Experimental performance of graph neural networks on random instances of max-cut

Yao, Weichi, Bandeira, Afonso, Villar, Soledad

Weichi Yao, Afonso S. Bandeira, Soledad Villar, "Experimental performance of graph neural networks on random instances of max-cut," Proc. SPIE 11138, Wavelets and Sparsity XVIII, 111380S (9 September 2019); doi: 10.1117/12.2529608

SPIE.

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

Experimental performance of graph neural networks on random instances of max-cut

Weichi Yao[a], Afonso S. Bandeira[b], and Soledad Villar[b]

[a]*Stern School of Business, New York University, 44 West 4th Street, New York, USA 10011*
[b]*NYU Center for Data Science and Courant Institute of Mathematical Sciences, New York University, 60 5th Ave, New York, USA 10012*

ABSTRACT

This note explores the applicability of unsupervised machine learning techniques towards hard optimization problems on random inputs. In particular we consider Graph Neural Networks (GNNs) – a class of neural networks designed to learn functions on graphs – and we apply them to the max-cut problem on random regular graphs. We focus on the max-cut problem on random regular graphs because it is a fundamental problem that has been widely studied. In particular, even though there is no known explicit solution to compare the output of our algorithm to, we can leverage the known asymptotics of the optimal max-cut value in order to evaluate the performance of the GNNs.

In order to put the performance of the GNNs in context, we compare it with the classical semidefinite relaxation approach by Goemans and Williamson (SDP), and with extremal optimization, which is a local optimization heuristic from the statistical physics literature. The numerical results we obtain indicate that, surprisingly, Graph Neural Networks attain comparable performance to the Goemans and Williamson SDP. We also observe that extremal optimization consistently outperforms the other two methods. Furthermore, the performances of the three methods present similar patterns, that is, for sparser, and for larger graphs, the size of the found cuts are closer to the asymptotic optimal max-cut value.

1. INTRODUCTION

Consider the fundamental problem of *max-cut*, where given a graph, the task is to find a partition of the vertices into two classes such that the number of edges across classes is maximized. More precisely, given a weighted undirected graph $G = (V, E)$ with $V = \{1, ..., n\}$ and with weights $w_{ij} = w_{ji} \geq 0$, the goal of the max-cut problem is to find a partition $(S, V - S)$ which maximizes the sum of the weights of edges between S and $V - S$. One can formulate the problem as the following quadratic program:

$$\begin{aligned} \max \quad & \frac{1}{2} \sum_{i<j} w_{ij}(1 - x_i x_j) \\ s.t. \quad & x_i \in \{-1, +1\}, \ \forall \ i \in [n] \end{aligned} \tag{P}$$

where the x_i are binary variables that indicate set membership, i.e., $x_i = 1$ if $i \in S$ and $x_i = -1$ otherwise. This integer quadratic program is known to be NP-complete (Garey and Johnson, 1990).

To date, several algorithms have been developed to approximately solve the max-cut problem, and some approaches, such as the Goemans–Williamson semidefinite relaxation (Goemans and Williamson, 1995), have inspired many algorithms for a variety of problems. Since optimal solutions of max-cut on sparse graphs tend to correspond to (almost) balanced partitions, this combinatorial optimization problem is closely related to max-bisection, or min-bisection on the complement graph. In fact, there has been a recent flurry of work on variations of these problems on graph with planted structures. For example, the stochastic block model is a popular modification of the Erdős–Rényi random graph, featuring a community structure, in which each edge probability is determined by whether the two vertices reside in a common planted community. The analysis of max-cut (or min/max -bisection) under such random models of data is convenient because there is a known ground truth to evaluate against, and this feature has contributed to the popularity of this approach (Abbe, 2017).

Wavelets and Sparsity XVIII, edited by Dimitri Van De Ville, Manos Papadakis, Yue M. Lu, Proc. of SPIE
Vol. 11138, 111380S · © 2019 SPIE · CCC code: 0277-786X/19/$21 · doi: 10.1117/12.2529608

In this paper, we consider the max-cut problem for random regular graphs without planted structures. Since there are no planted communities in the underlying random graph distribution, there is no known ground truth to compare against. Instead, we leverage the well-understood asymptotics of max-cut under this distribution by comparing the max-cut objective value of a given clustering to the asymptotic optimal value (Dembo et al., 2017; Zdeborová and Boettcher, 2010). We note that this framework has fundamental ties with various disciplines, and provides a mechanism for identifying computational-to-statistical gaps (Bandeira et al., 2018). In machine learning, the absence of a planted signal allows, e.g., one to study false positives in clustering.

