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

Alex Nowak *1 Soledad Villar *1 Afonso S. Bandeira 1 Joan Bruna 1

Abstract

Many inverse problems are formulated as optimization problems over certain appropriate input distributions. Recently, there has been a growing interest in understanding the computational hardness of these optimization problems, not only in the worst case, but in an average-complexity sense under this same input distribution.

In this note, we are interested in studying another aspect of hardness, related to the ability to learn how to solve a problem by simply observing a collection of previously solved instances. These are used to supervise the training of an appropriate predictive model that parametrizes a broad class of algorithms, with the hope that the resulting “algorithm” will provide good accuracy-complexity tradeoffs in the average sense.

We illustrate this setup on the Quadratic Assignment Problem, a fundamental problem in Network Science. We observe that data-driven models based on Graph Neural Networks offer intriguingly good performance, even in regimes where standard relaxation based techniques appear to suffer.

1. Introduction

Many tasks, spanning from discrete geometry to statistics, are defined in terms of computationally hard optimization problems. Loosely speaking, computational hardness appears when the algorithms to compute the optimum solution scale poorly with the problem size, say faster than any polynomial. For instance, in high-dimensional statistics we may be interested in the task of estimating a given object from noisy measurements under a certain generative model. In that case, the notion of hardness contains both a statistical aspect, that asks above which signal-to-noise ratios the estimation is possible, and a computational aspect, that restricts the estimation to be computed in polynomial time. An active research area in Theoretical Computer Science and Statistics is to understand the interplay between those statistical and computational detection thresholds; see [1] and references therein for an instance of this program in the community detection problem, or [3; 8; 5] for examples of statistical inference tradeoffs under computational constraints.

Instead of investigating a designed algorithm for the problem in question, we consider a data-driven approach to learn algorithms from solved instances of the problem. In other words, given a collection \((x_i, y_i)\) of problem instances drawn from a certain distribution, we ask whether one can learn an algorithm that achieves good accuracy at solving new instances of the same problem – also being drawn from the same distribution, and to what extent the resulting algorithm can reach those statistical/computational thresholds.

The general approach is to cast an ensemble of algorithms as neural networks \(\hat{y} = \Phi(x; \theta)\) with specific architectures that encode prior knowledge on the algorithmic class, parameterized by \(\theta \in \mathbb{R}^S\). The network is trained to minimize the empirical loss \(L(\theta) = L^{-1} \sum_i \ell(y_i, \Phi(x_i; \theta))\), for a given measure of error \(\ell\), using stochastic gradient descent. This leads to yet another notion of learnability hardness, that measures to what extent the problem can be solved with no prior knowledge of the specific algorithm to solve it, but only a vague idea of which operations it should involve.

In this note we focus on a particular NP-hard problem, the Quadratic Assignment Problem (QAP), and study data-driven approximations to solve it. Since the problem is naturally formulated in terms of graphs, a reasonable neural network model to consider is the so-called Graph Neural Network (GNN) model [27]. This neural network alternates between applying linear combinations of local graph operators – such as the graph adjacency or the graph Laplacian, and pointwise non-linearities, and has the ability to model some forms of non-linear message passing and spectral analysis, as illustrated for instance in the data-driven Community Detection methods in the Stochastic Block Model [7]. Existing tractable algorithms for the QAP include spectral alignment methods [29] and methods based on semidefinite programming relaxations [32; 13]. Our preliminary
experiments suggest that the GNN approach taken here may be able to outperform the spectral and SDP counterparts on certain random graph models, at a lower computational budget.

We devote special attention to the Travelling Salesman Problem (TSP), a particularly important instance of the QAP, and train our model to approximately solve it on small problem instances, showing promising results. This is a particularly good problem to investigate in this setting because there exists a remarkable collection of realistic problem instances and very effective heuristics to solve them that can be used to supervise the learning of our algorithms; see Section 4.

The rest of the paper is structured as follows. Section 2 presents the problem set-up and describes existing relaxations of the QAP. Section 3 describes the graph neural network architecture and Section 4 presents our numerical experiments. Finally, Section 5 describes some open research directions motivated by our initial findings.

2. Quadratic Assignment Problem

QAP is a classical problem in combinatorial optimization. For $A, B \in \mathbb{R}^{n \times n}$ symmetric matrices it can be expressed as

$$\text{minimize} \quad \text{trace}(AXBX^T), \quad \text{subject to } X \in \Pi,$$

where $\Pi$ is the set of all permutation matrices of size $n \times n$. Many combinatorial optimization problems can be formulated in this way. For instance, the network alignment problem consists on given $A$ and $B$ the adjacency graph of two networks, to find the best matching between them, i.e.:

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

By expanding the square in (2) one can obtain an equivalent optimization of the form (1). Also note that the value of (2) is 0 if and only if the graphs $A$ and $B$ are isomorphic.

