Subsampled Power Iteration: a Unified Algorithm for Block Models and Planted CSP's

Vitaly Feldman
IBM Research - Almaden
vitaly@post.harvard.edu

Will Perkins
University of Birmingham
w.f.perkins@bham.ac.uk

Santosh Vempala
Georgia Tech
vempala@cc.gatech.edu

Abstract

We present an algorithm for recovering planted solutions in two well-known models, the stochastic block model and planted constraint satisfaction problems (CSP), via a common generalization in terms of random bipartite graphs. Our algorithm matches up to a constant factor the best-known bounds for the number of edges (or constraints) needed for perfect recovery and its running time is linear in the number of edges used. The time complexity is significantly better than both spectral and SDP-based approaches.

The main contribution of the algorithm is in the case of unequal sizes in the bipartition that arises in our reduction from the planted CSP. Here our algorithm succeeds at a significantly lower density than the spectral approaches, surpassing a barrier based on the spectral norm of a random matrix.

Other significant features of the algorithm and analysis include (i) the critical use of power iteration with subsampling, which might be of independent interest; its analysis requires keeping track of multiple norms of an evolving solution (ii) the algorithm can be implemented statistically, i.e., with very limited access to the input distribution (iii) the algorithm is extremely simple to implement and runs in linear time, and thus is practical even for very large instances.

1 Introduction

A broad class of learning problems fits into the framework of obtaining a sequence of independent random samples from a unknown distribution, and then (approximately) recovering this distribution using as few samples as possible. We consider two natural instances of this framework: the stochastic block model in which a random graph is formed by choosing edges independently at random with probabilities that depend on whether an edge crosses a planted partition, and planted k-CSP's (or planted k-SAT) in which width-k boolean constraints are chosen independently at random with probabilities that depend on their evaluation on a planted assignment to a set of boolean variables.

We propose a natural bipartite generalization of the stochastic block model, and then show that planted k-CSP's can be reduced to this model, thus unifying graph partitioning and planted CSP's into one problem. We then give an algorithm for solving random instances of the model. Our algorithm is optimal up to a constant factor in terms of number of sampled edges and running time for the bipartite block model; for planted CSP's the algorithm matches up to log factors the best possible sample complexity in several restricted computational models and the best-known bounds for any algorithm. A key feature of the algorithm is that when one side of the bipartition is much

larger than the other, then our algorithm succeeds at significantly lower edge densities than using Singular Value Decomposition (SVD) on the rectangular adjacency matrix. Details are in Sec. 5.

The bipartite block model begins with two vertex sets, V_1 and V_2 (of possibly unequal size), each with a balanced partition, (A_1, B_1) and (A_2, B_2) respectively. Edges are added independently at random between V_1 and V_2 with probabilities that depend on which parts the endpoints are in: edges between A_1 and A_2 or B_1 and B_2 are added with probability δp, while the other edges are added with probability $(2-\delta)p$, where $\delta \in [0, 2]$ and p is the overall edge density. To obtain the stochastic block model we can identify V_1 and V_2. To reduce planted CSP's to this model, we first reduce the problem to an instance of noisy r-XOR-SAT, where r is the complexity parameter of the planted CSP distribution defined in [19] (see Sec. 2 for details). We then identify V_1 with literals, and V_2 with $(r-1)$-tuples of literals, and add an edge between literal $l \in V_1$ and tuple $t \in V_2$ when the r-clause consisting of their union appears in the formula. The reduction leads to a bipartition with V_2 much larger than V_1.

Our algorithm is based on applying power iteration with a sequence of matrices subsampled from the original adjacency matrix. This is in contrast to previous algorithms that compute the eigenvectors (or singular vectors) of the full adjacency matrix. Our algorithm has several advantages. Such an algorithm, for the special case of square matrices, was previously proposed and analyzed in a different context by Korada *et al.* [25].

