Discrete Optimizations using Graph Convolutional Networks

Radu Balan, Naveed Haghani

Department of Mathematics, Applied Mathematics Applied Statistics and Scientific Computing
University of Maryland, College Park, MD

August 13, 2019
SPIE Wavelets and Sparsity XVIII Conference, Session on "Applications of Frames and Related Transforms"

Acknowledgments

"This material is based upon work partially supported by the National Science Foundation under grant no. DMS-1816608 and LTS under grant H9823013D00560049. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation."

Collaborators: **Maneesh Singh, Julian Yarkoni** (Verisk, NJ)

Table of Contents:

Quadratic Optimization Problems

Consider two symmetric (and positive semidefinite) matrices $A, B \in \mathbb{R}^{n\times n}$. The *quadratic assignment problem* asks for the solution of

$$\begin{array}{ll} \textit{maximize} & \textit{trace}(\Pi A \Pi^T B) \\ \text{subject to:} & \\ \Pi \in S_n & \end{array}$$

where S_n denotes the symmetric group of $n \times n$ permutation matrices.
Purpose of this talk: Present a Graph Deep Learning architecture designed to solve the Quadratic Assignment Problem.

QAP
Motivation

Consider two $n \times n$ symmetric matrices A, B. In the alignment problem for quadratic forms one seeks an orthogonal matrix $U \in O(n)$ that minimizes

$$\|UAU^T - B\|_F^2 := trace((UAU^T - B)^2) = \|A\|_F^2 + \|B\|_F^2 - 2trace(UAU^T B).$$

The solution is well-known and depends on the eigendecomposition of matrices A, B: if $A = U_1 D_1 U_1^T$, $B = U_2 D_2 U_2^T$ then

$$U_{opt} = U_2 U_1^T \quad , \quad \|U_{opt} A U_{opt}^T - B\|_F^2 = \sum_{k=1}^{n} |\lambda_k - \mu_k|^2,$$

where $D_1 = diag(\lambda_k)$ and $D_2 = diag(\mu_k)$ are diagonal matrices with eigenvalues ordered monotonically.

QAP

Motivation 2

The challenging case is when U is constrained to belong to the permutation group. In this case, the previous minimization problem

$$\min_{U \in S_n} \|UAU^T - B\|_F$$

turns into the QAP:

$$\max_{U \in S_n} trace(UAU^T B).$$

In the case A, B are graph Laplacians (or adjacency matrices), an efficient solution to this optimization problem would solve the graph isomorphism problem, one of the remaining milenium problems: decide if two given graphs are the same modulo vertex labelling.

Prior work to discrete optimizations using deep learning

- Direct approach to discrete optimization: Pointer Networks (Ptr-Nets) utilize sequence-to-sequence Recurrent Neural Networks [Vinyals'15];
- Reinforcement learning and policy gradients: [Bello'16]
- Graph embedding and deep Q-learning: [Dai'17]
- QAP using graph deep learning: [Nowak'17] utilizes siamese graph neural networks that act on A and B independently to produce embeddings E_1 and E_2; then the product $E_1 E_2^T$ is transformed into a permutation matrix through soft-max and cross-entropy loss.

Shift Invariance Properties

Consider $A = A^T$ and $B = B^T$ (no positivity assumption).

Lemma

The QAP associated to (A, B) has the same optimizer as the QAP associated to $(A - \lambda I, B - \mu I)$, where $\lambda, \mu \in \mathbb{R}$.

Indeed, the proof of this lemma is based on the following direct computation:

$$trace(\Pi(A-\lambda I)\Pi^T(B-\mu I)) = trace(\Pi A \Pi^T B) - \mu trace(A) - \lambda trace(B) + n\lambda\mu$$

A consequence of this lemma is that, without loss of generality, we can assume $A, B \geq 0$. In fact, we can shift the spectrum to vanish the smallest eigenvalues of A, B.

