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 $(x_i, y_i)_{i \leq L}$ 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 $\mathcal{L}(\theta) = L^{-1} \sum_i \ell(y_i, \Phi(x_i; \theta))$, for a given measure of error ℓ, 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

[*]Equal contribution [1]Courant Institute of Mathematical Sciences and Center for Data Science, NYU, New York, USA. Correspondence to: Joan Bruna <bruna@cims.nyu.edu>.

Proceedings of the 34^{th} International Conference on Machine Learning, Sydney, Australia, PMLR 70, 2017.

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 $n \times n$ symmetric matrices it can be expressed as

$$\text{minimize } \operatorname{trace}(AXBX^\top), \quad \text{subject to } X \in \Pi, \tag{1}$$

where Π 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.:

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

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 B and the question is to find the shortest path that visits every vertex of B 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 $D : F \mapsto DF$ where $(DF)_i := deg(i) \cdot F_i$, $D(F) = \operatorname{diag}(A\mathbf{1})F$. The *adjacency operator* is the map $A : F \mapsto A(F)$ where $(AF)_i := \sum_{j \sim i} F_j$, with $i \sim j$ iff $(i, j) \in E$. 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 $\mathcal{A} = \{\mathbf{1}, D, A, A_1 \dots, A_J, U\}$ the *generator* family, a GNN layer receives as input a signal $x^{(k)} \in \mathbb{R}^{V \times d_k}$ and

produces $x^{(k+1)} \in \mathbb{R}^{V \times d_{k+1}}$ as

$$x_l^{(k+1)} = \rho\left(\sum_{B \in \mathcal{A}} B x^{(k)} \theta_{B,l}^{(k)}\right), \; l = 1 \dots \frac{d_{k+1}}{2}, \quad (3)$$

$$x_l^{(k+1)} = \sum_{B \in \mathcal{A}} B x^{(k)} \theta_{B,l}^{(k)}, \; l = \frac{d_{k+1}}{2} + 1, \dots d_{k+1},$$

where $\Theta = \{\theta_1^{(k)}, \dots, \theta_{|\mathcal{A}|}^{(k)}\}_k$, $\theta_B^{(k)} \in \mathbb{R}^{d_k \times d_{k+1}}$, are trainable parameters and $\rho(\cdot)$ is a point-wise non-linearity, chosen in this work to be $\rho(z) = \max(0, z)$. We thus consider a layer with linear "residual connections" [15], to both ease with the optimization when using large number of layers, but also to give the model the ability to perform power iterations. Since the spectral radius of the learned linear operators in (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 [26; 7]. In the QAP, one needs generators that are able to detect distinctive and stable local structures. Multiplicity of the spectrum of the graph adjacency operator is related to the (un)effectiveness of certain relaxations [2; 21] (the so-called *(un)friendly graphs*), suggesting that generator families $\mathcal{A}$ that contain non-commutative operators may be more robust on such examples.

4. Numerical Experiments

We consider the GNN and train it to solve random planted problem instances of the QAP. Given a pair of graphs G_1, G_2 with n nodes each, we consider a *siamese* GNN encoder producing normalized embeddings $E_1, E_2 \in \mathbb{R}^{n \times d}$. Those embeddings are used to predict a matching as follows. We first compute the outer product $E_1 E_2^T$, that we then map to a stochastic matrix by taking the softmax along each row/column. Finally, we use standard cross-entropy loss to predict the corresponding permutation index. We perform 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]:

$$G_2 = G_1 \odot (1 - Q) + (1 - G_1) \odot Q' \quad (4)$$

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 $\mathbb{P}(Q'_{ij} = 1) = p_{e_2}$ with $p_{e_2} = p_e \frac{p}{p-1}$. The choice of p_{e_2} 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 $\ell(G, C, \theta) = D_{KL}(\text{softmax}(\tilde{E}\tilde{E}^\top - \eta I) \parallel \frac{1}{2} A_C)$, where $\tilde{E}$ is the normalized embedding of G given by the GNN, A_C is the adjacency matrix of the ground truth cycle, and