Throughout machine learning, it has become popular to solve problems using neural networks, and so we are particularly interested in understanding the performance of such techniques in solving unplanted optimization problems; more specifically, we study the power of *graph* neural networks (GNNs) in solving random max-cut instances. GNNs are a type of neural network that directly operates on the graph structure. They where first proposed in Gori et al. (2005); Bronstein et al. (2017), and have emerged as a powerful class of algorithms to perform complex graph inference leveraging labeled data and tackling classification problems. Recently, GNNs were used to perform community detection in the stochastic block model, where they proved to be competitive with a variety of other solvers (Chen et al., 2017). In this paper, we illustrate that GNNs also perform well in our *unsupervised* setting (despite the lack of labelled data), and we further investigate different ways to adapt the GNN framework so as to perform unsupervised learning.

For the sake of comparison, we solve our max-cut instances using GNNs, as well as the Goemans–Williamson SDP approach and extremal optimization. Extremal optimization (EO) is a local method for combinatorial optimization problems that first appeared in the statistical physics literature Boettcher and Percus (1999). It has been successfully applied to solve the graph bi-partitioning problem (Boettcher and Percus, 2001) and the traveling salesman problem (Chen et al., 2007), among other discrete optimization problems. In the case of the bi-partitioning problem on random graphs, simulations performed in Boettcher and Percus (2001) suggest that EO converges to near optimal configurations in $O(N)$ time where N is the number of nodes, but approximation guarantees for this algorithm are not currently known.

Another algorithmic approach worth mentioning is a message passing algorithm developed by Montanari for optimization of the Sherrington-Kirkpatrick Hamiltonian (Montanari, 2018). The optimization of the Sherrington-Kirkpatrick (SK) Hamiltonian can be viewed as a "gaussian" analogue of max-cut on random graphs; and it was its study that allowed for the asymptotic understanding of max-cut on random d-regular graphs (Dembo et al., 2017). This algorithm is proved (Montanari, 2018) (conditioned on a mild statistical physics conjecture) to be asymptotic optimal in the SK setting. While it would be fascinating to understand the performance of an analogue of the algorithm in Montanari (2018) to the max-cut problem in random sparse graphs, this is outside of the scope of this paper.

The paper is organized as follows. In Section 2, we describe the Goemans and Williamson algorithm, the extremal optimization algorithm, and how we adapt the Line Graph Neural Network model from Chen et al. (2017) to the max-cut problem. Section 3 shows extensive numerical results, where the performance of the three methods is evaluated and compared. In Section 4, we conclude with an outlook on future work.

2. ALGORITHMS FOR MAX-CUT PROBLEMS

2.1 The Goemans-Williamson Algorithm

In their very influential paper, Goemans and Williamson (1995) introduce an approximation algorithm for the max-cut problem based on semidefinite programming. The algorithm consists of the following steps:
(i) Relax the optimization problem (P) to the vector programming (P'):

$$\begin{aligned} \max_{\mathbf{u}} \quad & \frac{1}{2}\sum_{i<j} w_{ij}(1-\mathbf{u}_i^T\mathbf{u}_j) \\ s.t. \quad & \|\mathbf{u}_i\|^2 = 1, \text{ and } \mathbf{u}_i \in \mathbb{R}^n, \ \forall\, i \in [n] \end{aligned} \tag{P'}$$

which is equivalent to the following semidefinite program (SDP):

$$\begin{aligned} \max_{X} \quad & \frac{1}{2}\sum_{i<j} w_{ij}(1-X_{ij}) \\ s.t. \quad & X \succeq 0, \text{ and } X_{ii}=1, \ \forall\, i \in [n]. \end{aligned} \tag{SDP}$$

Solve (P') and denote its solution as $\mathbf{u}_i$, $i \in [n]$.
(ii) Let $r \in \mathbb{R}^n$ be a vector uniformly distributed on the unit sphere
(iii) Produce a cut subset $S = \{i | r^T \mathbf{u}_i \geq 0\}$.

Goemans and Williamson (1995) showed that their algorithm is able to obtain a cut whose expected value is guaranteed to be no smaller than a particular constant α_{GW} times the optimum cut:

$$\mathsf{max\text{-}cut}(G) \geq \alpha_{\mathrm{GW}} \frac{1}{2}\sum_{i<j} w_{ij}(1-\mathbf{u}_i^T\mathbf{u}_j) \geq \alpha_{\mathrm{GW}}\mathsf{max\text{-}cut}(G)$$

with $\alpha_{\mathrm{GW}} = \min_{-1\leq x\leq 1} \frac{\frac{1}{\pi}\arccos(x)}{\frac{1}{2}(1-x)} \simeq 0.878$.

2.2 Extremal Optimization

Boettcher and Percus (1999, 2000) introduced a local search procedure called extremal optimization (EO), modeled after the Bak-Sneppen mechanism (Bak and Sneppen, 1993), which was originally proposed to describe the dynamics of co-evolving species.