The case of Rank One

Assume now $A = aa^T$ and $B = bb^T$ are non-negative rank one matrices. Then:

$$trace(\Pi A \Pi^T B) = |b^T \Pi a|^2 = (trace(\Pi ab^T))^2 = \frac{1}{trace(AB)}(trace(\Pi AB))^2$$

In this case we obtain the explicit solution to the QAP:

Lemma

Assume $A = aa^T$ and $B = bb^T$ are rank one. Then the QAP optimizer is the optimizer of one of the following two optimization problems:

$$\begin{array}{cc} maximize & trace(\Pi C) \\ subject\ to: & \\ \Pi \in S_n & \end{array} \quad \text{or} \quad \begin{array}{cc} minimize & trace(\Pi C) \\ subject\ to: & \\ \Pi \in S_n & \end{array}$$

where $C = AB$.

Linear Assignment Problems

Given a cost matrix $C \in \mathbb{R}^{n\times n}$, the *Linear Assignment Problem* (LAP) is defined by:

$$\begin{array}{ll} maximize & trace(\Pi C) \\ \text{subject to:} & \\ \Pi \in S_n & \end{array}$$

Without loss of generality, max can be replace by min, for instance by solving LAP for $-C$.

Linear Assignment Problems

Given a cost matrix $C \in \mathbb{R}^{n\times n}$, the *Linear Assignment Problem* (LAP) is defined by:

$$\begin{array}{ll} \textit{maximize} & \textit{trace}(\Pi C) \\ \text{subject to:} & \\ \Pi \in S_n & \end{array}$$

Without loss of generality, max can be replace by min, for instance by solving LAP for $-C$.
The key observation is that LAP can be solved efficiently by a linear program. Specifically, the convexification of LAP produces the same optimizer:

$$\begin{array}{ll} \textit{maximize} & \textit{trace}(WC) \\ \text{subject to:} & \\ W_{i,j} \geq 0 \ , \ 1 \leq i,j \leq n & \\ \sum_{i=1}^{n} W_{i,j} = 1 \ , \ 1 \leq j \leq n & \\ \sum_{j=1}^{n} W_{i,j} = 1 \ , \ 1 \leq i \leq n & \end{array}$$

Diagonal Matrices

Another case when we know the exact solution is when A and B are diagonal matrices. Say $A = diag(a)$ and $B = diag(b)$. Then

$$trace(\Pi A \Pi^T B) = trace(diag(\Pi a) diag(b)) = trace(\Pi a b^T) = trace(\Pi C)$$

where $C = ab^T$.

Lemma

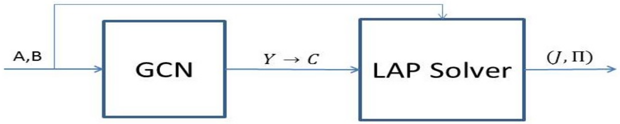
If $A = diag(a)$ and $B = diag(b)$ then the solution of the QAP is given by the solution of the LAP

$$\begin{array}{ll} maximize & trace(\Pi C) \\ subject\ to: & \\ \Pi \in S_n & \end{array}$$

where $C = ab^T$.

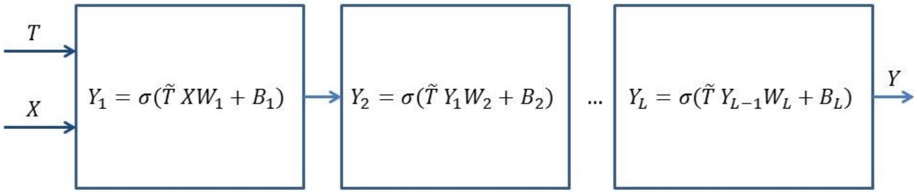
Approach

The idea is the following: First we convert the input data (A, B) into a cost matrix C, and then we solve two LAPs, one associated to C the other associated to $-C$. Finally we choose the permutation that produces the larger objective function.
The conversion step $(A, B) \mapsto C$ is performed by a Graph Convolutional Network (GCN).



