Iterative graph alignment via supermodular approximation

Aritra Konar
and
Nicholas D. Sidiropoulos

Department of Electrical and Computer Engineering

IEEE International Conference on Data Mining
Beijing, China
November 9, 2019
Graph Matching

Problem: Find correspondence mapping between vertex sets that best preserves adjacency relations

\[ A = \begin{bmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{bmatrix} \quad A \in \mathbb{R}^{n_A \times n_A} \]

\[ B = \begin{bmatrix} 0 & 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 & 1 \\ 1 & 1 & 0 & 1 & 0 \end{bmatrix} \quad B \in \mathbb{R}^{n_B \times n_B} \]

\[
\begin{align*}
\text{min.} & \quad \| B - PAP^T \|_F^2 \\
\text{s.t.} & \quad P \in \mathcal{P} := \{ P \in \{0, 1\}^{n_B \times n_A} \mid P^T P = I \}
\end{align*}
\]
Applications

- Network de-anonymization
- Electronic circuit design
- Scene matching
- Ontology alignment
Graph Matching

- **Vectorization:** Define $x = \text{vec}(P)$, $n = n_A n_B$

\[
\begin{align*}
\text{max.} & \quad \left\{ f(x) := x^T (A \otimes B) x \right\} \\
\text{s.t.} & \quad x \in \mathcal{M}
\end{align*}
\]

Maximize edge overlap

Set of maximum cardinality matchings in $\mathcal{G}_C$

$\mathcal{G}_C = (\mathcal{V}_A \cup \mathcal{V}_B, \mathcal{E}_C)$
Computational Challenges

- **Graph Matching**
  - Corresponds to a quadratic assignment problem [Koopmans-Beckmann 57]

- **Theoretical:** [Sahni-Gonzalez 76]
  - NP-hard (contains subgraph isomorphism as a special case)
  - NP-hard to approximate within constant-factor of optimum

- **Practical:**
  - Space and time complexity of computing + storing $A \otimes B$
  - Requires quadratic memory in the size of the graphs
Prior Art

Continuous domain

- Linear programming relaxation
  [Sherali-Adams 13]
- Semidefinite programming relaxation
  [Sherali-Adams 13]
- Spectral methods: [Umeyama 1988, Singh et al. 09, Feizi et al. 19]

Discrete domain

- Branch and bound [Bazara 83]
- Message passing [Bayati et al. 10]
Is there a more principled approach that works entirely in the combinatorial domain?

- Represent discrete function as a set function

\[ f : \{0, 1\}^n \rightarrow \mathbb{R} \]

\[ F : 2^n \rightarrow \mathbb{R} \]

- Discrete problem = subset selection problem
Problem Reformulation

- Final formulation:

\[
\max_{S \in \mathcal{I}_A \cap \mathcal{I}_B} \left\{ F(S) := 1_S^T (A \otimes B) 1_S \right\}
\]

where

\[
\begin{align*}
\mathcal{I}_A &= \{ S \subset \mathcal{E}_C, |S \cap \delta(i)| \leq 1, \forall i \in \mathcal{V}_A \}, \\
\mathcal{I}_B &= \{ S \subset \mathcal{E}_C, |S \cap \delta(j)| \leq 1, \forall j \in \mathcal{V}_B \}
\end{align*}
\]

- Conventional wisdom:
  - Constraints are “harder” to handle compared to the objective

- Our perspective:
  - The opposite is true
  - Constraints: \( S \in \mathcal{I}_A \cap \mathcal{I}_B \iff \text{matroid intersection} \)
A closer look: objective function

- Key fact: $F(S)$ is a monotone, supermodular function [Konar-Sidiropoulos 19]

- Monotonicity: $A \subseteq B \implies F(A) \leq F(B)$

- Supermodularity: For all $A \subseteq B \subseteq \mathcal{E}_C \setminus \{e\}$

\[
F(A \cup \{e\}) - F(A) \leq F(B \cup \{e\}) - F(B)
\]

An improving returns property, reminiscent of convexity
Graph Matching

- **Key Result:**
  - Graph matching is a supermodular maximization problem subject to matroid intersection constraints!

- **Take-away:**
  - Constraints are manageable, but objective function is difficult to maximize

- **Can we exploit supermodularity for approximate maximization?**
Main Idea:

- Iteratively maximize sequence of global lower bounds on reward

\[ \partial_f(x) = \{ g \in \mathbb{R} \mid f(y) \geq f(x) + g(y - x), \forall y \in \mathbb{R} \} \]

Does the idea carry over to the discrete domain?
Key fact: Supermodular functions possess (discrete) subgradients! [Jegelka-Bilmes 11]

\[ \partial_F(\mathcal{X}) = \{ g \in \mathbb{R}^n \mid F(\mathcal{Y}) \geq F(\mathcal{X}) + G(\mathcal{Y}) - G(\mathcal{X}), \forall \mathcal{Y} \subseteq \mathcal{E}_C \} \]

where \( G(\mathcal{Y}) = g^T \mathbf{1}_\mathcal{Y} = \sum_{i \in \mathcal{Y}} g_i \)

Construction of global lower bound: [Bai-Bilmes 18]

- Pick any \( g \in \partial_F(\mathcal{X}) \), and define

\[ m_\mathcal{X}(\mathcal{Y}) := F(\mathcal{X}) + G(\mathcal{Y}) - G(\mathcal{X}) \]

- Furthermore:

\[ m_\mathcal{X}(\mathcal{X}) := F(\mathcal{X}) \text{ and } m_\mathcal{X}(\mathcal{Y}) \leq F(\mathcal{Y}), \forall \mathcal{Y} \subseteq \mathcal{E}_C \]

