PROCEEDINGS OF SPIE

SPIEDigitalLibrary.org/conference-proceedings-of-spie

Discrete optimizations using graph convolutional networks

Balan, Radu, Haghani, Naveed

Radu Balan, Naveed Haghani, "Discrete optimizations using graph convolutional networks," Proc. SPIE 11138, Wavelets and Sparsity XVIII, 1113806 (9 September 2019); doi: 10.1117/12.2529432

SPIE.

Event: SPIE Optical Engineering + Applications, 2019, San Diego, California, United States

Downloaded From: https://www.spiedigitallibrary.org/conference-proceedings-of-spie on 16 Feb 2020 Terms of Use: https://www.spiedigitallibrary.org/terms-of-use

Discrete Optimizations using Graph Convolutional Networks

Radu Balan[a] and Naveed Haghani[a]

[a]Department of Mathematics, University of Maryland, College Park, MD 20740

ABSTRACT

In this paper we discuss the use of graph deep learning in solving quadratic assignment problems (QAP). The quadratic assignment problem is an NP hard optimization problem. We shall analyze an approach using Graph Convolutional Networks (GCN). We prove that a specially designed GCN produces the optimal solution for a broad class of assignment problems. By appropriate training, the class of problems correctly solved is thus enlarged. Numerical examples compare this method with other simpler methods.

1. INTRODUCTION

Discrete optimization problems have been the subject of study for a long time. Many of them are NP hard, but some are shown to be solved in a polynomial time. Two such discrete optimization problems are the Linear Assignment Problems (LAP) and the Quadratic Assignment Problems (QAP). Linear Assignment Problems are known to be solved by Linear Programs, and hence they admit an exact convex relaxation. On the other hand, Quadratic Assignment Problems are much harder to solve. In fact a polynomial solution to QAPs would imply a polynomial complexity solution to the graph isomorphism problem (one of the remaining Millenium problems).

Consider three matrices $A, B, C \in \mathbb{R}^{n\times n}$ where $A = A^T$ and $B = B^T$ are symmetric. Denote by S_n the group of permutation matrices, i.e., $\Pi \in S_n$ if $\Pi^{-1} = \Pi^T$ and $\Pi_{i,j} \in \{0, 1\}$ for every $1 \le i, j \le n$. Note $S_n \subset O(n)$, where $O(n)$ denotes the group of orthogonal matrices.

The *Linear Assignment Problem* (LAP) associated to matrix C is given by the following optimization problem:

$$\begin{array}{ll}\text{maximize} & trace(\Pi C) \\ \Pi \in S_n & \end{array} . \tag{1.1}$$

We shall use $maxLAP(C)$ to denote the maximization problem above, and $minLAP(C)$ to denote the minimization problem. Note that LAP associated to $-C$ produces a solution to the corresponding minimization problem: $minLAP(C) = -maxLAP(-C)$ and $argminLAP(C) = argmaxLAP(-C)$.

The *Quadratic Assignment Problem* (QAP) associated to the pair (A, B) is given by the following optimization problem:

$$\begin{array}{ll}\text{maximize} & trace(\Pi A \Pi^T B) \\ \Pi \in S_n & \end{array} . \tag{1.2}$$

The main object of this paper is to present a novel approach to solving QAPs based on graph deep learning.

The LAP is shown to be equivalent to its convex relaxation, where S_n is replaced by its convex hull, namely the set of doubly stochastic matrices. Thus LAPs can be efficiently solved by Linear Programs.

Both LAP and QAP arise from other metric problems:

1. The closest permutation matrix Π to C^T w.r.t. Frobenious norm, $min_{\Pi\in S_n}\|C^T - \Pi\|_F$, is solved by (1.1).
2. The permutation matrix that minimizes the alignment problem $\min_{\Pi\in S_n}\|\Pi A\Pi^T - B\|_F$ is given by (1.2).

Since the optimizer in QAP (1.2) does not change if one adds a multiple of identity to A or B (the objective function does change though), without loss of generality we can assume $A, B \ge 0$. The organization of the paper is the following: in section 2 we present prior works on using deep learning in optimization; in section 3 we present our approach using Graph Convolutional Networks (GCN); in section 4 we introduce two other simple algorithms (for comparison); in section 5 we present numerical results; section 6 contains our conclusions and is followed by the bibliography.

Radu Balan: E-Mail: rvbalan@math.umd.edu
Naveed Haghani: E-Mail: nhaghan1@math.umd.edu

Wavelets and Sparsity XVIII, edited by Dimitri Van De Ville, Manos Papadakis, Yue M. Lu, Proc. of SPIE
Vol. 11138, 1113806 · © 2019 SPIE · CCC code: 0277-786X/19/$21 · doi: 10.1117/12.2529432

Downloaded From: https://www.spiedigitallibrary.org/conference-proceedings-of-spie on 16 Feb 2020
Terms of Use: https://www.spiedigitallibrary.org/terms-of-use

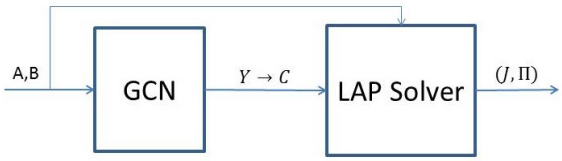

Figure 1. GCN based Approach. The GCN block can be replaced by any Graph Deep Learning (GDL) scheme, such as Message Passing Schemes

2. PRIOR WORKS

Numerous efforts have been made to leverage deep learning methodologies to address standard optimization problems. Many of these attempts have focused directly on discrete optimization. Popular among these algorithms has been Pointer Networks (Ptr-Nets).[1] Ptr-Nets utilize a sequence-to-sequence[2] Recurrent Neural Network (RNN) model to take in a series of data elements and output a series of pointers that reference directly back, in a particular order, to the inputs provided. Any problem whose solution can be represented by a partial or complete permutation of the inputs can be addressed through Ptr-Nets. This includes finding planar convex hulls, computing Delaunay triangulations, and the traveling salesman problem; all of which the addressed directly by the authors.

Other works have addressed discrete optimization problems through reinforcement learning and policy gradient schemes. Bello et al.[3] apply a Ptr-Net trained through policy gradients. They use REINFORCE[4] to evaluate their model, both in a supervised manner with predetermined solutions and in an unsupervised, iterative manner through active search. They note that combining both techniques provides the best results. Kool et al.[5] also use REINFORCE to train their model but forego the use of Ptr-Net or any recurrent neural network encoder in favor of attention layers that produce an embedding. Dai et al.[6] solve graph based combinatorial optimization problems by first producing graph embeddings for a given problem, then applying Deep Q-learning.[7] The embeddings are obtained using Structure2Vec[8] and their Q-learning agent ultimately learns a greedy policy that iteratively updates the graph embeddings during each step toward the solution.