Graph Convolutional Networks (GCN)

Kipf and Welling (2016) introduced a network structure that performs local processing according to a modified adjacency matrix:


T
X
$Y_1 = \sigma(\tilde{T} X W_1 + B_1)$
$Y_2 = \sigma(\tilde{T} Y_1 W_2 + B_2)$
...
$Y_L = \sigma(\tilde{T} Y_{L-1} W_L + B_L)$
Y

Here $\tilde{T} = I + T$, where T is an input adjacency matrix, or graph weight matrix. The L-layer GCN has parameters $(W_1, B_1, W_2, B_2, \cdots, W_L, B_L)$. As activation map σ we choose the ReLU (Rectified Linear Unit).

The Specific GCN Architecture

For the QAP associated to matrices (A, B) we design a specific GCN architecture:

$$X = \begin{bmatrix} A & 0 \\ B & 0 \end{bmatrix} , \quad \tilde{T} = \begin{bmatrix} I_n & \frac{1}{\|A\|_F \|B\|_F} AB \\ \frac{1}{\|A\|_F \|B\|_F} BA & I_n \end{bmatrix} \tag{3.1}$$

where the 0 matrices in X are designed to fit the appropriate size of W_1. For σ we choose the ReLU (Rectified Linear Unit) function in each layer except for the last one; in the last layer we do not use any activation function (i.e., $\sigma = Identity$). The biases $B_1, \cdots, B_L$ are chosen of the form $B_k = 1 \cdot \beta_k^T$, i.e., each row β_k^T is repeated.

GCN Guarantee

The following result applies to this network.

Theorem

Assume $A = aa^T$ and $B = bb^T$ are rank one matrices, and consider the GCN with L layers and activation map ReLU as described above. Then for any nontrivial weights $W_1, \cdots, W_L$ and biases $B_1, \cdots, B_L$ (whose rows are repeated), the network output Y partitioned $Y = \begin{bmatrix} Y^1 \\ Y^2 \end{bmatrix}$ into two blocks of n rows each, satisfies $Y^1 Y^{2T} = \gamma AB$, for some constant $\gamma \in \mathbb{R}$. In particular, the max-LAP and min-LAP applied to the latent representation matrix $C = Y^1 Y^{2T}$ are guaranteed to produce the optimal solution of the QAP.

Reference Algorithms

We compare the GCN based optimizer with two different algorithms.
1. The *AB Method* bypasses the GCN block. Thus $Y = X$ and the cost matrix inputted into the LAP solver is simply $C = AB$ (hence the name of the method). Similar to the GCN approach, the AB Method is exact on rank 1 inputs. But there is no adaptation of the cost matrix for other input matrices.
2. The *Iterative* algorithm is based on alternating max-LAP or min-LAP as follows:

$$\Pi_{k+1} \in \left\{ \begin{array}{ll} \text{argmax} & trace(\Pi A \Pi_k^T B) \\ \Pi \in S_n & \end{array} , \begin{array}{ll} \text{argmin} & trace(\Pi A \Pi_k^T B) \\ \Pi \in S_n & \end{array} \right\}$$

where $\Pi_0 = I$ (identity), and the choice of permutation at each k is based on which permutation produces a larger $trace(\Pi A \Pi^T B)$.

Comparison with Ground Truth

Results for $2 \leq n \leq 10$ and raw data normal distributed

Average relative difference w.r.t. maximum objective function:

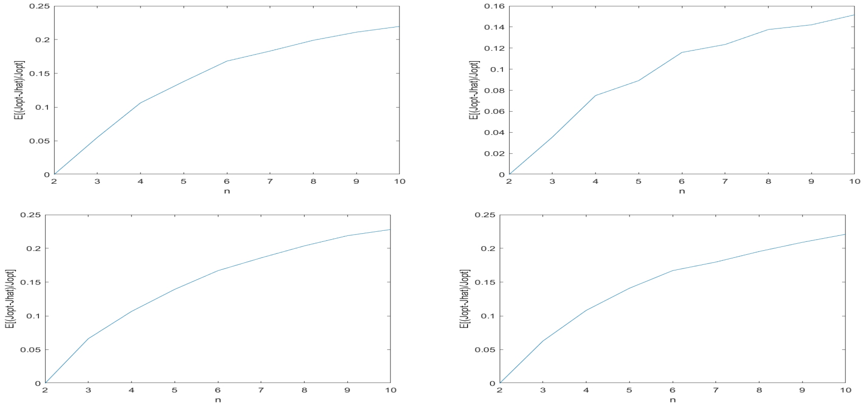


Figure: Top left: ABMethod, Top right: Iterative algorithm, Bottom left: GCN with L=2 layers and bais, Bottom right: GCN with $L = 3$ layers and bias