A global lower bound on the reward function!
Discrete Majorization Minimization

- **Simplification:** For any given $S \subseteq \mathcal{M}$

  **Option I:**
  \[ g_1(j) = \begin{cases} 
  2\deg_B(\pi(i))\deg_A(i), & \forall j \in S \\
  2b_{\pi(i)}^T P a_i, & \forall j \notin S 
  \end{cases} \]

  **Option II:**
  \[ g_2(j) = \begin{cases} 
  2b_{\pi(i)}^T P a_i, & \forall j \in S \\
  0, & \forall j \notin S 
  \end{cases} \]

- No Kronecker products required!
- In practice, use Option II (linear memory in size of input graph)
Discrete Majorization Minimization

- The algorithm:
  - Initialization: \( S^{(0)} \in \mathcal{M} \)
  - Iterate: \( k = \{0, 1, 2 \cdots \} \)
    - Obtain subgradient \( \mathbf{g}^{(k)} \in \partial_F(S^{(k)}) \)
    - Compute update

\[
S^{(k+1)} \in \arg \max_{S \in \mathcal{I}_A \cap \mathcal{I}_B} \left\{ m_{S^{(k)}}(S) := F(S^{(k)}) + \mathbf{G}_k(S) - G_k(S^{(k)}) \right\}
\]

\[
S^{(k+1)} \in \arg \max_{S \in \mathcal{I}_A \cap \mathcal{I}_B} \left\{ G_k(S) = (\mathbf{g}^{(k)})^T \mathbf{1}_S \right\}
\]

Linear assignment / maximum weight bipartite matching problem

- Repeat
Discrete Majorization Minimization

- **Features:**
  - Purely combinatorial - solves a few weighted bipartite matching problems
  - Guaranteed to improve the reward function:
    \[ F(S^{(0)}) \leq F(S^{(1)}) \leq F(S^{(2)}) \leq F(S^{(3)}) \leq \cdots \]
  - Guaranteed to maintain feasibility:
    \[ S^{(k)} \in \mathcal{M}, \forall k \in \{0, 1, 2, \cdots \} \]
  - Complexity: Dominated by cost of solving weighted bipartite matching problem
    - (For \(n_A = n_B\)) Hungarian algorithm [Kuhn-Munkres 58] / Jonker-Volgenant algorithm [Jonker-Volgenant, 87]
    - (For \(n_A < n_B\)) Network-Simplex algorithm [Orlin 97]
    - Greedy matching
    - Sinkhorn Matrix Balancing [Cuturi 13, Sinkhorn 67]
Experiments

Setup:
- Given real world graph $A$, generate noisy graph $B$

\[ B = P(A + (1 - A) * Q)P^T \]

- where $Q$ is a random Erdos-Renyi noise graph

Benchmarks:
- Umeyama’s Method: full EVD of each adjacency [Umeyama 1988]
- Eigen-Align (EA): top eigen-vector of each adjacency [Feizi et. al 2016]
- IsoRank: Random-walk based [Singh et. al 2008]
- Feature Engineering (FE): local + egonet features [Berlingerio et. al 2012]

- Apply greedy matching on output of each algorithm to obtain final correspondence mapping
Experiments

- **Implementation:**
  - Initialization: Use output of FE
  - Regularization: Use node-level similarity matrix of FE
  - Inner-solver:
    - Exact: Jonker-Volgenant algorithm
    - Inexact: 5 iterations of Sinkhorn Matrix balancing + Greedy

- **Evaluation Metrics:**
  - Edge correctness
  - Relative degree difference (by degree)

\[
\text{rdd}(i, \pi(i)) = \left( 1 + \frac{|\deg(i) - \deg(\pi(i))|}{(\deg(i) + \deg(\pi(i))/2)\right)^{-1}}
\]

- Runtime
Results

A. Thaliana (PPI): $n = 2,082, \ m = 4,145$

15 % accuracy improvement over FE, MM-Inexact best performance overall
Results

A. Thaliana (PPI): $n = 2,082$, $m = 4,145$

Significant improvement in RDD alignment scores for bottom 5% nodes
Conclusions

- **Graph Matching through the lens of supermodularity:**
  - Maximizing a supermodular function subject to matroid intersection constraints
  - Combinatorial local search based on discrete MM
    - Solve a sequence of bipartite matching problems
    - Does not require computing expensive Kronecker products
    - FE + Inexact version yields state-of-the-art performance on real-world data

- **Future Work:**
  - Instance specific approximation guarantees
  - Joint embedding + matching
Thank you!
A closer look: the constraints

- **Interpretation:** $S \in \mathcal{I}_A \cap \mathcal{I}_B$

- **Set $\mathcal{I}_A$:**
  - For every source vertex, only one *outgoing* edge can be selected
  - A partition matroid on the edges of $\mathcal{G}_C$

- **Set $\mathcal{I}_B$:**
  - For every target vertex, only one *incoming* edge can be selected
  - Also a partition matroid on the edges of $\mathcal{G}_C$

- Matching sets equivalent to intersection of partition matroids
Exact or Inexact?

A. Thaliana (PPI): n = 2082, m = 4145

Inexact: 5 % performance loss, 10x speedup, approx. convergence in 1 iteration
A. Thaliana (PPI): $n = 2,082$, $m = 4,145$
Results

Stanford-CS (web): $n = 2,759$, $m = 10,270$

10 % accuracy improvement over FE, MM-Inexact best performance overall
Results

Stanford-CS (web): n = 2,759, m = 10,270

Significant improvement in RDD alignment scores of bottom 5% nodes
Results

Ca-GrQc (co-authorship): \( n = 5,242, m = 14,490 \)

10% accuracy improvement over FE, MM-Inexact best performance overall
Results

Ca-GrQc (co-authorship): \( n = 5,242, m = 14,490 \)

Significant improvement in RDD alignment scores of bottom 5% nodes