There are also algorithms that focus on graph deep learning methods. Nowak et al.[9] employ a graph neural network[10] to solve the quadratic assignment problem, focusing mainly on graph alignment. They train a network in a supervised fashion over problem instance with predetermined solutions. Li et al.[11] use a supervised approach to solve four known NP-hard problems: satisfiability, maximal independent set, minimum vertex cover, and maximal clique. They use a GCN to produce a set of embeddings that serve as probability maps for whether a particular node is placed in the solution. Different probability maps provide different solutions that are ultimately searched through for the best result.

3. THE GCN BASED APPROACH

Our approach is based on the following observation. Assume A and B are non-negative rank one matrices, namely $A = aa^T$ and $B = bb^T$. Then a little algebra shows

$$trace(\Pi A \Pi^T B) = \left(b^T \Pi a\right)^2 = \left(trace(\Pi a b^T)\right)^2 = \frac{1}{(\langle a, b\rangle)^2} \left(trace(\Pi A B)\right)^2 \tag{3.3}$$

The QAP reduces to finding the optimum to either

$$\underset{\Pi \in S_n}{\text{maximize}} \quad trace(\Pi C) \quad \text{or} \quad \underset{\Pi \in S_n}{\text{minimize}} \quad trace(\Pi C) \tag{3.4}$$

where $C = AB$. Thus the QAP reduces to solving a LAP (maxLAP or minLAP) associated to matrix C, where $C = AB$.

In general, whether A, B are rank 1 or not, our task is to produce a latent representation C of data (A, B) so that the LAP applied to C (or $-C$) produces the optimizer of the QAP (see Figure 1).

Downloaded From: https://www.spiedigitallibrary.org/conference-proceedings-of-spie on 16 Feb 2020
Terms of Use: https://www.spiedigitallibrary.org/terms-of-use

The GCN has been introduced a few years back by Kipf and Weilling[12] and has been used many times in a myriad of applications.

Consider a graph with N nodes and adjacency (or weight) matrix T, and input feature matrix $X \in \mathbb{R}^{N\times d}$. Thus each node has a feature vector of size d corresponding to each row of X. A GCN with L layers operates recursively on this data:

$$Y_k = \sigma(\tilde{T}Y_{k-1}W_k + b_k), \quad 1 \le k \le L \tag{3.5}$$

where $Y_0 = X$, $(W_1, \ldots, W_L)$ are trainable weights matrices, $(b_1, \ldots, b_L)$ are trainable biases, and σ is the activation map acting entry-wise. $\tilde{T}$ is a modified form of the adjacency matrix T where self connections are added if not already present and each row is normalized over the degree of the corresponding node. The output of the GCN is $Y = Y_L$.

Due to the observation we mentioned earlier in regards to the case of rank 1 matrices (A, B), we design the specific components of the GCN as follows:

$$X = \begin{bmatrix} A & 0 \\ B & 0 \end{bmatrix}, \; T = \begin{bmatrix} I_n & \frac{1}{\|A\|_F\|B\|_F}AB \\ \frac{1}{\|A\|_F\|B\|_F}BA & I_n \end{bmatrix} \tag{3.6}$$

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 $b_1, \ldots, b_L$ are chosen of the form $b_k = 1 \cdot \beta_k$, i.e., each row is repeated.

The following result applies to this network.

THEOREM 3.1. *Assume $A = aa^T$ and $B = bb^T$ are rank one matrices, and $a, b \ge 0$ are nevtors with nonnegative entries. Consider the GCN with L layers and activation map ReLU as described above. Then for any nontrivial weights $W_1, \ldots, W_L$ and no biases $b_1, \ldots, b_L = 0$, the network output Y partitioned $Y = \begin{bmatrix} Y^1 \\ Y^2 \end{bmatrix}$ into two blocks of n rows each, satisfies $Y^1Y^{2T} = \gamma AB$, for some constant $\gamma \in \mathbb{R}$. In particular, either the max-LAP or the min-LAP applied to the latent representation matrix $C = Y^1Y^{2T}$ are guaranteed to produce the optimal solution.*

Proof The proof is based on computing the outputs in each layer. If the input in one layer is of the form $Input = \begin{bmatrix} au^T \\ bv^T \end{bmatrix}$, i.e., two rank 1 blocks, then the output of that layer with ReLU activation and bias as described in the theorem is of the form $Output = \begin{bmatrix} a\tilde{u}^T \\ b\tilde{v}^T \end{bmatrix}$, where $\tilde{u} = ReLU(W^T(u + \frac{\langle a,b\rangle}{\|a\|^2}v))$ and $\tilde{v} = ReLU(W^T(v + \frac{\langle b,a\rangle}{\|b\|^2}u)$. □.

4. ALTERNATIVE SIMPLER ALGORITHMS

We compare the GCN based optimizer with two different algorithms.

1. The *AB Method* uses the same framework as in Figure 1 except that the GCN block is entirely bypassed. Thus $Y = X$ and the cost matrix inputted into the LAP solver is simply $C = AB$ (hence the name of the method). Similar to Theorem 3.1, 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 maxLAP or minLAP as follows:

$$\Pi_{k+1} \in \left\{ \begin{array}{l} \underset{\Pi \in S_n}{\text{argmax}} \quad trace(\Pi A \Pi_k^T B) \\ \end{array} , \begin{array}{l} \underset{\Pi \in S_n}{\text{argmin}} \quad trace(\Pi A \Pi_k^T B) \\ \end{array} \right\} \tag{4.7}$$

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

Downloaded From: https://www.spiedigitallibrary.org/conference-proceedings-of-spie on 16 Feb 2020
Terms of Use: https://www.spiedigitallibrary.org/terms-of-use

5. NUMERICAL RESULTS

In this section we present our numerical simulations. We implemented the three algorithms described earlier, the *AB Method*, the *Iterative Algorithm*, and four different *Graph Convolutional Network* (GCN) architectures. Hence a total of six algorithms.