In Boettcher and Percus (2001) the EO algorithm is applied to the graph bi-partitioning problem (GBP) and it is empirically suggested that EO efficiently approximates the optimal bi-partition on Erdős–Rényi and random 3-regular graphs. Specifically, what their experiments actually show, is that the EO produces a partition which cut value scales in the same way than the expected value of the optimal cut with the number of nodes. They also remark that the convergence of EO appears to be linear on the number of steps. However, mathematical guarantees for the EO algorithm are not yet known.

It is straightforward to adapt the EO algorithm from GBP to max-cut. In statistical physics, max-cut and GBP can be formulated in terms of finding the ground state configurations of an Ising model on random d-regular graphs. For any graph, the Ising spin Hamiltonian is given by

$$\mathcal{H}(\mathbf{x}) = -\sum_{(i,j)\in E} J_{ij}x_i x_j$$

where each spin variable $x_i \in \{-1,+1\}$ is connected to each of its nearest neighbors j via a bond variable $J_{ij} \in \{-1,+1\}$, assigned at random. The configuration space Ω consist of all configurations $\mathbf{x} = (x_1, x_2, ..., x_n) \in \Omega$ where $|\Omega| = 2^n$. The goal of EO is to minimize the cost function $\mathcal{H}(\mathbf{x})$.

Note that,

$$\mathsf{max\text{-}cut}(G) = \max_{\mathbf{x}\in\{\pm 1\}^n} \frac{1}{2}\sum_{i<j} w_{ij}(1-x_i x_j) = |E(G)| - \min_{\mathbf{x}\in\{\pm 1\}^n} \frac{1}{2}\sum_{i<j} w_{ij}x_i x_j.$$

Instead of finding a ground state of the ferromagnetic Ising model with magnetization fixed at zero as in the GBP, the max-cut problem is to find a ground state configurations of the anti-ferromagnetic Ising model by minimizing the Hamiltonian $\mathcal{H}$ that can be written as

$$\mathcal{H}(\mathbf{x}) = \frac{1}{2}\sum_{i<j} w_{ij}x_i x_j$$

where $w_{i,j}$ is the (i,j) entry of the adjacency matrix $W = W^T$ of the graph. To find low-energy configurations, EO assigns a fitness to each x_i

$$\lambda_i = -x_i \left(\frac{1}{2} \sum_j w_{ij} x_j \right),$$

so that

$$\mathcal{H}(\mathbf{x}) = -\sum_{i=1}^{n} \lambda_i.$$

heuristically, the fitness of each variable assesses its contribution to the total cost. Note that $\lambda_i = -\frac{d}{2} + b_i$, where g_i is the number of edges connecting i to other vertices within its same set ("good" edges), and b_i is the number of and edges connecting i to vertices across the partition ("bad" edges). We consider a normalized version,

$$\lambda_i = \frac{b_i}{g_i + b_i} = \frac{b_i}{d} \tag{1}$$

as in Boettcher and Percus (2001), so that $\lambda_i \in [0, 1]$.

EO proceeds through a local search of Ω by sequentially changing variables with "bad" fitness on each update. It ranks all the variables x_i according to fitness λ_i, i.e., find a permutation Π of the variable labels i with

$$\lambda_{\Pi(1)} \leq \lambda_{\Pi(2)} \leq \cdots \leq \lambda_{\Pi(n)}.$$

If one always only updates the lowest-ranked variable, the search will risk reaching a "dead end" in the form of a poor local minimum. In contrast, the idea is to consider a probability distribution over the ranks k,

$$P_k \propto k^{-\tau}, \quad 1 \leq k \leq n, \tag{2}$$

for a given value of the parameter τ. At each update, select a rank k according to P_k, then if advantageous for fitness, change the state of the variable x_i with $i = \Pi(k)$. After each update, the fitness of the changed variable and of all its neighbours are re-evaluated according to (1).

2.3 Graph Neural Network

Recently, Chen et al. (2017) proposed a family of Graph Neural Networks (GNNs) (Gori et al., 2005; Bronstein et al., 2017) for community detection on graphs. Inspired by the success of regularized spectral methods based on non-backtracking random walks for community detection by Krzakala et al. (2013), Saade et al. (2014) and Bordenave et al. (2015), Chen et al. (2017) extend the Graph Neural Networks architecture to express such objects. These GNNs learn clustering algorithms in a purely (supervised) data-driven manner without access to the underlying generative models, and even empirically improving upon the state of the art for some specific regimes. They have also been adapted to problems like graph matching and traveling salesman Nowak et al. (2018).