- Up to a constant factor, the algorithm matches the best-known (and in some cases the best-possible) edge or constraint density needed for complete recovery of the planted partition or assignment. The algorithm for planted CSP's finds the planted assignment using $O(n^{r/2} \cdot \log n)$ clauses for a clause distribution of complexity r (see Sec. 2 for the formal definition), nearly matching computational lower bounds for SDP hierarchies [30] and the class of statistical algorithms [19].
- The algorithm is fast, running in time linear in the number of edges or constraints used, unlike other approaches that require computing eigenvectors or solving semi-definite programs.
- The algorithm is conceptually simple and easy to implement. In fact it can be implemented in the statistical query model, with very limited access to the input graph [19].
- It is based on the idea of iteration with subsampling which may have further applications in the design and analysis of algorithms.
- Most notably, the algorithm succeeds where generic spectral approaches fail. For the case of the planted CSP, when $|V_2| \gg |V_1|$, our algorithm succeeds at a polynomial factor sparser density than the approaches of McSherry [28], Coja-Oghlan [7], and Vu [33]. The algorithm succeeds despite the fact that the 'energy' of the planted vector with respect to the random adjacency matrix is far below the spectral norm of the matrix. In previous analyses, this was believed to indicate failure of the spectral approach. See Sec. 5.

1.1 Related work

The algorithm of Mossel, Neeman and Sly [29] for the standard stochastic block model also runs in near linear time, while other known algorithmic approaches for planted partitioning that succeed near the optimal edge density [28, 7, 27] perform eigenvector or singular vector computations and thus require superlinear time, though a careful randomized implementation of low-rank approximations can reduce the running time of McSherry's algorithm substantially [2].

For planted satisfiability, the algorithm of Flaxman for planted 3-SAT works for a subset of planted distributions (those with distribution complexity at most 2 in our definition below) using $O(n)$ constraints, while the algorithm of Coja-Oghlan, Cooper, and Frieze [8] works for planted 3-SAT distributions that exclude unsatisfied clauses and uses $O(n^{3/2} \ln^{10} n)$ constraints.

The only previous algorithm that finds the planted assignment for all distributions of planted k-CSP's is the SDP-based algorithm of Bogdanov and Qiao [5] with the folklore generalization to r-wise independent predicates (*cf.* [30]). Similar to our algorithm, it uses $\tilde{O}(n^{r/2})$ constraints. This algorithm effectively solves the noisy r-XOR-SAT instance and therefore can be also used to solve our general version of planted satisfiability using $\tilde{O}(n^{r/2})$ clauses (via the reduction in Sec. 4).

Notably for both this algorithm and ours, having a completely satisfying planted assignment plays no special role: the number of constraints required depends only on the distribution complexity.To the best of our knowledge, our algorithm is the first for the planted k-SAT problem that runs in linear time in the number of constraints used.

It is important to note that in planted k-CSP's, the planted assignment becomes recoverable with high probability after at most $O(n \log n)$ random clauses yet the best known efficient algorithms require $n^{\Omega(r/2)}$ clauses. Problems exhibiting this type of behavior have attracted significant interest in learning theory [4, 12, 31, 15, 32, 3, 10, 16] and some of the recent hardness results are based on the conjectured computational hardness of the k-SAT refutation problem [10, 11].

Our algorithm is arguably simpler than the approach in [5] and substantially improves the running time even for small k. Another advantage of our approach is that it can be implemented using restricted access to the distribution of constraints referred to as statistical queries [24, 17]. Roughly speaking, for the planted SAT problem this access allows an algorithm to evaluate multi-valued functions of a single clause on randomly drawn clauses or to estimate expectations of such functions, *without direct access to the clauses themselves*. Recently, in [19], lower bounds on the number of clauses necessary for a polynomial-time statistical algorithm to solve planted k-CSPs were proved. It is therefore important to understand the power of such algorithms for solving planted k-CSPs. A statistical implementation of our algorithm gives an upper bound that nearly matches the lower bound for the problem. See [19] for the formal details of the model and statistical implementation of our algorithm.