The GCN were trained on 50 thousand randomly generated problem instances of size $n = 10$. To maintain applicability for testing on smaller problem sizes, datasets with $n < 10$ were augmented with zero columns to match the size of the first weight matrix W_1. For each problem instance, the matrices A and B were both positive semidefinite and full rank. Each are generated as a product of a randomly generated factor matrix multiplied by its transpose, $A = UU^T, B = VV^T$. Each element of the factor matrix is a random variate drawn from the standard uniform distribution, $(U)_{i,j} \sim Uniform(0, 1)$. Training is handled through stochastic gradient descent on batch sizes of 100 using the ADAM.[13]

The Iterative algorithm (4.7) is run for 10 steps or until the difference between successive values of the objective function $trace(\Pi_k A \Pi_k^T B)$ varies by less than 10^{-6}.

For small n (i.e., $n \leq 10$) where the ground truth is available, we tested these algorithms on two types of datasets. In each case, matrices were generated as $A = UU^T$ and $B = VV^T$ where $U, V \in \mathbb{R}^{n \times r}$. For each rank rank $1 \leq r \leq n$ there are 1000 independent pairs (U, V) generated with entries uniform in $[0, 1]$, and 1000 independent pairs (U, V) with entries standard normal $N(0, 1)$.

For large n (i.e., $n = 100, 200$) where the optimal solution is unknown, we compared these algorithms with each other. For large n, we generated full rank datasets, where entries were drawn either standard normal, or uniformly in $[0, 1]$.

For small n and each problem instance (A, B) each algorithm produced an estimate $\hat{\pi}$ of the optimal permutation π_{opt}, and achieved an objective function $\hat{J}$ smaller or equal than the optimal objective function J_{opt}. The plots summarize the following statistics:

1. Relative difference of the objective $\frac{J_{opt} - \hat{J}}{\hat{J}}$.
2. The objective value rank within the first 50 largest values of the objective functions. Specifically, this is the number s so that for only s permutations $\pi_1, \ldots, \pi_s$, $J(\pi_k) \geq \hat{J}$, $1 \leq k \leq s$.
3. The frequency by which $\hat{\pi} = \pi_{opt}$.
4. The frequency that the second stage (the LAP) returns the solution associated to maximization problem of $trace(\Pi C)$.

For the testing dataset associated to a uniform distribution of entries in (U, V), the first two statistics are plotted in Figure 2-7.

For the testing dataset associated to standard normal distribution of entries in (U, V), the associated results are plotted in Figures 8-13.

For large n we generated 100 random realizations (A, B). For each realization we applied the six algorithms (the *AB Method*, the *Iterative Algorithm* and four *GCN*s) and then we compared among themselves. Specifically we estimated two statistics:

1. For each algorithm, frequency it produces the best objective value;
2. For each algorithm, the average of relative offset from the largest value. For algorithm k, $1 \leq k \leq 6$,

$$RelativePerformance_k = \frac{1}{100} \sum_{i=1}^{100} \frac{J_k(A_i, B_i) - \max_{1 \leq p \leq 6} J_p(A_i, B_i)}{\max_{1 \leq p \leq 6} J_p(A_i, B_i)}$$

Plots of these statistics when entries of U and V are drawn uniformly are included in Figures 14-16.

Plots of these statistics when entries of U and V are drawn standard normal are included in Figures 17-19.

Downloaded From: https://www.spiedigitallibrary.org/conference-proceedings-of-spie on 16 Feb 2020
Terms of Use: https://www.spiedigitallibrary.org/terms-of-use

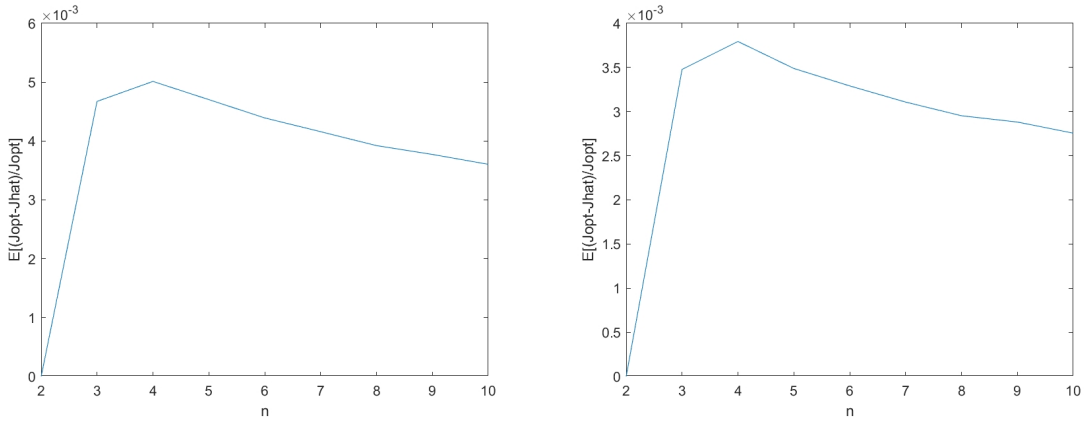

Figure 2. Relative difference of the objective $\frac{J_{opt}-\hat{J}}{\hat{J}}$: for the ABMethod (left) and Iterative Algorithm (right), for uniform datasets

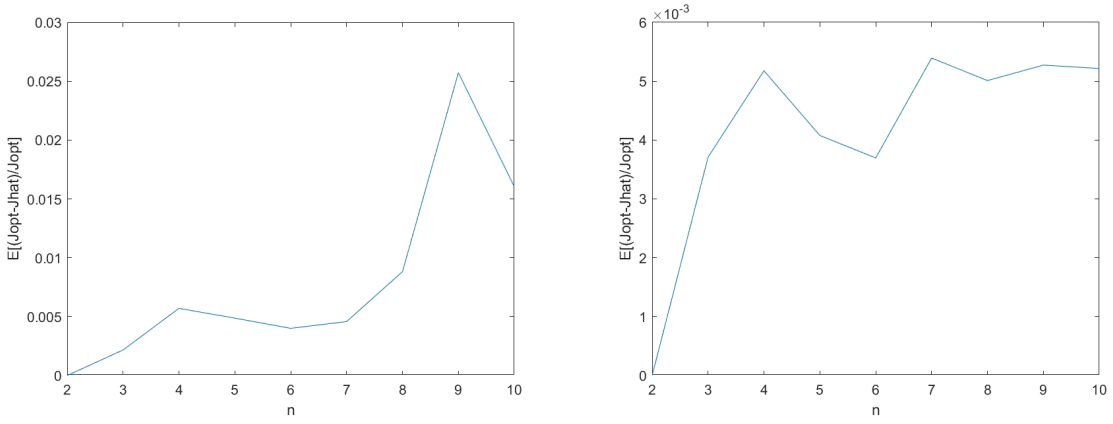

Figure 3. Relative difference of the objective $\frac{J_{opt}-\hat{J}}{\hat{J}}$: for the GCN with 2 layers and no bias (left) and GCN with 2 layers and same bias vector per node (right), for uniform datasets