[1] Code available at https://github.com/alexnowakvila/QAP_pt

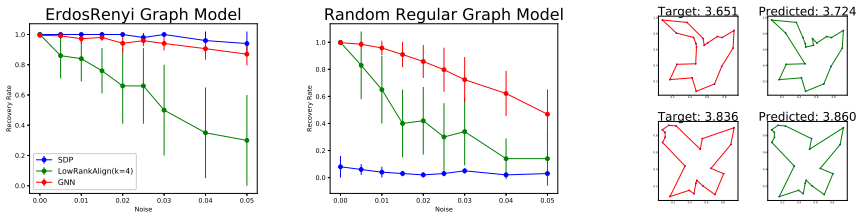

Figure 1. Comparison of recovery rates for the SDP [25], LowRankAlign [12] and our data-driven GNN, for the Erdos-Renyi model (left) and random regular graphs (middle). All graphs have $n = 50$ nodes and edge density $p = 0.2$. The recovery rate is measured as the average number of matched nodes from the ground truth. Experiments have been repeated 100 times for every noise level except the SDP, which have been repeated 5 times due to its high computational complexity. (right) Examples of ground truth solution of the TSP problem followed by the solution found by the GNN. Note that the paths may be quite different but the costs are very close.

η is a large constant to avoid the diagonal's influence. We address the metric TSP, which is an instance of the more general graph TSP (but still NP-Hard). We generated 20k training examples and tested on 1k other instances. Each one generated by uniformly sampling $\{x_i\}_{i=1}^{20} \in [0,1]^2$. We build a complete graph with $A_{i,j} = d_{\max} - d_2(x_i, x_j)$ as weights. The ground truth cycles are generated with [16], which has an efficient implementation of the Lin-Kernighan TSP Heuristic. The architecture has 40 layers and 80 feature maps per layer. The predicted cycles are generated with a beam search strategy with beam size of 40. The resulting approximation ratio [2] over the test set is **1.027**, slightly worse than the auto-regressive data-driven model [31] (1.013) and Christofides [9] (1.010). Although we hypothesize that the performance gap with PtrNets [31] may come from architectural issues, the big limitation of the model compared to heuristics algorithms is the current data driven approach, namely, the need for expensive ground truth examples, and more importantly, the fact that the model is imitating an heuristic rather than directly optimizing the TSP cost.

5. Discussion

Problems are often labeled to be as hard as their hardest instance. However, many hard problems can be efficiently solved for a large subset of inputs. This note attempts to learn an algorithm for QAP by learning from solved problem instances drawn from a distribution of inputs. The algorithm's effectiveness is evaluated by investigating how well it works on new inputs from the same distribution. This can be seen as a general approach and not restricted to QAP. In fact, another notable example is the community detection problem under the Stochastic Block Model (SBM) [7]. That problem is another particularly good case of study because there exists very precise predictions for the regimes where the recovery problem is (i) impossible, (ii) possible and efficiently solvable, or (iii) believed that even though possible, may not be solvable in polynomial time.

[2] defined as $\frac{\text{average predicted cost}}{\text{average ground truth cost}}$

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

[1] Emmanuel Abbe. Community detection and stochastic block models: recent developments. *arXiv preprint arXiv:1703.10146*, 2017.

[2] Yonathan Aflalo, Alex Bronstein, and Ron Kimmel. Graph matching: relax or not? *arXiv preprint arXiv:1401.7623*, 2014.

[3] Arash A Amini and Martin J Wainwright. High-dimensional analysis of semidefinite relaxations for sparse principal components. In *Information Theory, 2008. ISIT 2008. IEEE International Symposium on*, pages 2454–2458. IEEE, 2008.

[4] Irwan Bello, Hieu Pham, Quoc V Le, Mohammad Norouzi, and Samy Bengio. Neural combinatorial optimization with reinforcement learning. *arXiv preprint arXiv:1611.09940*, 2016.

[5] Q. Berthet and P. Rigollet. Complexity theoretic lower bounds for sparse principal component detection. *Conference on Learning Theory (COLT)*, 2013.

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

[7] Joan Bruna and Xiang Li. Community detection with graph neural networks. *arXiv preprint arXiv:1705.08415*, 2017.