The generic GNN, introduced in Scarselli et al. (2009) and later simplified in Li et al. (2015); Duvenaud et al. (2015); Sukhbaatar et al. (2016) is a neural network architecture that is based on finding a way to combine local operators on a undirected graph $G = (V, E)$ to provide the desired output. The usual choice of local operators are degree operator $(Dx)_i := \deg(i) \cdot x_i$, $D(x) = \mathrm{diag}(A\mathbf{1})x$, and the adjacency operator A, which is the linear map given by the adjacency matrix $A_{i,j} = 1$ when $(i,j) \in E$. Chen et al. (2017) further consider the power graph adjacency $A_j = \min(1, A^{2^j})$, which encodes 2^j-hop neighborhoods into a binary graph and therefore allows to combine and aggregate local information at different scales. The architecture has two folds, one is the operation on the original graph G, the other is on the line graph $L(G) = (V_L, E_L)$, which represents the edge adjacency structure of G, with non-backtracking operator B. The vertices V_L of $L(G)$ are the ordered edges in E, that is

$V_L = \{(i \to j); (i,j) \in E\} \bigcup \{(j \to i); (i,j) \in E\}$, so $|V_L| = 2|E|$. The non-backtracking operator $B \in \mathbb{R}^{2|E| \times 2|E|}$ encodes the edge adjacency structure as follows,

$$B_{(i\to j),(i'\to j')} = \begin{cases} 1 & \text{if } j = i' \text{ and } j' \neq i, \\ 0 & \text{otherwise.} \end{cases}$$

with the corresponding degree $D_B = \mathrm{diag}(B\mathbf{1})$ operators. This effectively defines edge features that are diffused and updated according to the edge adjacency of G. Edge and node features are combined at each layer using the edge indicator matrices Pm, Pd $\in \{0,1\}^{|V| \times 2|E|}$, defined as $\mathrm{Pm}_{i,(i\to j)} = 1$, $\mathrm{Pm}_{j,(i\to j)} = 1$, $\mathrm{Pd}_{i,(i\to j)} = 1$, $\mathrm{Pd}_{j,(i\to j)} = -1$ and 0 otherwise. Then, for each layer, with an input signal $u^{(k)} \in \mathbb{R}^{|V| \times b_k}$ on G and $v^{(k)} \in \mathbb{R}^{|V_L| \times b_k}$ on line graph $L(G)$, the model produces $u^{(k+1)} \in \mathbb{R}^{|V| \times b_{k+1}}$ and $v^{(k+1)} \in \mathbb{R}^{|V_L| \times b_{k+1}}$ as

$$\begin{aligned}
u_{i,l}^{(k+1)} &= \rho\left[u_i^{(k)}\theta_{1,l}^{(k)} + \sum_{j=0}^{J-1}(A^{2^j}u^{(k)})_i\theta_{3+j,l}^{(k)} + \{\mathrm{Pm},\mathrm{Pd}\}v^{(k)}\theta_{3+J,l}^{(k)}\right], l = 1,...,b_{k+1}/2, i \in V, \\
v_{i',l}^{(k+1)} &= \rho\left[v_{i'}\gamma_{1,l}^{(k)} + (D_{L(G)}v^{(k)})_{i'}\gamma_{2,l}^{(k)} + \sum_{j=0}^{J}(A_{L(G)}^{2^j}v^{(k)})_{i'}\gamma_{3+j,l}^{(k)} + [\{\mathrm{Pm},\mathrm{Pd}\}^T u^{(k+1)}]_{i'}\gamma_{3+J,l}^{(k)}\right], i \in V_L, \\
u_{i,l}^{(k+1)} &= u_i^{(k)}\theta_{1,l}^{(k)} + \sum_{j=0}^{J-1}(A^{2^j}u^{(k)})_i\theta_{3+j,l}^{(k)} + \{\mathrm{Pm},\mathrm{Pd}\}v^{(k)}\theta_{3+J,l}^{(k)}, l = b_{k+1}/2+1,...,b_{k+1}, i \in V, \\
v_{i',l}^{(k+1)} &= v_{i'}\gamma_{1,l}^{(k)} + (D_{L(G)}v^{(k)})_{i'}\gamma_{2,l}^{(k)} + \sum_{j=0}^{J}(A_{L(G)}^{2^j}v^{(k)})_{i'}\gamma_{3+j,l}^{(k)} + [\{\mathrm{Pm},\mathrm{Pd}\}^T u^{(k+1)}]_{i'}\gamma_{3+J,l}^{(k)}, i \in V_L
\end{aligned} \tag{3}$$

where $\Theta = \{\theta_1^(k), ..., \theta_{J+3}^{(k)}, \gamma_1^{(k)}, ..., \gamma_{J+3}^{(k)}\}$, $\theta_s^{(k)}$, $\gamma_s^{(k)} \in \mathbb{R}^{b_k \times b_{k+1}}$ are trainable parameters and $\rho(\cdot)$ is a point-wise nonlinearity, chosen in this work to be $\rho(z) = \max(0, z)$. The model (3) may seem a little daunting, but it's basically an unrolled overparametrized power iteration, on node features as well as edge features, where the parameters Θ indicate how to combine the operators of the graph to produce an optimized *regularized spectral method*.