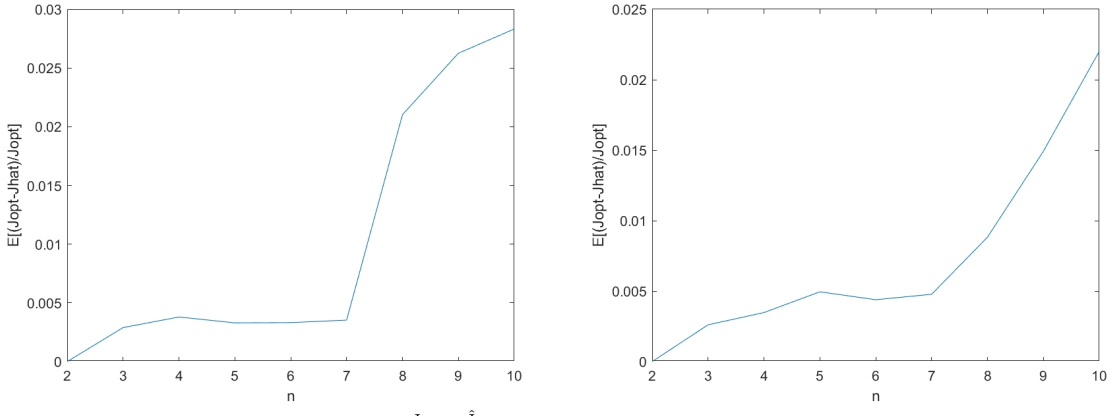

Figure 4. Relative difference of the objective $\frac{J_{opt}-\hat{J}}{\hat{J}}$: for the GCN with 3 layers and no bias (left) and GCN with 3 layers and same bias vector per node (right), for uniform datasets

6. CONCLUSIONS

In this paper we explored the use of Graph Convolutional Networks (GCN) in solving Quadratic Assignment Problems. We created a complete testing data set consisting of about 432000 files for problems ranging in size

Downloaded From: https://www.spiedigitallibrary.org/conference-proceedings-of-spie on 16 Feb 2020
Terms of Use: https://www.spiedigitallibrary.org/terms-of-use

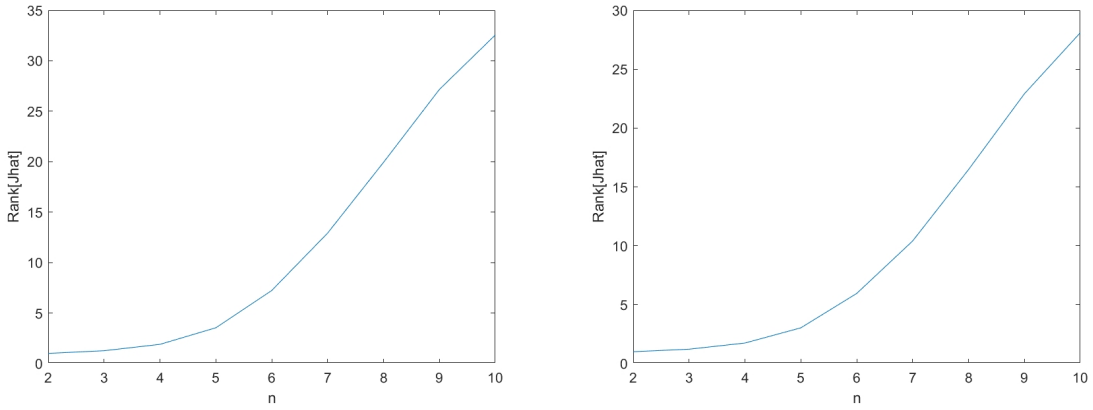

Figure 5. Rank of the objective function $\hat{J}$ out of $n!$ possible solutions: for the ABMethod (left) and Iterative Algorithm (right), for uniform datasets

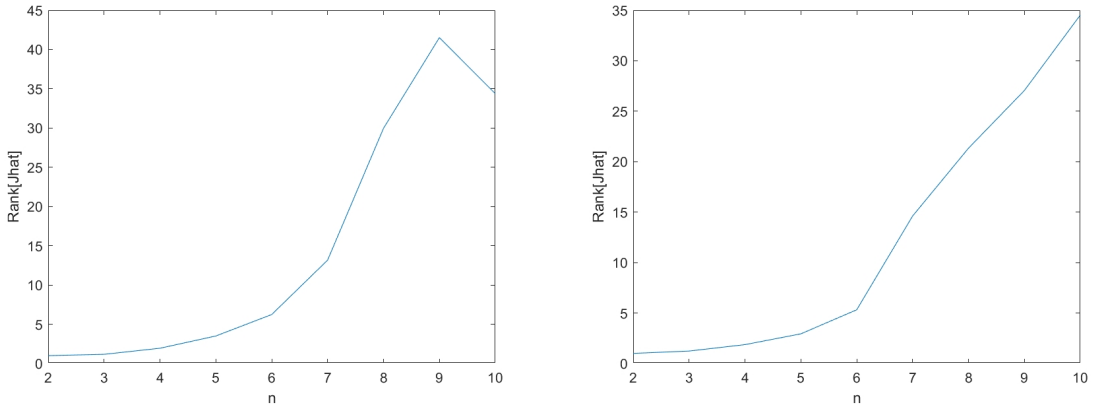

Figure 6. Rank of the objective function $\hat{J}$ out of $n!$ possible solutions: for the GCN with 2 layers and no bias (left) and GCN with 2 layers and same bias vector per node (right), for uniform datasets