Korada, Montanari and Oh [25] analyzed the 'Gossip PCA' algorithm, which for the special case of an equal bipartition is the same as our subsampled power iteration. The assumptions, model, and motivation in the two papers are different and the results incomparable. In particular, while our focus and motivation are on general (nonsquare) matrices, their work considers extracting a planting of rank k greater than 1 in the square setting. Their results also assume an initial vector with non-trivial correlation with the planted vector. The nature of the guarantees is also different.

2 Model and results

Bipartite stochastic block model:

Definition 1. *For $\delta \in [0,2] \setminus \{1\}$, n_1, n_2 even, and $\mathcal{P}_1 = (A_1, B_1)$, $\mathcal{P}_2 = (A_2, B_2)$ bipartitions of vertex sets V_1, V_2 of size n_1, n_2 respectively, we define the bipartite stochastic block model $B(n_1, n_2, \mathcal{P}_1, \mathcal{P}_2, \delta, p)$ to be the random graph in which edges between vertices in A_1 and A_2 and B_1 and B_2 are added independently with probability δp and edges between vertices in A_1 and B_2 and B_1 and A_2 with probability $(2-\delta)p$.*

Here δ is a fixed constant while p will tend to 0 as $n_1, n_2 \to \infty$. Note that setting $n_1 = n_2 = n$, and identifying A_1 and A_2 and B_1 and B_2 gives the usual stochastic block model (with loops allowed); for edge probabilities a/n and b/n, we have $\delta = 2a/(a+b)$ and $p = (a+b)/2n$, the overall edge density. For our application to k-CSP's, it will be crucial to allow vertex sets of very different sizes, i.e. $n_2 \gg n_1$.

The algorithmic task for the bipartite block model is to recover one or both partitions (completely or partially) using as few edges and as little computational time as possible. In this work we will assume that $n_1 \leq n_2$, and we will be concerned with the algorithmic task of recovering the partition $\mathcal{P}_1$ completely, as this will allow us to solve the planted k-CSP problems described below. We define complete recovery of $\mathcal{P}_1$ as finding the exact partition with high probability over the randomness in the graph and in the algorithm.

Theorem 1. *Assume $n_1 \leq n_2$. There is a constant C so that the Subsampled Power Iteration algorithm described below completely recovers the partition $\mathcal{P}_1$ in the bipartite stochastic block model $B(n_1, n_2, \mathcal{P}_1, \mathcal{P}_2, \delta, p)$ with probability $1 - o(1)$ as $n_1 \to \infty$ when $p \geq \frac{C \log n_1}{(\delta-1)^2 \sqrt{n_1 n_2}}$. Its running time is $O\left(\sqrt{n_1 n_2} \cdot \frac{\log n_1}{(\delta-1)^2}\right)$.*

Note that for the usual stochastic block model this gives an algorithm using $O(n \log n)$ edges and $O(n \log n)$ time, which is the best possible for complete recovery since that many edges are needed for every vertex to appear in at least edge. With edge probabilities $a \log n/n$ and $b \log n/n$, our

results require $(a-b)^2 \geq C(a+b)$ for some absolute constant C, matching the dependence on a and b in [6, 28] (see [1] for a discussion of the best possible threshold for complete recovery).

For any n_1, n_2, at least $\sqrt{n_1 n_2}$ edges are necessary for even non-trivial partial recovery, as below that threshold the graph consists only of small components (and even if a correct partition is found on each component, correlating the partitions of different components is impossible). Similarly at least $\Omega(\sqrt{n_1 n_2} \log n_1)$ are needed for complete recover of $\mathcal{P}_1$ since below that density, there are vertices in V_1 joined only to vertices of degree 1 in V_2.

For very lopsided graphs, with $n_2 \gg n_1 \log^2 n_1$, the running time is sublinear in the size of V_2; this requires careful implementation and is essential to achieving the running time bounds for planted CSP's described below.

Planted k-CSP's: We now describe a general model for planted satisfiability problems introduced in [19]. For an integer k, let $\mathcal{C}_k$ be the set of all ordered k-tuples of literals from $x_1, \ldots, x_n, \overline{x}_1, \ldots, \overline{x}_n$ with no repetition of variables. For a k-tuple of literals C and an assignment σ, $\sigma(C)$ denotes the vector of values that σ assigns to the literals in C. A planting distribution $Q : \{\pm 1\}^k \to [0,1]$ is a probability distribution over $\{\pm 1\}^k$.

