose respective d-neighborhoods:

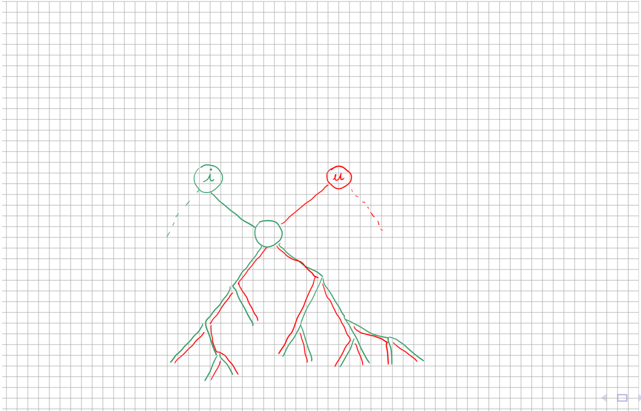
- are trees $\mathcal{T}_1$, $\mathcal{T}_2$
- with large matching weight $\mathcal{W}_d(\mathcal{T}_1, \mathcal{T}_2)$

A first attempt

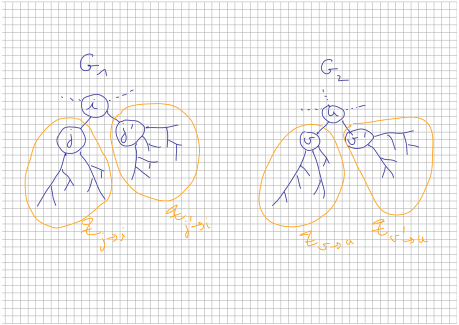
Match vertices i of G_1 and u of G_2 whose respective d-neighborhoods:

- are trees $\mathcal{T}_1$, $\mathcal{T}_2$
- with large matching weight $\mathcal{W}_d(\mathcal{T}_1, \mathcal{T}_2)$

Problem: false positives caused by nearby nodes



Neighborhood tree matching algorithm



Pair of nodes $(i, u) \in V_1 \times V_2$ whose d-neighborhood in G_1, resp. G_2 is a tree:

- if $\exists j, j' \overset{1}{\sim} i,\ v, v' \overset{2}{\sim} u$ such that $\mathcal{W}_d(\mathcal{T}_{j\to i}, \mathcal{T}_{v\to u}),\ \mathcal{W}_d(\mathcal{T}_{j'\to i}, \mathcal{T}_{v'\to u}) > \tau$, add pair (i, u) to set $\mathcal{S}$
- Then for $d = \Theta(\log n),\ \tau = \Theta((\lambda s)^d)$, with high probability:
  $\frac{1}{n}\sum_{i\in V_1} \mathbb{I}_{(i,\sigma(i))\in\mathcal{S}} = \Omega(1)$,
  $\frac{1}{n}\sum_{i\in V_1} \mathbb{I}_{\exists u\neq\sigma(i):(i,u)\in\mathcal{S}} = o(1)$.

Matching weights for independent random trees

$\mathcal{T}$, $\mathcal{T}'$: two independent Galton-Watson branching random trees, with offspring distribution Poisson(λ).

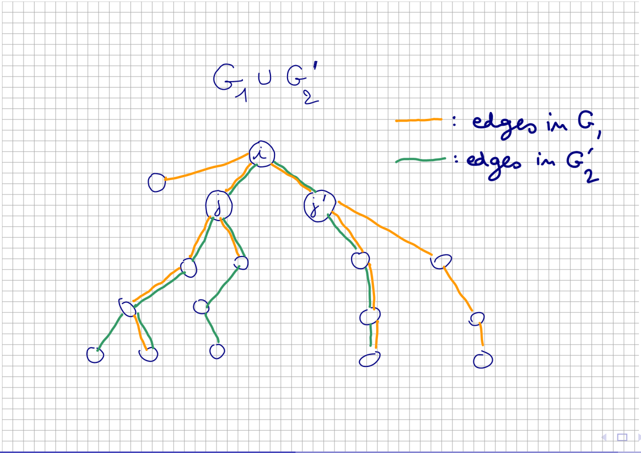
Theorem

For $\lambda \in (1, \lambda_0]$ and $s \in (s^*(\lambda), 1]$, then there exists $\gamma < \lambda s$ such that $\mathcal{W}_d(\mathcal{T}, \mathcal{T}') \ll \gamma^d$ as $d \to \infty$.

Proof: Probabilistic bounds on $\mathcal{W}_d(\mathcal{T}, \mathcal{T}')$ established by induction on d.

Arguments for main result: local structure of graphs G_1, G_2

- Local neighborhood of $i \in V_1$ in G_1: Poisson(λ) Galton-Watson branching process.
- Local structure of union graph $G_1 \cup G_2'$: three-type branching process
- Local structure of intersection graph $G_1 \cap G_2'$: Poisson(λs) Galton-Watson branching process.