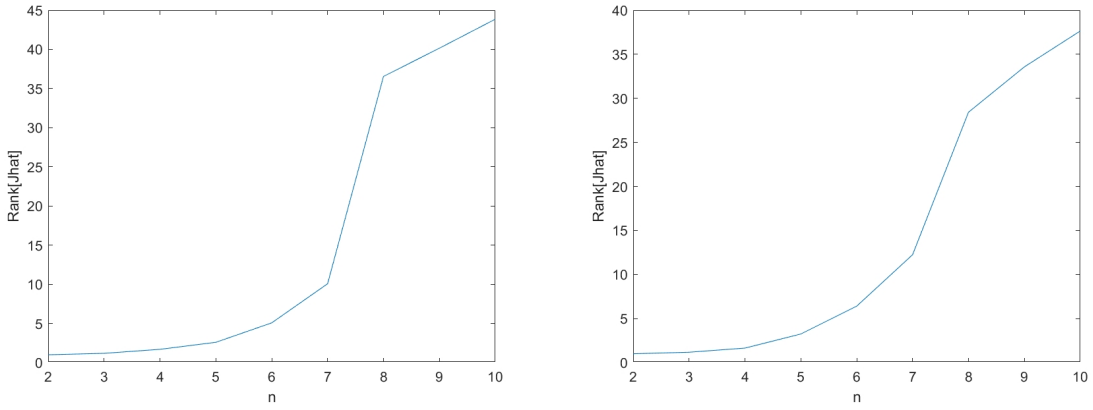

Figure 7. Rank of the objective function $\hat{J}$ out of $n!$ possible solutions: for the GCN with 3 layers and no bias (left) and GCN with 3 layers and same bias vector per node (right), for uniform datasets

from $n = 2$ to $n = 10$. The upper limit of $n = 10$ is due to Matlab memory limitations ($10! = 3628800$). Against the exact optimal solution, we tested a total of six algorithms: the AB Method, that solves a Linear Assignment Problem (LAP) instead; the Iterative Method, that iterates up to 10 LAP steps; four Graph Convolutional Network architectures: 2 layers with no bias; 2 layers with bias; 3 layers with no bias, and 3 layers with bias. The

Downloaded From: https://www.spiedigitallibrary.org/conference-proceedings-of-spie on 16 Feb 2020
Terms of Use: https://www.spiedigitallibrary.org/terms-of-use

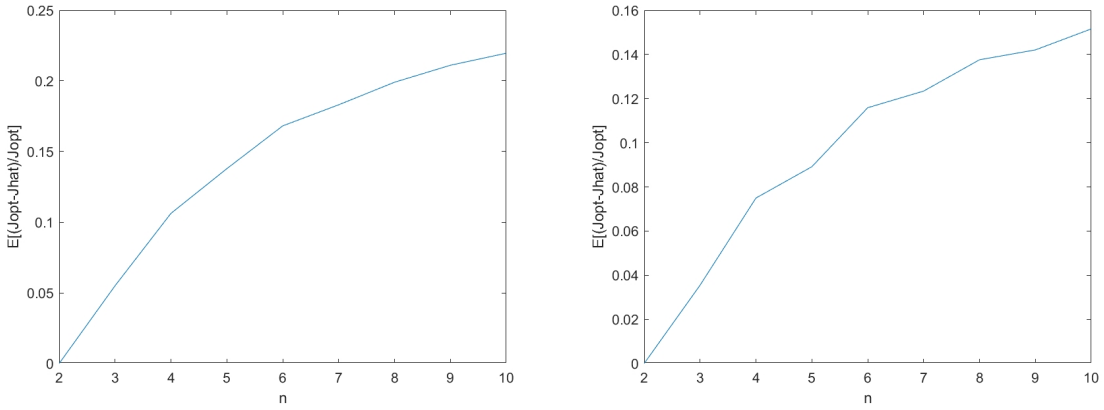

Figure 8. Relative difference of the objective $\frac{J_{opt}-\hat{J}}{\hat{J}}$: for the ABMethod (left) and Iterative Algorithm (right), for gaussian datasets

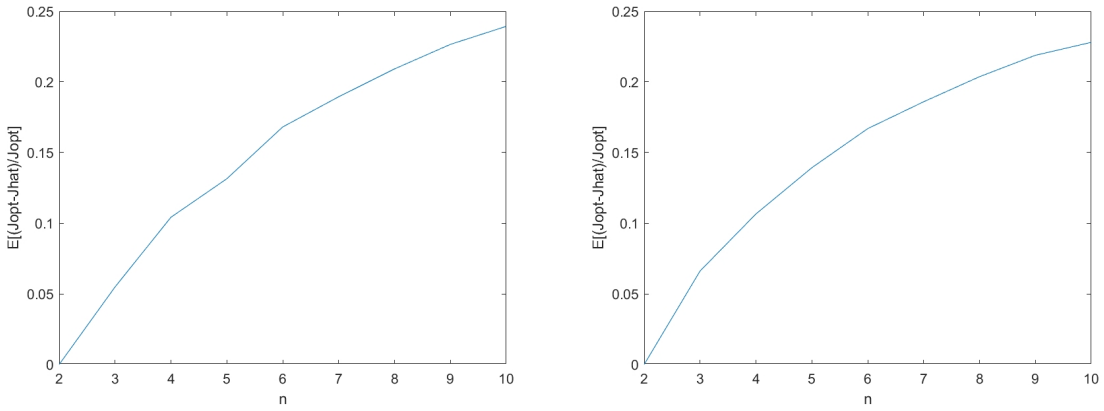

Figure 9. Relative difference of the objective $\frac{J_{opt}-\hat{J}}{\hat{J}}$: for the GCN with 2 layers and no bias (left) and GCN with 2 layers and same bias vector per node (right), for gaussian datasets

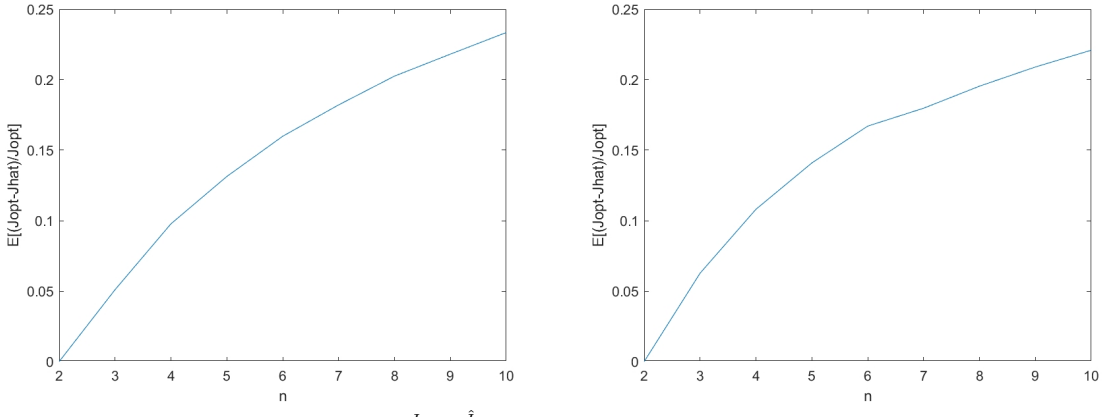

Figure 10. Relative difference of the objective $\frac{J_{opt}-\hat{J}}{\hat{J}}$: for the GCN with 3 layers and no bias (left) and GCN with 3 layers and same bias vector per node (right), for gaussian datasets

GCN architectures (the adjacency matrix and the bias structure) has been designed to guarantee exact solution for rank 1 problems. Indeed, the testing confirmed this result.