Definition 2. *Given a planting distribution $Q : \{\pm 1\}^k \to [0,1]$, and an assignment $\sigma \in \{\pm 1\}^n$, we define the random constraint satisfaction problem $F_{Q,\sigma}(n,m)$ by drawing m k-clauses from $\mathcal{C}_k$ independently according to the distribution*

$$Q_\sigma(C) = \frac{Q(\sigma(C))}{\sum_{C' \in \mathcal{C}_k} Q(\sigma(C'))}$$

where $\sigma(C)$ is the vector of values that σ assigns to the k-tuple of literals comprising C.

Definition 3. *The distribution complexity $r(Q)$ of the planting distribution Q is the smallest integer $r \geq 1$ so that there is some $S \subseteq [k]$, $|S| = r$, so that the discrete Fourier coefficient $\hat{Q}(S)$ is non-zero.*

In other words, the distribution complexity of Q is r if Q is an $(r-1)$-wise independent distribution on $\{\pm 1\}^k$ but not an r-wise independent distribution. The uniform distribution over all clauses, $Q \equiv 2^{-k}$, has $\hat{Q}(S) = 0$ for all $|S| \geq 1$, and so we define its complexity to be ∞. The uniform distribution does not reveal any information about σ, and so inference is impossible. For any Q that is not the uniform distribution over clauses, we have $1 \leq r(Q) \leq k$.

Note that the uniform distribution on k-SAT clauses with at least one satisfied literal under σ has distribution complexity $r = 1$. $r = 1$ means that there is a bias towards either true or false literals. In this case, a very simple algorithm is effective: for each variable, count the number of times it appears negated and not negated, and take the majority vote. For distributions with complexity $r \geq 2$, the expected number of true and false literals in the random formula are equal and so this simple algorithm fails.

Theorem 2. *For any planting distribution Q, there exists an algorithm that for any assignment σ, given an instance of $F_{Q,\sigma}(n,m)$ completely recovers the planted assignment σ for $m = O(n^{r/2} \log n)$ using $O(n^{r/2} \log n)$ time, where $r \geq 2$ is the distribution complexity of Q. For distribution complexity $r = 1$, there is an algorithm that gives non-trivial partial recovery with $O(n^{1/2})$ constraints and complete recovery with $O(n \log n)$ constraints.*

3 The algorithm

We now present our algorithm for the bipartite stochastic block model. We define vectors u and v of dimension n_1 and n_2 respectively, indexed by V_1 and V_2, with $u_i = 1$ for $i \in A_1$, $u_i = -1$ for $i \in B_1$, and similarly for v. To recover the partition $\mathcal{P}_1$ it suffices to find either u or $-u$. We will find this vector by multiplying a random initial vector x_0 by a sequence of centered adjacency matrices and their transposes.

We form these matrices as follows: let G_p be the random bipartite graph drawn from the model $B(n_1, n_2, \mathcal{P}_1, \mathcal{P}_2, \delta, p)$, and T a positive integer. Then form T different bipartite graphs $G_1, \ldots, G_T$ on the same vertex sets V_1, V_2 by placing each edge from G_p uniformly and independently at random into one of the T graphs. The resulting graphs have the same marginal distribution.

Next we form the $n_1 \times n_2$ adjacency matrices $A_1, \ldots, A_T$ for $G_1, \ldots G_T$ with rows indexed by V_1 and columns by V_2 with a 1 in entry (i, j) if vertex $i \in V_1$ is joined to vertex $j \in V_2$. Finally we center the matrices by defining $M_i = A_i - \frac{p}{T}J$ where J is the $n_1 \times n_2$ all ones matrix.

The basic iterative steps are the multiplications $y = M^T x$ and $x = My$.

Algorithm: Subsampled Power Iteration.