Assume that a training set $\{(G_t, \mathbf{y}_t)\}_{t \le T}$ is given, with any signal $\mathbf{y} = (y_1, ..., y_n) : V \to \{1, 2, ..., C\}^n$ encoding a partition of V into C groups. It is then used by LGNN to learn a model $\widehat{\mathbf{y}} = \Phi(G, \Theta)$ trained by minimizing

$$L(\Theta) = \frac{1}{T}\sum_{t \le T} l(\Phi(G_t, \Theta), \mathbf{y}_t). \tag{4}$$

In the max-cut problem, $C = 2$. Unlike community detection on SBM, the max-cut problem on random graphs is cast as an unsupervised learning problem with no knowledge of $\{\mathbf{y}_t\}$ in the training process. We define the loss function as the cut itself, then the model $\widehat{\mathbf{y}} = \Phi(G, \theta)$ is learned by training

$$\max_{\Theta} L(\Theta) = \max_{\Theta} \frac{1}{T}\sum_{t \le T} l(\Phi(G_t, \Theta)) = -\min_{\Theta} \frac{1}{T}\sum_{t \le T} -l(\Phi(G_t, \Theta)).$$

with

$$l(\Phi(G_t, \Theta)) = \frac{1}{4}\mathbf{x}^T \mathcal{L}_G \mathbf{x} =: f(\mathbf{x}, G_t) \tag{5}$$

where $\mathbf{x} = (x_1, x_2, ..., x_n) \in \{\pm 1\}^n$ is the resulted configurations and where $\mathcal{L}_G$ is the graph Laplacian. Note that, $f(\mathbf{x}, G_t)$ is not differentiable, but we wish to apply stochastic gradient decent methods. To this end, we propose two different methods: a relaxation method, and a policy gradient method. We make use of the probability matrix $\pi(\mathbf{x}|\Theta, G) \in [0,1]^{n \times 2}$ (with (i,j) entry equal to $\mathbb{P}(x_i = j|\Theta, G)$), $j = \pm 1$, which is an intermediate result produced by LGNN.

2.3.1 Relaxation method on loss function for Max-Cut problem

Denote $p_i = \mathbb{P}(x_i = 1|\Theta, G)$. Substitute $\mathbf{x}$ with $2\mathbf{p} - 1 \in [-1, 1]^n$, then we can write the loss function as

$$l(\Phi(G, \Theta)) = \frac{1}{4}(2\mathbf{p} - 1)^T \mathcal{L}_G (2\mathbf{p} - 1)$$

which is differentiable with respect to the parameters of the model.

2.3.2 Policy gradient methods on loss function for Max-Cut problem

Policy gradient is a technique from reinforcement learning that relies upon optimizing parameterized policies with respect to the expected return by gradient descent. Informally the "policies" we consider in this context correspond with the distribution of choices of cuts given a graph, which is a function of the set of parameters Θ. The expected return under that policy is the expected value of the cut given the graph.

In the max-cut problem, for any given graph G, consider the following optimization

$$\Theta^* = \arg\max_{\Theta} \mathbb{E}_{\mathbf{x}\sim\pi_\Theta(\cdot|G)}[f(\mathbf{x}, G)].$$

Here $\mathbf{x} \sim \pi_\Theta(\cdot|G)$ can be viewed as the policy and $\mathbb{E}_{\mathbf{x}\sim\pi_\Theta(\cdot|G)}[f(\mathbf{x}, G)]$ as the expected return.

Many machine learning approaches base their optimization on variations of policy gradients (see for instance Degris et al. (2012); Watkins and Dayan (1992)). We use the simplest policy gradient formulation, corresponding to the seminal work by Sutton et al. (1999). Note that

$$\begin{aligned}\nabla_\Theta \mathbb{E}_{\mathbf{x}\sim\pi_\Theta(\cdot|G)}[f(\mathbf{x}, G)] &= \int f(\mathbf{x}, G)\pi_\Theta(\mathbf{x}|G)\nabla_\Theta \log \pi_\Theta(\mathbf{x}|G) d\mathbf{x} \\ &= \mathbb{E}_{\mathbf{x}\sim\pi_\Theta(\cdot|G)}[f(\mathbf{x}, G)\nabla_\Theta \log \pi_\Theta(\mathbf{x}|G)]. \end{aligned} \tag{6}$$

The intermediate output probability matrix $\pi(\cdot|\Theta, G)$ from LGNN enables us to sample $\mathbf{x}$ from its distribution (that depends on the current Θ and the given training graph G). We sample K independent instances of $\mathbf{x}$ obtaining $\{\mathbf{x}_k\}_{k=1}^K$. Then (6) can be approximated by

$$\nabla_\Theta \mathbb{E}_{\mathbf{x}\sim\pi_\Theta(\cdot|G)}[f(\mathbf{x}, G)] \approx \frac{1}{K}\sum_{k=1}^K [f(\mathbf{x}_k, G)\nabla_\Theta \log \pi_\Theta(\mathbf{x}_k|G)] \tag{7}$$

$$= \nabla_\Theta \left\{ \frac{1}{K}\sum_{k=1}^K [f(\mathbf{x}_k, G) \log \pi_\Theta(\mathbf{x}_k|G)] \right\}. \tag{8}$$