Comparison with Ground Truth

Results for $2 \le n \le 10$ and raw data uniform distributed

Average relative difference w.r.t. maximum objective function:

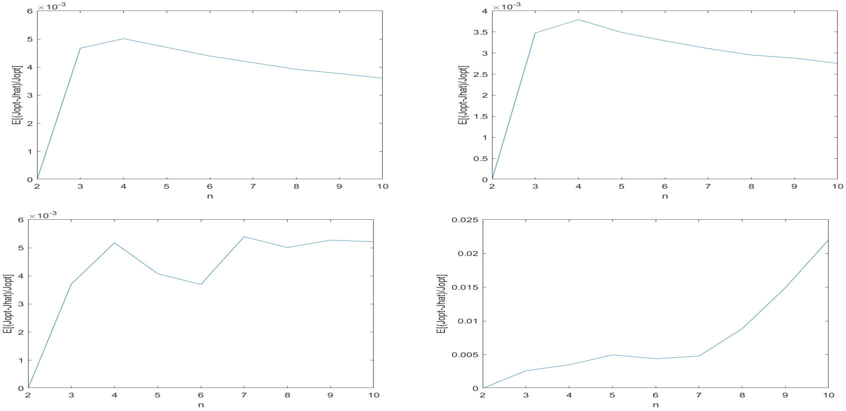


Figure: Top left: ABMethod, Top right: Iterative algorithm, Bottom left: GCN with L=2 layers and bais, Bottom right: GCN with $L = 3$ layers and bias

Relative Comparison

Results for $n = 100$ and $n = 200$ with raw data normal distributed

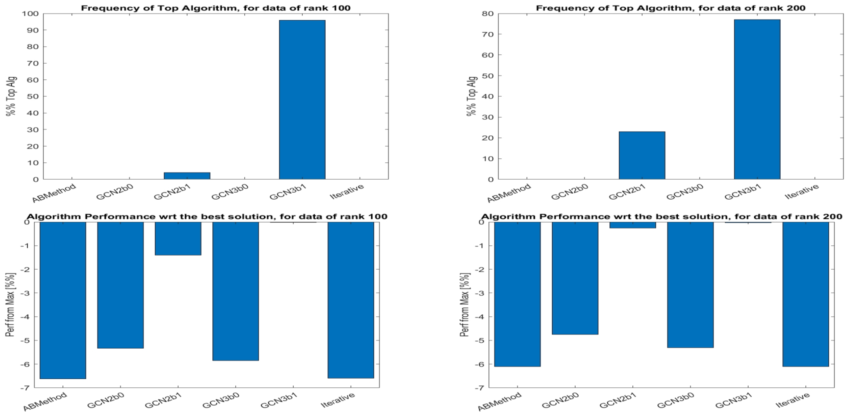


Figure: Top row: Frequency of optimal algorithm for $n = 100$ (left), and $n = 200$ (right). Borrom row: Relative performance [%] to the best algorithm for $n = 100$ (left) and $n = 200$ (right)