1. Form $T = 10 \log n_1$ matrices $M_1, \ldots, M_T$ by uniformly and independently assigning each edge of the bipartite block model to a graph $G_1, \ldots, G_T$, then forming the matrices $M_i = A_i - \frac{p}{T}J$, where A_i is the adjacency matrix of G_i and J is the all ones matrix.
2. Sample $x \in \{\pm 1\}^{n_1}$ uniformly at random and let $x^0 = \frac{x}{\sqrt{n_1}}$.
3. For $i = 1$ to $T/2$ let
$$y^i = \frac{M_{2i-1}^T x^{i-1}}{\|M_{2i-1}^T x^{i-1}\|}; \quad x^i = \frac{M_{2i} y^i}{\|M_{2i} y^i\|}; \quad z^i = \text{sgn}(x^i).$$
4. For each coordinate $j \in [n_1]$ take the majority vote of the signs of z_j^i for all $i \in \{T/4, \ldots, T/2\}$ and call this vector $\overline{v}$:
$$\overline{v}_j = \text{sgn}\left(\sum_{i=T/2}^{T} z_j^i\right).$$
5. Return the partition indicated by $\overline{v}$.

The analysis of the resampled power iteration algorithm proceeds in four phases, during which we track the progress of two vectors x^i and y^i, as measured by their inner product with u and v respectively. We define $U_i := u \cdot x^i$ and $V_i := v \cdot y^i$. Here we give an overview of each phase:

- **Phase 1.** Within $\log n_1$ iterations, $|U_i|$ reaches $\log n_1$. We show that conditioned on the value of U_i, there is at least a $1/2$ chance that $|U_{i+1}| \geq 2|U_i|$; that U_i never gets too small; and that in $\log n_1$ steps, a run of $\log \log n_1$ doublings pushes the magnitude of U_i above $\log n_1$.
- **Phase 2.** After reaching $\log n_1$, $|U_i|$ makes steady, predictable progress, doubling at each step whp until it reaches $\Theta(\sqrt{n_1})$, at which point we say x^i has strong correlation with u.
- **Phase 3.** Once x^i is strongly correlated with u, we show that z^{i+1} agrees with either u or $-u$ on a large fraction of coordinates.
- **Phase 4.** We show that taking the majority vote of the coordinate-by-coordinate signs of z^i over $O(\log n_1)$ additional iterations gives complete recovery whp.

Running time If $n_2 = \Theta(n_1)$, then a straightforward implementation of the algorithm runs in time linear in the number of edges used: each entry of $x^i = My^i$ (resp. $y^i = M^T x^{i-1}$) can be computed as a sum over the edges in the graph associated with M. The rounding and majority vote are both linear in n_1. However, if $n_2 \gg n_1$, then simply initializing the vector y^i will take too much time. In this case, we have to implement the algorithm more carefully.

Say we have a vector x^{i-1} and want to compute $x^i = M_{2i} y^i$ without storing the vector y^i. Instead of computing $y^i = M_{2i-1}^T x^{i-1}$, we create a set $S^i \subset V_2$ of all vertices with degree at least 1 in the current graph G_{2i-1} corresponding to the matrix M_{2i-1}. The size of S^i is bounded by the number of edges in G_{2i-1}, and checking membership can be done in constant time with a data structure of size $O(|S^i|)$ that requires expected time $O(|S^i|)$ to create [21].

Recall that $M_{2i-1} = A_{2i-1} - qJ$. Then we can write
$$y^i = (A_{2i-1} - qJ)^T x^{i-1} = \hat{y} - q\left(\sum_{j=1}^{n_1} x_j^{i-1}\right)\mathbf{1}_{n_2} = \hat{y} - qL\mathbf{1}_{n_2},$$

where $\hat{y}$ is 0 on coordinates $j \notin S^i$, $L = \sum_{j=1}^{n_1} x_j^{i-1}$, and $\mathbf{1}_{n_2}$ is the all ones vector of length n_2. Then to compute $x^i = M_{2i}y^i$, we write

$$\begin{aligned} x^i = (A_{2i} - qJ)y^i &= (A_{2i} - qJ)(\hat{y} - qL\mathbf{1}_{n_2}) \\ &= (A_{2i} - qJ)\hat{y} - qLA_{2i}\mathbf{1}_{n_2} + q^2LJ\mathbf{1}_{n_2} \\ &= A_{2i}\hat{y} - qJ\hat{y} - qLA_{2i}\mathbf{1}_{n_2} + q^2Ln_2\mathbf{1}_{n_1} \end{aligned}$$