The loss function can then set to be

$$l(\Phi(G, \Theta)) = \frac{1}{K}\sum_{k=1}^K [f(\mathbf{x}_k, G) \log \pi_\Theta(\mathbf{x}_k|G)],$$

with $\{\mathbf{x}_k\}_{k=1}^K$ sampled from $\pi_\Theta(\cdot|G)$; and we can use the gradient from (8) to implement stochastic gradient descent on the loss function (4).

3. EXPERIMENTAL EVALUATION

Denoting by $G^{\text{reg}}(n, d)$ the uniform distribution over graphs with n vertices and uniform degree d, Dembo et al. (2017) have proved that, with high probability, as $n \to \infty$ the size of the max-cut satisfies

$$\mathsf{MaxCut}(G^{\text{Reg}}(n, d)) = n\left(\frac{d}{4} + P_*\sqrt{\frac{d}{4}} + o_d\left(\sqrt{d}\right)\right) + o(n)$$

with $P_* \approx 0.7632$ an universal constant related to the ground state energy of the Sherrington-Kirkpatrick model, that can be expressed analytically via Parisi's formula (Dembo et al., 2017; Percus et al., 2008; Talagrand, 2006).

Motivated by these asymptotics, for any candidate solution $\mathbf{x} = (x_1, ..., x_n)$, and the corresponding cut size $z = \frac{1}{2}\sum_{i<j} w_{ij}(1 - x_i x_j)$, we evaluate it by computing its corresponding P as

$$P = \frac{z/n - d/4}{\sqrt{d/4}}. \tag{9}$$

The larger the P the better the cut, and moreover, if P is closer $P_* \approx 0.7632$ we expect the candidate cut to be among the best possible in the graph.

It is worth noting that the SDP relaxation is not tight in the asymptotic limit (Montanari and Sen, 2016). Actually, in the limit the corresponding P of the SDP's (fractional) solution is 1, which is not an improvement over a simple spectral method. However our experiments in the next Section indicate that, at least for the parameter range we are investigating, the rounding procedure still produces non-trivial cuts.

We are also interested in how much the resulted configurations for a given graph overlap with each other. We define the overlap measure ν for any two configuration $\mathbf{x}_i = (x_1, x_2, ..., x_n) \in \{\pm 1\}^n$, $i = 1, 2$, as

$$\nu = \frac{1}{n} \left| \langle \mathbf{x}_1, \mathbf{x}_2 \rangle \right|. \tag{10}$$

In our numerical simulations, we mainly focus on the performance of the discussed methods when the graphs are relatively sparse. To obtain a more thorough comparison of the performance of the three methods, various values of node size n and degree d were chosen; the cut values were computed over 1000 random d-regular graphs with n vertices.

The values of the tuning parameters of the three methods used in the simulations are given and explained as follows. We use SDPNAL+ (Yang et al., 2015) to solve the semidefinite program (SDP), and the random rounding procedure chooses the best cut over 500 iterations. For EO, there is only one parameter, the exponent τ in the probability distribution in equation (2), governing the update process and consequently the performance of EO. Boettcher and Percus (2001) investigated numerically the optimal value for τ, which they view as a function of runtime $t_{\max}$ and node size n. They observed optimal performance on graphs of node size $n \leq 10000$ with $t_{\max} \geq 100n$ with $\tau \approx 1.3 \sim 1.6$. Based on their numerical findings, they claim that the value of τ does not significantly change the value of the solutions as long as the algorithm runs for enough iterations. Therefore, following their discussion, we simply fixed $\tau = 1.4$ and set a sufficiently large number of iterations $t_{\max} = 10^4 n$. In addition, as the labels are randomly assigned at the beginning of the EO algorithm, it is expected that the different initialization may lead to different outcomes. Therefore, multiple runs of the EO is suggested. Here we chose the best run out of two. To train Chen et al. (2017) model on the max-cut problem, we set the input signals to be $x^{(0)} = \deg(G)$ and $y^{(0)} = \deg(L(G))$. Other parameters were set to be the same as what was used in Chen et al. (2017)'s simulations, $J = 3$, $K = 30$, $b_k = 10$, $k = 2, ..., K-1$ ($b_0 = 1$ and $b_K = C$, where $C = 2$ for the max-cut problem) in (3). We trained the model on 5000 graphs with the same node size n and degree d. Though different sets of tuning parameters in GNNs may change the overall performance, we believe such settings still can provide us a reasonable understanding of how well GNNs perform compared to the other two methods.

