A Note on Learning Algorithms for Quadratic Assignment with Graph Neural Networks

{ Alex Nowak, Soledad Villar, Afonso S. Bandeira, Joan Bruna} Center for Data Science, CIMS, NYU

Objectives

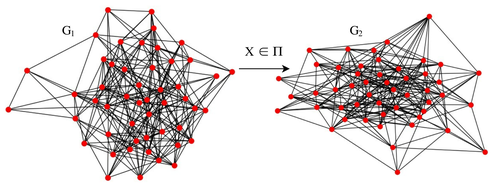
- Automatically learn algorithms for quadratic assignment from solved instances using neural networks.
- Give an approach to check whether there exists any statistical to computational gap for the quadratic assignment problem.

Quadratic Assignment

Graph matching:
For A, B $n \times n$ adjacency matrices,

$\text{minimize}_X \ \|AX - XB\|_F^2, \ \text{ subject to } X \in \Pi.$

Π is the set of all permutation matrices.



Traveling salesman:
Graph matching between A pairwise distance matrix, and B, adjacency matrix of the cycle.

QAP is a very interesting problem:

- NP-hard and hard to approximate.
- Natural statistical models for the inputs.
- Recovery thresholds not fully understood.

References

[1] Jiming Peng, Hans Mittelmann, and Xiaoxue Li. A new relaxation framework for quadratic assignment problems based on matrix splitting. *Mathematical Programming Computation*, 2(1):59–77, 2010.

[2] Alaa Saade, Florent Krzakala, and Lenka Zdeborová. Spectral clustering of graphs with the bethe hessian. In *Advances in Neural Information Processing Systems*, pages 406–414, 2014.

[3] Michael M. Bronstein, Joan Bruna, Yann LeCun, Arthur Szlam, and Pierre Vandergheynst. Geometric deep learning: going beyond euclidean data. *CoRR*, abs/1611.08097, 2016.

Graph Neural Networks

- GNN [3]
Given a signal $v^0 \in \mathbb{R}^{n \times d_0}$ on the vertices of a graph G, a Graph Neural Network computes

$$v_l^{k+1} = \rho\left(\sum_{M \in \mathcal{M}} \theta_M^k \left(M v^k\right)_l\right), \ l = 1 \dots d_{k+1}$$

$\mathcal{M} = \{I_n, D, A, A^2, \dots, A^{2^J}, U\} : \mathbb{R}^{n \times d} \to \mathbb{R}^{n \times d}$ generator family,
$\Theta = \{\theta_1^k, \dots, \theta_{|\mathcal{M}|}^k\}$, $\theta_M^k \in \mathbb{R}^{d_k \times d_{k+1}}$ trainable parameters.
The output of the GNN is $E = v^K \in \mathbb{R}^{n \times d_K}$ where K is the number of layers.

- Simaese GNN
Apply the GNN to G_1, G_2 graphs and outputs a squared matrix resulting from the outer product of the embeddings: $E_1^T E_2 \in \mathbb{R}^{n \times n}$

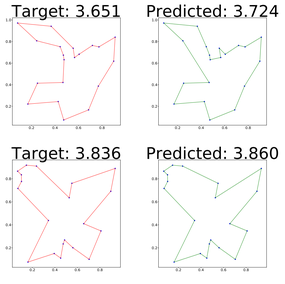
- Neural Net naturally adapted to graphs.
- Number of parameters independent of input size.
- Scalability for sparse graphs.
- Choice of graph generators encodes prior information of the task.

TSP with GNNs

Learning TSP from solved instances
Feed G into a GNN and try to find the cycle with minimum cost:

$$\text{minimize}_\Theta \ D_{KL}(\text{softmax}(E^T E - \eta I_n) \,\|\, \frac{1}{2} A_C)$$

The predicted cycle is computed from the matrix $\text{softmax}(E^T E - \eta I_n)$ using a beam search strategy. The approximation ratio over the test set is **1.027**. We perform still worse than Christofides (1.010).



Matching Random Graphs with Siamese GNNs

Feed G_1, G_2 into a Siamese GNN and train the model to recover the identity.

$$\text{minimize}_\Theta \ D_{KL}(\text{softmax}(E_A^T E_B) \,\|\, I_n)$$

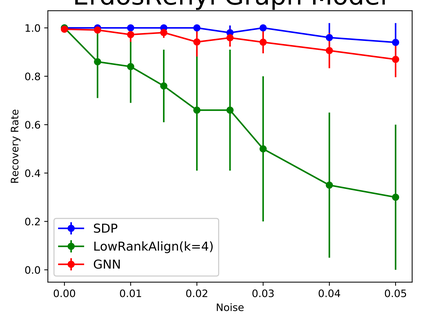
We compare the Siamese GNN $\mathcal{O}(n^2)$ with the SDP $\mathcal{O}(n^4)$ [1] and LowRankAlign $\mathcal{O}(n^3)$ [2].

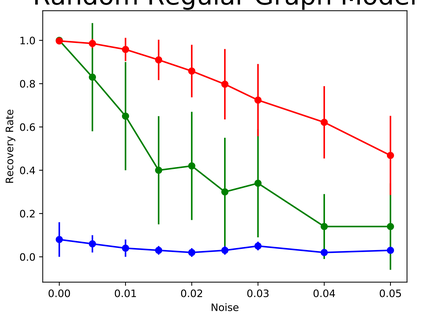
Matching Erdos-Renyi Graphs
G_1 is a random ER graph with edge density p_e. G_2 is a small perturbation of G_1 according to:

$$G_2 = G_1 \odot (1 - Q) + (1 - G_1) \odot Q'$$

where Q and Q' are binary random matrices whose entries are drawn from i.i.d. Bernoulli distributions such that $\mathbb{P}(Q_{ij} = 1) = p_e$ and $\mathbb{P}(Q'_{ij} = 1) = p_{e_2}$ with $p_{e_2} = p_e \frac{p}{p-1}$.

Matching Regular Graphs
G_1 is a random regular graph and G_2 is a perturbation of G_1 according to the previous noise model. Although G_2 is in general not a regular graph, the "signal" to be matched to, G_1, is a regular graph.





Discussion and Open Problems

Results
We show how under natural graph statistical models, the data-driven approach with GNNs outperforms the other state-of-the-art baselines with significantly fewer complexity.

Open Problems

1. Is it possible to outperform advanced heuristic methods for TSP with data-driven models? What about optimizing directly the cost?
2. Can a GNN express an efficient algorithm when lying above the computational threshold? And if it does, can it be learned by SGD from examples?
3. The Stochastic Block Model (SBM) is a very good example; precise predictions of statistical and computational thresholds. Can we detect large number of communities with GNNs where the information and computational thresholds suspect to differ?
4. The performance of this algorithm depends on which operators are used in the GNN. Does it exist a principled way of choosing the generator family from the task?
5. Understand the limits of the QAP, both statistically and computationally. Understand the limits of the GNN approach.