This dataset was generated using two types of distributions: one half of the dataset used uniform distribution

Downloaded From: https://www.spiedigitallibrary.org/conference-proceedings-of-spie on 16 Feb 2020
Terms of Use: https://www.spiedigitallibrary.org/terms-of-use

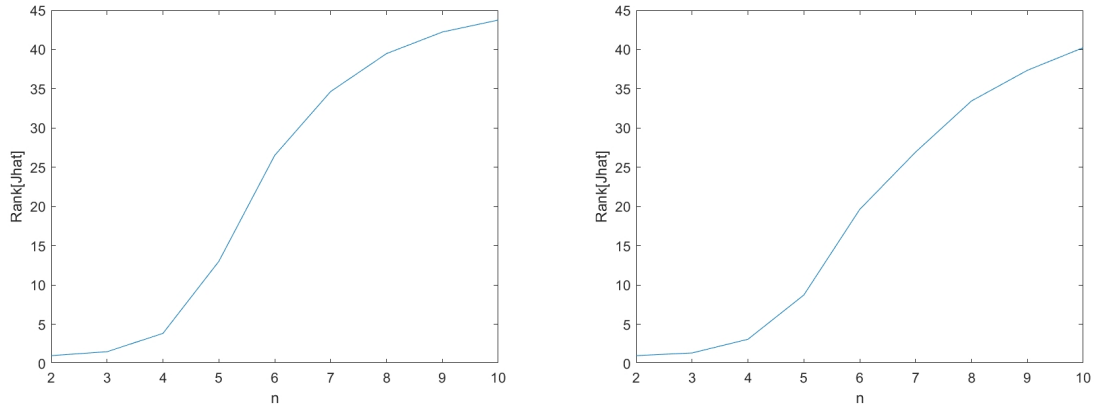

Figure 11. Rank of the objective function $\hat{J}$ out of $n!$ possible solutions: for the ABMethod (left) and Iterative Algorithm (right), for gaussian datasets

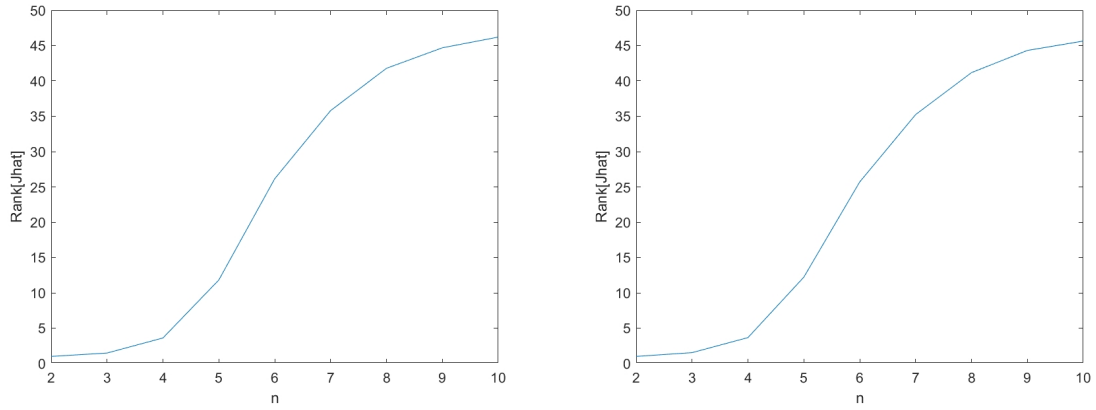

Figure 12. Rank of the objective function $\hat{J}$ out of $n!$ possible solutions: for the GCN with 2 layers and no bias (left) and GCN with 2 layers and same bias vector per node (right), for gaussian datasets

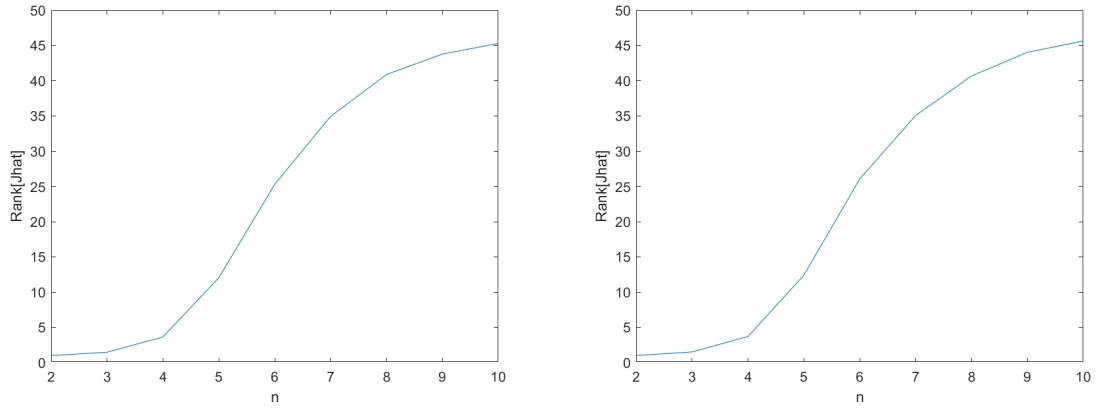

Figure 13. Rank of the objective function $\hat{J}$ out of $n!$ possible solutions: for the GCN with 3 layers and no bias (left) and GCN with 3 layers and same bias vector per node (right), for gaussian datasets

for the entries of the factor matrices generating the problems; the other half was generated using the standard normal distribution.