All our code is publicly available in Yao (2019).

3.1 The size of the computed cut

Same node size n with different degrees d. First we explore the setting where we fix the node size n, and let the degree d change. Table 1 provides the results on graphs with $n = 500$ nodes. The results on graphs of other node sizes are similar.

The experiments reported in Table 1 show that the computed P improves as the degree decreases. Overall, EO appears to give the best performance. SDP outperforms GNNs while GNNs seem to catch up when the graph gets sparser.

n	d	Methods			
		GNNs		SDP	EO
		Relaxation	Policy Gradient		
500	20	0.6136	0.6259	0.6742	0.7315
	15	0.6070	0.6289	0.6783	0.7359
	10	0.6704	0.5989	0.6820	0.7352
	5	0.7014	0.6682	0.6898	0.7369
	3	0.7074	0.6928	0.7015	0.7266

Table 1. Computed P values for different methods, with same node size $n = 500$ and various degrees d.

Same degree d with increasing node size n Asymptotically, the optimal P on random d-regular graphs should approach P_* as $n \to \infty$ and $d \to \infty$. Tables 2 and 3 provide two instances when $d = 3$ and $d = 10$, respectively. In both cases, the computed P for all three methods increase as the node size n increases, getting closer to P_*. Furthermore, the EO method shows greater advantage over the other methods as the node size n increases. The performance of SDP is still comparable to that of GNNs when node size n is up to 500.

d	n	Methods			
		GNN		SDP	EO
		Relaxation	Policy Gradient		
3	50	0.6716	0.6868	0.6981	0.6985
	100	0.6978	0.6914	0.7090	0.7118
	200	0.7075	0.7010	0.7091	0.7210
	500	0.7074	0.6928	0.7015	0.7266

Table 2. Computed P values for different methods, with same degrees $d = 3$ but different node sizes n.

d	n	Methods			
		GNN		SDP	EO
		Relaxation	Policy Gradient		
10	50	0.5874	0.5995	0.6614	0.6643
	100	0.6296	0.6368	0.6889	0.7033
	200	0.6456	0.6598	0.6919	0.7241
	500	0.6704	0.5989	0.6820	0.7369

Table 3. Computed P values for different methods, with same degrees $d = 10$ but different node sizes n.

4. CONCLUSIONS

The experimental results presented in this paper suggest that unsupervised machine learning techniques can be successfully adapted to hard optimization problems on random inputs. We observe this in the particular case of max-cut on random regular graphs, where graph neural networks learn algorithms that attain comparable performance to the Goemans and Williamson SDP for this particular model. This finding is noteworthy due to the unsupervised –and computationally hard– nature of the task.

Overall, the three methods we consider –SDP, EO, and GNNs– appear to compute solutions with comparable objective values. Preliminary numerical simulations, not reported in the current paper but available in Yao

(2019), suggest that the overlap (or correlation) between different solutions with similar objective value can still be quite different, which is a statement on the difficulty of the optimization problem.

Throughout these simulations we observe that extremal optimization is the most computationally efficient and it consistently outperforms the SDP and GNNs under both, dense and sparse regimes. We therefore believe there is a need for further theoretical analysis of extremal optimization. In our simulations EO appeared to be somewhat initialization dependent, which if not a finite n artifact, could prove a difficulty when trying to theoretically analyze this method.

Finally, there exists significant room for improvement on the implementation of the GNNs. It would not be surprising if a better architecture or choice of hyperparameters can be used to improve upon our results.

Acknowledgments

The authors would like to thank Dustin Mixon and Zhengdao Chen. SV is partly supported by NSF-DMS 1913134, EOARD FA9550-18-1-7007 and the Simons Algorithms and Geometry (A&G) Think Tank. ASB was partially supported by NSF grants DMS-1712730 and DMS-1719545, and by a grant from the Sloan Foundation.

References

M. R. Garey and D. S. Johnson. *Computers and intractability: A guide to the theory of NP-completeness*. W. H. Freeman & Co., New York, NY, USA, 1990.

M. X. Goemans and D. P. Williamson. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programing. *Journal of the ACM (JACM)*, 42(6):1115–1145, 1995.

Emmanuel Abbe. Community detection and stochastic block models: recent developments. *The Journal of Machine Learning Research*, 18(1):6446–6531, 2017.

A. Dembo, A. Montanari, and S. Sen. Extremal cuts of sparse random graphs. *The Annals of Probability*, 45(2): 1190–1217, 2017.

Lenka Zdeborová and Stefan Boettcher. A conjecture on the maximum cut and bisection width in random regular graphs. *Journal of Statistical Mechanics: Theory and Experiment*, 2010(02):P02020, 2010.