[8] Venkat Chandrasekaran and Michael I Jordan. Computational and statistical tradeoffs via convex relaxation. *Proceedings of the National Academy of Sciences*, 110(13):E1181–E1190, 2013.

[9] Nicos Christofides. Worst-case analysis of a new heuristic for the travelling salesman problem. Technical report, DTIC Document, 1976.

[10] Hanjun Dai, Elias B Khalil, Yuyu Zhang, Bistra Dilkina, and Le Song. Learning combinatorial optimization algorithms over graphs. *arXiv preprint arXiv:1704.01665*, 2017.

[11] David K Duvenaud, Dougal Maclaurin, Jorge Iparraguirre, Rafael Bombarell, Timothy Hirzel, Alán Aspuru-Guzik, and Ryan P Adams. Convolutional networks on graphs for learning molecular fingerprints. In *Advances in neural information processing systems*, pages 2224–2232, 2015.

[12] Soheil Feizi, Gerald Quon, Mariana Recamonde-Mendoza, Muriel Médard, Manolis Kellis, and Ali Jadbabaie. Spectral alignment of networks. *arXiv preprint arXiv:1602.04181*, 2016.

[13] Jose FS Ferreira, Yuehaw Khoo, and Amit Singer. Semidefinite programming approach for the quadratic assignment problem with a sparse graph. *arXiv preprint arXiv:1703.09339*, 2017.

[14] Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. Neural message passing for quantum chemistry. *arXiv preprint arXiv:1704.01212*, 2017.

[15] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016.

[16] Keld Helsgaun. *An effective implementation of K-opt moves for the Lin-Kernighan TSP heuristic*. PhD thesis, Roskilde University. Department of Computer Science, 2006.

[17] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

[18] Jeong Han Kim and Van H Vu. Generating random regular graphs. In *Proceedings of the thirty-fifth annual ACM symposium on Theory of computing*, pages 213–222. ACM, 2003.

[19] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[20] Yujia Li, Daniel Tarlow, Marc Brockschmidt, and Richard Zemel. Gated graph sequence neural networks. *arXiv preprint arXiv:1511.05493*, 2015.

[21] Vince Lyzinski, Donniell E Fishkind, Marcelo Fiori, Joshua T Vogelstein, Carey E Priebe, and Guillermo Sapiro. Graph matching: Relax at your own risk. *IEEE transactions on pattern analysis and machine intelligence*, 38(1):60–73, 2016.

[22] Ryan O'Donnell, John Wright, Chenggang Wu, and Yuan Zhou. Hardness of robust graph isomorphism, lasserre gaps, and asymmetry of random graphs. In *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1659–1677. SIAM, 2014.

[23] Efe Onaran, Siddharth Garg, and Elza Erkip. Optimal de-anonymization in random graphs with community structure. *arXiv preprint arXiv:1602.01409*, 2016.

[24] Panos M Pardalos, Henry Wolkowicz, et al. *Quadratic Assignment and Related Problems: DIMACS Workshop, May 20-21, 1993*, volume 16. American Mathematical Soc., 1994.

[25] 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.

[26] 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.

[27] Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. The graph neural network model. *IEEE Transactions on Neural Networks*, 20(1):61–80, 2009.

[28] Sainbayar Sukhbaatar, Rob Fergus, et al. Learning multiagent communication with backpropagation. In *Advances in Neural Information Processing Systems*, pages 2244–2252, 2016.

[29] Shinji Umeyama. An eigendecomposition approach to weighted graph matching problems. *IEEE transactions on pattern analysis and machine intelligence*, 10(5):695–703, 1988.

[30] Soledad Villar, Afonso S Bandeira, Andrew J Blumberg, and Rachel Ward. A polynomial-time relaxation of the gromov-hausdorff distance. *arXiv preprint arXiv:1610.05214*, 2016.

[31] Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. Pointer networks. In *Advances in Neural Information Processing Systems*, pages 2692–2700, 2015.

[32] Qing Zhao, Stefan E Karisch, Franz Rendl, and Henry Wolkowicz. Semidefinite programming relaxations for the quadratic assignment problem. *Journal of Combinatorial Optimization*, 2(1):71–109, 1998.