The results showed an unexpected result: the GCN architectures performed comparable to the AB Method

Downloaded From: https://www.spiedigitallibrary.org/conference-proceedings-of-spie on 16 Feb 2020
Terms of Use: https://www.spiedigitallibrary.org/terms-of-use

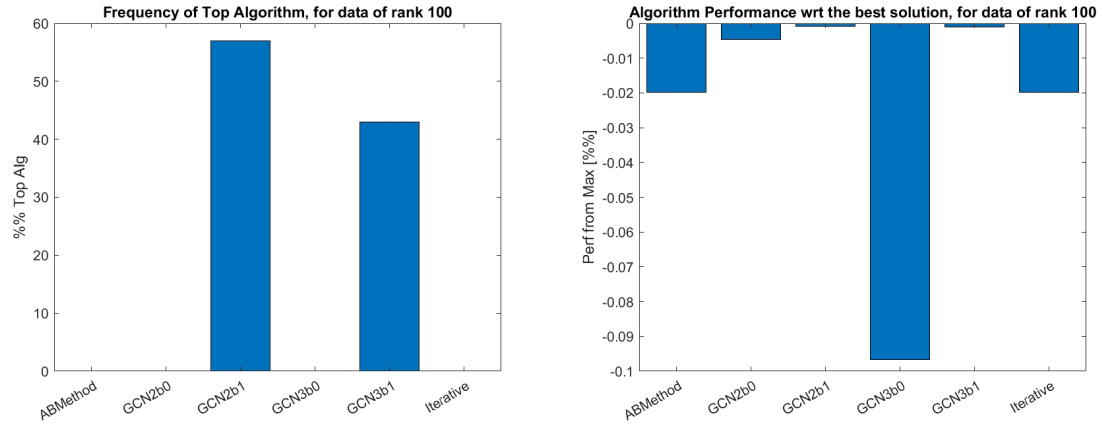

Figure 14. Frequency of Top Algorithm (left plot) and relative difference to the maximum objective (right plot) for six algorithms: AB Method, Iterative Algorithms, and four GCNs. Matrices of size $n = 100$. Entries of U, V are drawn randomly from the uniform distribution on $[0, 1]$

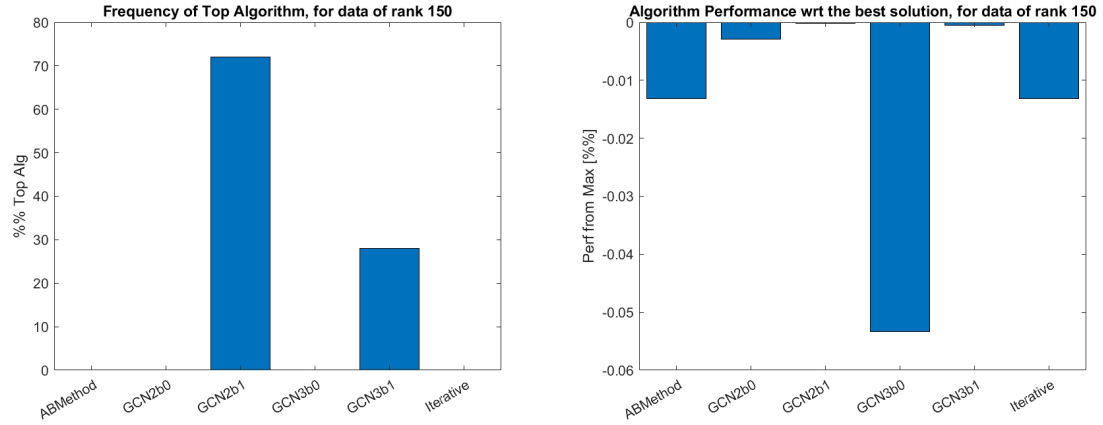

Figure 15. Frequency of Top Algorithm (left plot) and relative difference to the maximum objective (right plot) for six algorithms: AB Method, Iterative Algorithms, and four GCNs. Matrices of size $n = 150$. Entries of U, V are drawn randomly from the uniform distribution on $[0, 1]$

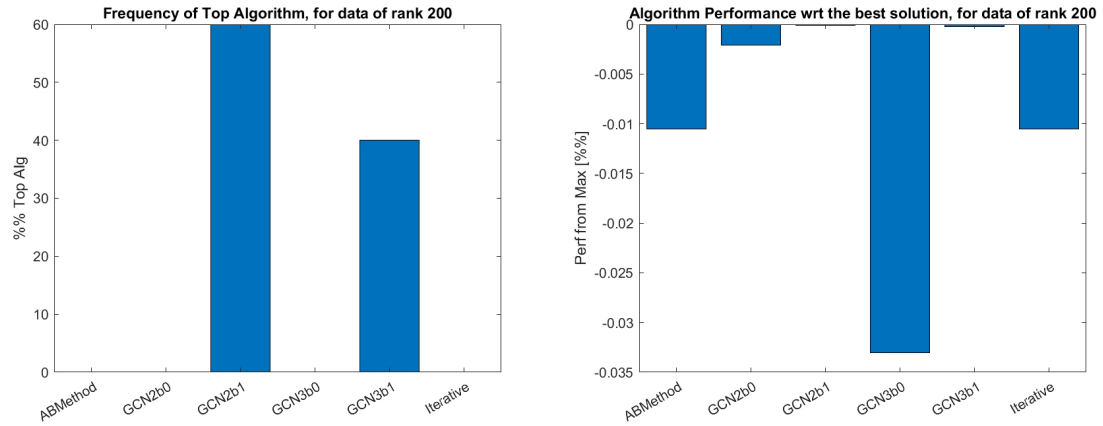

Figure 16. Frequency of Top Algorithm (left plot) and relative difference to the maximum objective (right plot) for six algorithms: AB Method, Iterative Algorithms, and four GCNs. Matrices of size $n = 200$. Entries of U, V are drawn randomly from the uniform distribution on $[0, 1]$

Downloaded From: https://www.spiedigitallibrary.org/conference-proceedings-of-spie on 16 Feb 2020
Terms of Use: https://www.spiedigitallibrary.org/terms-of-use