The traveling salesman problem (TSP) can also be formulated as a QAP. In TSP one is given a weighted graph $G$ and the question is to find the shortest path that visits every vertex of $G$ exactly once and returns to the starting vertex. This problem is equivalent to asking for the best matching between $A$, the complement of cycle graph in $n$ nodes, and $B$. The minimum bisection problem asks, given a graph $B$, to partition it in two equal sized subsets such that the number of edges across partitions is minimized. This problem is natural to consider in community detection and can be expressed as finding the best matching between $A$, a graph with two equal sized disconnected cliques, and $B$.

The quadratic assignment problem is known to be NP-hard and also hard to approximate [24]. Several methods and heuristics had been proposed to address the QAP. We refer the reader to [12] for a recent review of different methods and numerical comparison. According to the experiments performed in [12] the most accurate algorithm for recovering the best alignment between two networks in the distributions of problem instances considered below is a semidefinite programming relaxation (SDP) first proposed in [32]. However, such relaxation requires to lift the variable $X$ to an $n^2 \times n^2$ matrix and solve an SDP that becomes practically intractable for $n > 20$. The recent work in [13] has further relaxed the semidefinite formulation to reduce the complexity by a factor of $n$, and proposed an augmented Lagrangian alternative to the SDP which is significantly faster but not as accurate, and it consists of a convex optimization algorithm with $O(n^3)$ variables.

There are known examples where the SDP is not able to prove that two non-isomorphic graphs are actually not isomorphic (i.e. the SDP produces pseudo-solutions that achieve the same objective value as an isomorphism but that do not correspond to permutations [22; 30]). Such adverse example consists on highly regular graphs whose spectrum have repeated eigenvalues, so-called unfriendly graphs [2]. We find QAP to be a good case study for our investigations for two reasons. It is a problem that is known to be NP-hard but for which there are natural statistical models of inputs, such as models where one of the graphs is a relabelled small random perturbation of the other, on which the problem is believed to be tractable. On the other hand, producing algorithms capable of achieving this task for large perturbations appears to be difficult. It is worth noting that, for statistical models of this sort, when seen as inverse problems, the regimes on which the problem of recovering the original labeling is possible, impossible, or possible but potentially computationally hard are not fully understood.

3. Graph Neural Networks

The Graph Neural Network, introduced in [27] and further simplified in [20; 11; 28] is a neural network architecture based on local operators of a graph $G = (V,E)$, offering a powerful balance between expressivity and sample complexity; see [6] for a recent survey on models and applications of deep learning on graphs.

Given an input signal $F \in \mathbb{R}^{V \times d}$ on the vertices of $G$, we consider graph intrinsic linear operators that act locally on this signal: The degree operator is the linear map $D : \mathbb{R}^{V} \mapsto \mathbb{R}^{V}$ where $(DF)_i := \text{deg}(i) \cdot F_i$. The adjacency operator is the map $A : \mathbb{R}^{V} \mapsto \mathbb{R}^{V}$ where $(AF)_i := \sum_{j} A_{i,j} F_j$. Similarly, $2^j$-th powers of $A$, $A_j = \min(1, A^{2^j})$ encode $2^j$-hop neighborhoods of each node, and allow us to aggregate local information at different scales, which is useful in regular graphs. We also include the average operator $(U(F))_i := \frac{1}{|V|} \sum_j F_j$, which allows to broadcast information globally at each layer, thus giving the GNN the ability to recover average degrees, or more generally moments of local graph properties. By denoting $\mathcal{A} = \{1, D, A, A_1, \ldots, A_J, U\}$ the generator family, a GNN layer receives as input a signal $x^{(k)} \in \mathbb{R}^{V \times d_k}$ and
We consider the GNN and train it to solve random planted problems when using large number of layers, but (3) can grow as the optimization progresses, the cascade of GNN layers can become unstable to training. In order to mitigate this effect, we use spatial batch normalization [17] at each layer. The network depth is chosen to be of the order of the graph diameter, so that all nodes obtain information from the entire graph. In sparse graphs with small diameter, this architecture offers excellent scalability and computational complexity.

Cascading layers of the form (3) gives us the ability to approximate a broad family of graph inference algorithms, including some forms of spectral estimation. Indeed, power iterations are recovered by bypassing the nonlinear components and sharing the parameters across the layers. Some authors have observed [14] that GNNs are akin to message passing algorithms, although the formal connection has not been established, and is out of the scope of this note.

The choice of graph generators encodes prior information on the nature of the estimation task. For instance, in the community detection task, the choice of generators is motivated by a model from Statistical Physics, the Bethe free energy of GNN layers can become unstable to training. In order to mitigate this effect, we use spatial batch normalization [17] at each layer. The network depth is chosen to be of the order of the graph diameter, so that all nodes obtain information from the entire graph. In sparse graphs with small diameter, this architecture offers excellent scalability and computational complexity.