Arguments, continued

$\rightarrow$ For $u = \sigma(i)$, $\mathcal{W}_d(\mathcal{T}_i, \mathcal{T}_u) \geq$ size at generation d of Poisson(λs) Galton-Watson branching tree, hence $\approx (\lambda s)^d$

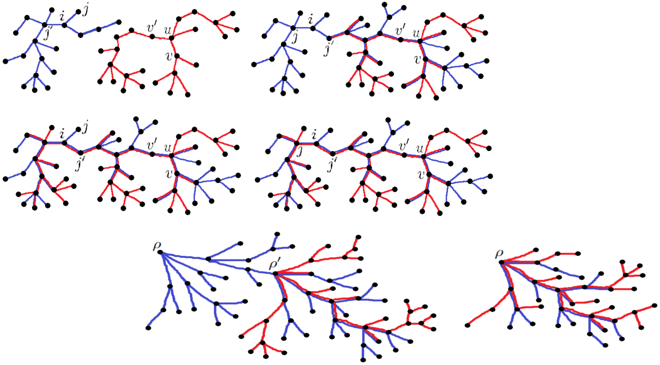
$\rightarrow$ For nodes i, u "far apart" in union graph, $\mathcal{W}_d(\mathcal{T}_i, \mathcal{T}_u) \approx \mathcal{W}_d(\mathcal{T}, \mathcal{T}')$ for $\mathcal{T}, \mathcal{T}'$: independent, Poisson($\lambda$) branching trees, hence $\ll \gamma^d$ for $\gamma < \lambda s$.

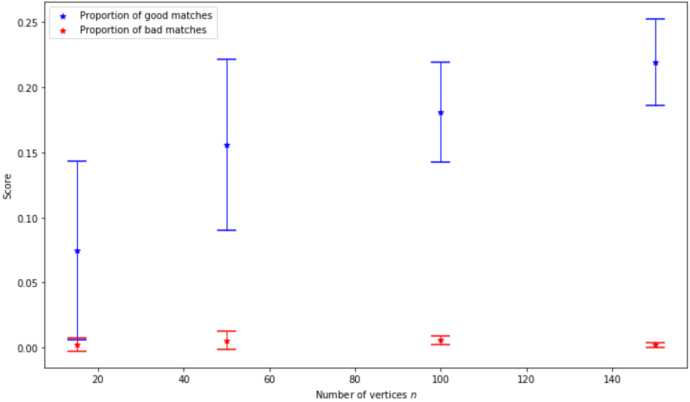
Arguments, continued

$\rightarrow$ For $u = \sigma(i)$, $\mathcal{W}_d(\mathcal{T}_i, \mathcal{T}_u) \geq$ size at generation d of Poisson(λs) Galton-Watson branching tree, hence $\approx (\lambda s)^d$

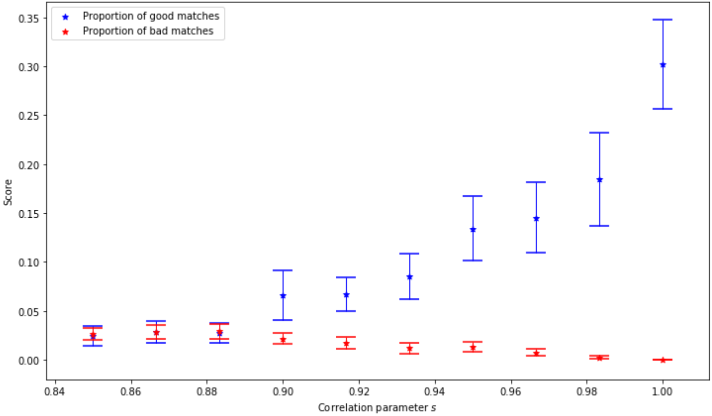
$\rightarrow$ For nodes i, u "far apart" in union graph, $\mathcal{W}_d(\mathcal{T}_i, \mathcal{T}_u) \approx \mathcal{W}_d(\mathcal{T}, \mathcal{T}')$ for $\mathcal{T}, \mathcal{T}'$: independent, Poisson$(\lambda)$ branching trees, hence $\ll \gamma^d$ for $\gamma < \lambda s$.

Several other cases need to be dealt with...





Mean score of NTMA–2 for $\lambda = 2.1$, $d = 5$ (25 iterations per value of n).



(a) $n = 150, \lambda = 1.4, d = 5.$

Conclusions and outlook

- Graph alignment: important unsupervised learning problem with many applications
- NTMA: first method proven to succeed at partial alignment in relevant regime of sparse graphs
- To be done: boundaries of phases in (λ, s) diagram, in particular IT-feasibility and poly-time feasibility of partial alignment (see [Hall-M'20]: partial alignment IT-feasible for $nqs = \Theta(1),\ 1 - s = \Omega(1)$)
- Extend NTMA for better scalability and handling of denser graphs (with more cycles)

Thanks!