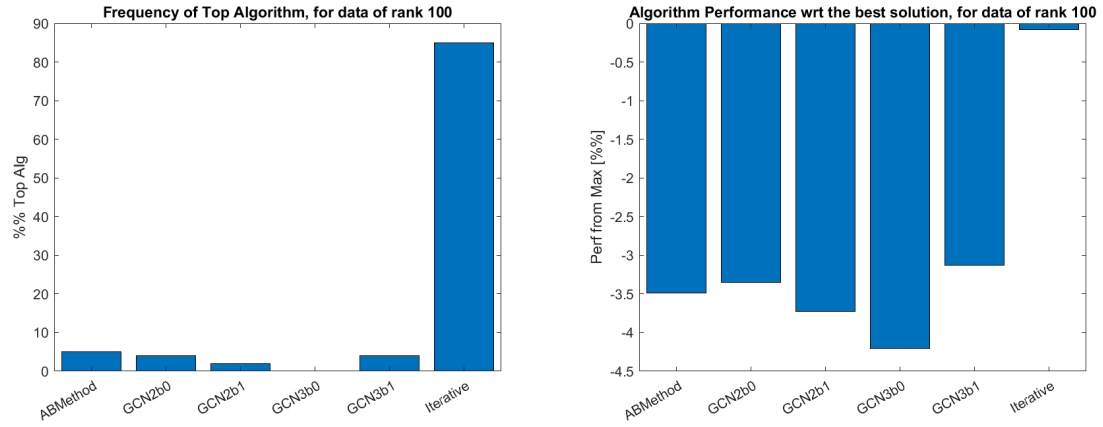

Figure 17. Frequency of Top Algorithm (left plot) and relative difference to the maximum objective (right plot) for six algorithms: AB Method, Iterative Algorithms, and four GCNs. Matrices of size $n = 100$. Entries of U, V are drawn randomly from the standard normal distribution

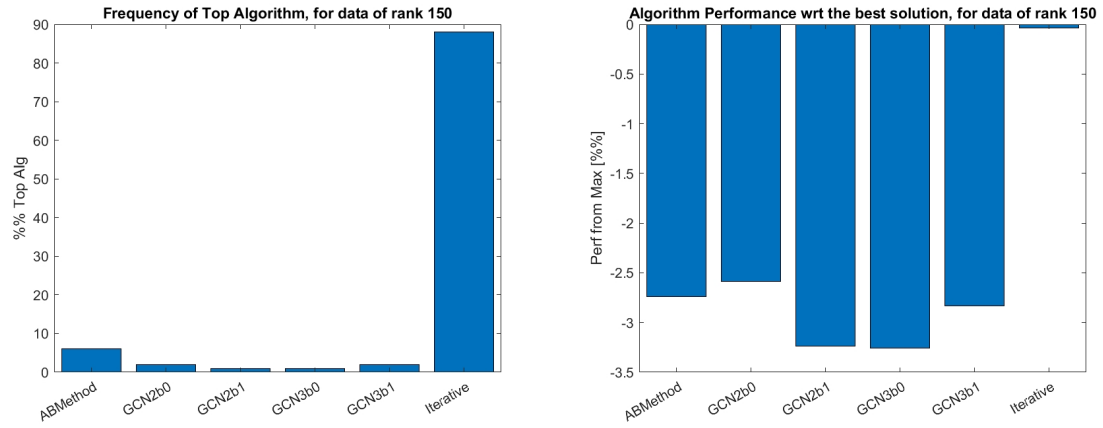

Figure 18. Frequency of Top Algorithm (left plot) and relative difference to the maximum objective (right plot) for six algorithms: AB Method, Iterative Algorithms, and four GCNs. Matrices of size $n = 150$. Entries of U, V are drawn randomly from the standard normal distribution

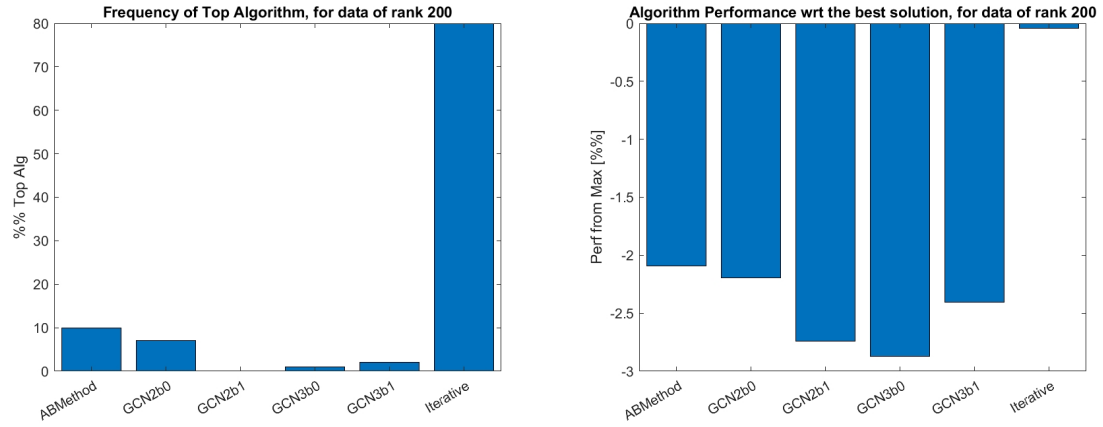

Figure 19. Frequency of Top Algorithm (left plot) and relative difference to the maximum objective (right plot) for six algorithms: AB Method, Iterative Algorithms, and four GCNs. Matrices of size $n = 200$. Entries of U, V are drawn randomly from the standard normal distribution

Downloaded From: https://www.spiedigitallibrary.org/conference-proceedings-of-spie on 16 Feb 2020
Terms of Use: https://www.spiedigitallibrary.org/terms-of-use

and in general worse than the Iterative algorithm. The dataset associated to uniform distribution was easier than the gaussian dataset as all algorithms had a better performance. This is consistent with the intuition: the matrices A, B in the uniform dataset have all nonnegative entries; in the gaussian dataset, half of the entries are likely negative. Among the GCN architectures, the 2 layer with bias architecture seems to have a small advantage compared to the other three architectures.

For large matrix size, we compared the algorithms among themselves and for $n = 100, 150, 200$ we consistently obtained that the Iterative algorithm is comparable or outperforms the AB Method as well as the GCN algorithms. However the ground truth is not available in these cases. Interestingly, for the case of uniformly $[0, 1]$ case the GCN schemes with bias provide the best objective function, albeit with very small gain, whereas in the gaussian case the Iterative algorithm provides the best objective value, by about $2 - 4\%$. The relative variation between these algorithms is less than 0.03% in average from the largest objective value when entries in the U, V matrices are drawn from the uniform $[0, 1]$ distribution; but the difference increases up to 4% when entries in U, V are drawn from the standard normal distribution.

ACKNOWLEDGMENTS

The authors have been partially supported by the NSF DMS-1816608, and LTS under grant H9823013D00560049.They also acknowledge and thank for fruitful discussions with Maneesh Singh and Julian Yarkoni (both from Verisk).

REFERENCES

[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).

[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).

[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).

Downloaded From: https://www.spiedigitallibrary.org/conference-proceedings-of-spie on 16 Feb 2020
Terms of Use: https://www.spiedigitallibrary.org/terms-of-use