Afonso S Bandeira, Amelia Perry, and Alexander S Wein. Notes on computational-to-statistical gaps: predictions using statistical physics. *arXiv preprint arXiv:1803.11132*, 2018.

M. Gori, G. Monfardini, and F. Scarselli. A new model for learning in graph domains. In *Proceedings. 2005 IEEE International Joint Conference on Neural Networks, 2005.*, volume 2, pages 729–734, 2005.

M. M. Bronstein, J. Bruna, Y. LeCun, A. Szlam, and P. Vandergheynst. Geometric deep learning: Going beyond euclidean data. *IEEE Signal Processing Magazine*, 34(4):18–42, 2017.

Z. Chen, X. Li, and J. Bruna. Supervised community detection with line graph neural networks. *arXiv e-prints*, art. arXiv:1705.08415, 2017.

S. Boettcher and A. G. Percus. Extremal optimization: Methods derived from co-evolution. *Proceedings of the 1999 Genetic and Evolutionary Computation Conference (GECCO '99)*, pages 825–832, 1999.

S. Boettcher and A. G. Percus. Extremal optimization for graph partitioning. *Physical Review E*, 64, 026114, 2001.

Yu-Wang Chen, Yong-Zai Lu, and Peng Chen. Optimization with extremal dynamics for the traveling salesman problem. *Physica A: Statistical Mechanics and its Applications*, 385(1):115–123, 2007.

Andrea Montanari. Optimization of the Sherrington-Kirkpatrick hamiltonian. *arXiv preprint arXiv:1812.10897*, 2018.

S. Boettcher and A. G. Percus. Nature's way of optimizing. *Artificial Intelligence*, 119:275–286, 2000.

P. Bak and K. Sneppen. Punctuated equilibrium and criticality in a simple model of evolution. *Physical Review Letters*, 71(24), 1993.

Florent Krzakala, Cristopher Moore, Elchanan Mossel, Joe Neeman, Allan Sly, Lenka Zdeborová, and Pan Zhang. Spectral redemption in clustering sparse networks. *Proceedings of the National Academy of Sciences*, 110(52): 20935–20940, 2013.

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.

Charles Bordenave, Marc Lelarge, and Laurent Massoulié. Non-backtracking spectrum of random graphs: community detection and non-regular ramanujan graphs. In *2015 IEEE 56th Annual Symposium on Foundations of Computer Science*, pages 1347–1357. IEEE, 2015.

Alex Nowak, Soledad Villar, Afonso S Bandeira, and Joan Bruna. Revised note on learning quadratic assignment with graph neural networks. In *2018 IEEE Data Science Workshop (DSW)*, pages 1–5. IEEE, 2018.

F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini. The graph neural network model. *IEEE Trans. Neural Networks*, 20(1):61–80, 2009.

Y. Li, D. Tarlow, M. Brockschmidt, and R. Zemel. Gated graph sequence neural networks. *arXiv preprint arXiv:1511.05493*, 2015.

D. Duvenaud, D. Maclaurin, J. Aguilera-Iparraguirre, A. Aspuru-Guzik R. Gómez-Bombarelli, T. Hirzel, and R. P. Adams. Convolutional networks on graphs for learning molecular fingerprints. Neural Information Processing Systems, 2015.

S. Sukhbaatar, A. Szlam, and R. Fergus. Learning multiagent communication with backpropagation. Advances in Neural Information Processing Systems, pages 2244–2252, 2016.

T. Degris, P. M. Pilarski, and R. S. Sutton. Model-free reinforcement learning with continuous action in practice. In *2012 American Control Conference (ACC)*, pages 2177–2182, 2012.

C. Watkins and P. Dayan. Q-learning. *Machine Learning*, 8(3):279–292, 1992.

R. S. Sutton, D. A. McAllester, S. P. Singh, and Y. Mansour. Policy gradient methods for reinforcement learning with function approximation. In *Neural Information Processing Systems 12*, pages 1057–1063, 1999.

Allon G Percus, Gabriel Istrate, Bruno Gonçalves, Robert Z Sumi, and Stefan Boettcher. The peculiar phase structure of random graph bisection. *Journal of Mathematical Physics*, 49(12):125219, 2008.

Michel Talagrand. The parisi formula. *Annals of mathematics*, pages 221–263, 2006.

Andrea Montanari and Subhabrata Sen. Semidefinite programs on sparse random graphs and their application to community detection. In *Proceedings of the forty-eighth annual ACM symposium on Theory of Computing*, pages 814–827. ACM, 2016.

Liuqin Yang, Defeng Sun, and Kim-Chuan Toh. SDPNAL+: a majorized semismooth newton-cg augmented lagrangian method for semidefinite programming with nonnegative constraints. *Mathematical Programming Computation*, 7(3):331–366, 2015.

Weichi Yao. Implementation of GNN, SDP and EO for max-cut. `https://github.com/ElainaYao/maxCut`, 2019.