Experiments of the proposed data-driven model both for the matching and TSP problems 1. Models are trained using Adamax [19] with \( lr = 0.001 \) and batches of size 32. We note that the complexity of this algorithm is at most \( O(n^2) \).

4.1. Matching Erdos-Renyi Graphs

In this experiment, we consider \( G_1 \) to be a random Erdos-Renyi graph with edge density \( p_e \). The graph \( G_2 \) is a small perturbation of \( G_1 \) according to the following error model considered in [12]:

\[
G_2 = G_1 \odot (1 - Q) + (1 - G_1) \odot Q' \tag{4}
\]

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 \( \mathbb{P}(Q'_{ij} = 1) = p_{e2} \) with \( p_{e2} = p_e e^{-\frac{1}{2}} \). The choice of \( p_{e2} \) guarantees that the expected degrees of \( G_1 \) and \( G_2 \) are the same. We train a GNN with 20 layers and 20 feature maps per layer on a data set of 20k examples. We fix the input embeddings to be the degree of the corresponding node. In Figure 1 we report its performance in comparison with the SDP from [25] and the LowRankAlign method from [12].

4.2. Matching Random Regular Graphs

Regular graphs are an interesting example because they tend to be considered harder to align due to their more symmetric structure. Following the same experimental setup as in [12], \( G_1 \) is a random regular graph generated using the method from [18] and \( G_2 \) is a perturbation of \( G_1 \) according to the noise model (4). Although \( G_2 \) is in general not a regular graph, the “signal” to be matched to, \( G_1 \), is a regular graph. Figure 1 shows that in that case, the GNN is able to extract stable and distinctive features, outperforming the non-trainable alternatives. We used the same architecture as 4.1, but now, due to the constant node degree, the embeddings are initialized with the 2-hop degree.

4.3. Travelling Salesman Problem

Data-driven approaches to the TSP can be formulated in two different ways. First, one can use both the input graph and the ground truth TSP cycle to train the model to predict the ground truth. Alternatively, one can consider only the input graph and train the model to minimize the cost of the predicted cycle. The latter is more natural since it optimizes the TSP cost directly, but the cost of the predicted cycle is not differentiable w.r.t. model parameters. Some authors have used reinforcement learning techniques to address this issue [10], [4]. We show promising empirical results of the first method. Given a graph \( G \) and cycle \( C \), the loss is defined as \( \ell(G, C, \theta) = D_{KL}(\text{softmax}(\mathcal{E} \mathcal{E}^\top - \eta I) \parallel \frac{1}{2} A_G) \), where \( \mathcal{E} \) is the normalized embedding of \( G \) given by the GNN, \( A_G \) is the adjacency matrix of the ground truth cycle, and

1Code available at https://github.com/alexnowakvila/QAP-pt
If one believes that a problem is computationally hard for most instances in a certain regime, then this would mean that no choice of parameters for the GNN could give a good algorithm. However, even when there exist efficient algorithms to solve the problem, it does not mean necessarily that an algorithm will exist that is expressible by a GNN. On top of all of this, even if such an algorithm exists, it is not clear whether it can be learned with Stochastic Gradient Descent on a loss function that simply compares with known solved instances. However, experiments in [7] suggest that GNNs are capable of learning algorithms for community detection under the SBM essentially up to optimal thresholds, when the number of communities is small. We believe that gaining theoretical understanding of this approach to this problem (even for two communities) is a fascinating direction of research. The authors also plan to make a thorough empirical investigation of its performance for a large number of communities (where the information theoretic and computational thresholds are suspected to differ).

The performance of these algorithms depends on which operators are used in the GNN. Adjacency matrices and Laplacians are natural choices for the types of problem we considered, but different problems may require different sets of operators. A natural question is to find a principled way of choosing the operators. Going back to QAP, it would be interesting to understand the limits of this problem, both statistically [23], but also computationally. In particular the authors would like to better understand the limits of the GNN approach and more generally of any approach that first embeds the graphs, and then does linear assignment.

In general, understanding whether the regimes for which GNNs produce meaningful algorithms matches believed computational thresholds for some classes of problems is, in our opinion, a thrilling research direction. It is worth noting that this approach has the advantage that the algorithms are learned automatically. However, they may not generalize in the sense that if the GNN is trained with examples below a certain input size, it is not clear that it will be able to interpret much larger inputs, that may need larger networks.
Acknowledgements

ASB was partially supported by NSF DMS-1712730 and NSF DMS-1719545. SV was partially supported by the Simons Algorithms and Geometry (A&G) Think Tank.

References