Relative Comparison

Results for $n = 100$ and $n = 200$ with raw data normal distributed

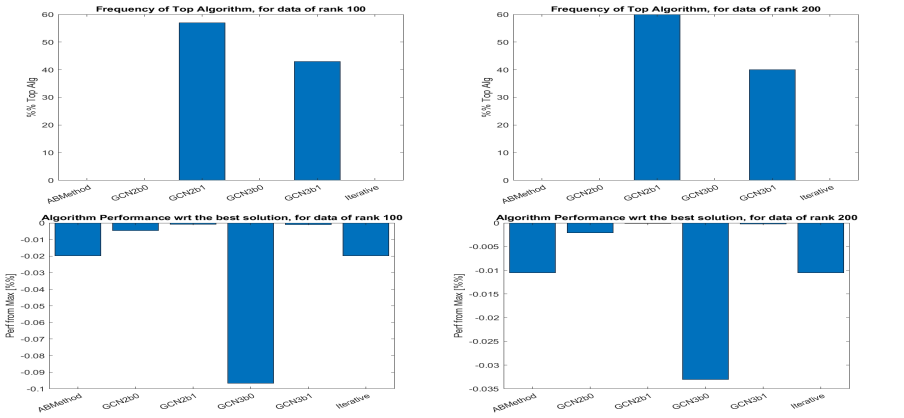


Figure: Top row: Frequency of optimal algorithm for $n = 100$ (left), and $n = 200$ (right). Borrom row: Relative performance [%] to the best algorithm for $n = 100$ (left) and $n = 200$ (right)

Conclusions

The results showed an unexpected result:

- For small n when ground truth is available, the GCN architectures performs comparable to the AB Method and in general worse than the Iterative algorithm. Among the GCN architectures, the 2 layer with bias architecture seems to have a small advantage compared to the other three GCN architectures.
- For large matrix size, the GCN algorithms consistently outperform the AB Method as well as the Iterative algorithm. However the ground truth is not available in these cases. Interestingly, for the case of uniformly $[0,1]$ case the GCN schemes with no bias provide the best objective function, whereas for the gaussian case the GCN schemes with bias provide the best objective functions. Yet, in all cases, the GCN with bias have the smallest relative difference to the largest objective value in each instance.

Thank you!

Questions?

Bibliography

[1] Vinyals, O., Fortunato, M., and Jaitly, N., Pointer Networks, arXiv e-prints , arXiv:1506.03134 (Jun 2015).
[2] Sutskever, I., Vinyals, O., and Le, Q. V., Sequence to Sequence Learning with Neural Networks, arXiv e-prints , arXiv:1409.3215 (Sep 2014).
[3] Bello, I., Pham, H., Le, Q. V., Norouzi, M., and Bengio, S., Neural Combinatorial Optimization with Reinforcement Learning, arXiv e-prints , arXiv:1611.09940 (Nov 2016).
[4] Williams, R. J., Simple statistical gradient-following algorithms for connectionist reinforcement learning, Machine learning 8(3-4), 229-256 (1992).
[5] Kool, W., van Hoof, H., and Welling, M., Attention, Learn to Solve Routing Problems, arXiv e-prints , arXiv:1803.08475 (Mar 2018).

Bibliography

[6] Dai, H., Khalil, E. B., Zhang, Y., Dilkina, B., and Song, L., Learning Combinatorial Optimization Algorithms over Graphs, arXiv e-prints , arXiv:1704.01665 (Apr 2017).
[7] Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., et al., Human-level control through deep reinforcement learning, Nature 518(7540), 529 (2015).
[8] Dai, H., Dai, B., and Song, L., Discriminative embeddings of latent variable models for structured data, in International conference on machine learning, 2702-2711 (2016).
[9] Nowak, A., Villar, S., Bandeira, A. S., and Bruna, J., Revised Note on Learning Algorithms for Quadratic Assignment with Graph Neural Networks, arXiv e-prints , arXiv:1706.07450 (Jun 2017).

Bibliography

[10] Scarselli, F., Gori, M., Tsoi, A. C., Hagenbuchner, M., and Monfardini, G., The graph neural network model, IEEE Transactions on Neural Networks 20(1), 61-80 (2008).
[11] Li, Z., Chen, Q., and Koltun, V., Combinatorial Optimization with Graph Convolutional Networks and Guided Tree Search, arXiv e-prints , arXiv:1810.10659 (Oct 2018).
[12] Kipf, T. N. and Welling, M., Semi-Supervised Classification with Graph Convolutional Networks, arXiv e-prints , arXiv:1609.02907 (Sep 2016).
[13] Kingma, D. P. and Ba, J., Adam: A Method for Stochastic Optimization, arXiv e-prints , arXiv:1412.6980 (Dec 2014).