We bound the running time of the computation as follows: we can compute $\hat{y}$ in linear time in the number of edges of G_{2i-1} using S^i. Given $\hat{y}$, computing $A_{2i}\hat{y}$ is linear in the number of edges of G_{2i} and computing $qJ\hat{y}$ is linear in the number of non-zero entries of $\hat{y}$, which is bounded by the number of edges of G_{2i-1}. Computing $L = \sum_{j=1}^{n_1} x_j^{i-1}$ is linear in n_1 and gives $q^2Ln_2\mathbf{1}_{n_1}$. Computing $qLA_{2i}\mathbf{1}_{n_2}$ is linear in the number of edges of G_{2i}. All together this gives our linear time implementation.

4 Reduction of planted k-CSP's to the block model

Here we describe how solving the bipartite block model suffices to solve the planted k-CSP problems. Consider a planted k-SAT problem $F_{Q,\sigma}(n, m)$ with distribution complexity r. Let $S \subseteq [k]$, $|S| = r$, be such that $\hat{Q}(S) = \eta \neq 0$. Such an S exists from the definition of the distribution complexity. We assume that we know both r and this set S, as trying all possibilities (smallest first) requires only a constant factor (2^r) more time.

We will restrict each k-clause in the formula to an r-clause, by taking the r literals specified by the set S. If the distribution Q is known to be symmetric with respect to the order of the k-literals in each clause, or if clauses are given as unordered sets of literals, then we can simply sample a random set of r literals (without replacement) from each clause.

We will show that restricting to these r literals from each k-clause induces a distribution on r-clauses defined by $Q^\delta : \{\pm 1\}^r \to \mathbb{R}^+$ of the form $Q^\delta(C) = \delta/2^r$ for $|C|$ even, $Q^\delta(C) = (2-\delta)/2^r$ for $|C|$ odd, for some $\delta \in [0, 2]$, $\delta \neq 1$, where $|C|$ is the number of TRUE literals in C under σ. This reduction allows us to focus on algorithms for the specific case of a parity-based distribution on r-clauses with distribution complexity r.

Recall that for a function $f : \{-1, 1\}^k \to \mathbb{R}$, its Fourier coefficients are defined for each subset $S \subset [k]$ as

$$\hat{f}(S) = \mathop{\mathbb{E}}_{x \sim \{-1,1\}^k}[f(x)\chi_S(x)]$$

where χ_S are the Walsh basis functions of $\{\pm 1\}^k$ with respect to the uniform probability measure, i.e., $\chi_S(x) = \prod_{i \in S} x_i$.

Lemma 1. *If the function $Q : \{\pm 1\}^k \to \mathbb{R}^+$ defines a distribution Q_σ on k-clauses with distribution complexity r and planted assignment σ, then for some $S \subseteq [k]$, $|S| = r$ and $\delta \in [0, 2]\backslash\{1\}$, choosing r literals with indices in S from a clause drawn randomly from Q_σ yields a random r-clause from Q^δ_σ.*

Proof. From Definition 3 we have that there exists an S with $|S| = r$ such that $\hat{Q}(S) \neq 0$. Note that by definition,

$$\begin{aligned} \hat{Q}(S) = \mathop{\mathbb{E}}_{x \sim \{\pm 1\}^k}[Q(x)\chi_S(x)] &= \frac{1}{2^k} \sum_{x \in \{\pm 1\}^k} Q(x)\chi_S(x) \\ &= \frac{1}{2^k} \left(\sum_{x : \in \{\pm 1\}^k : x_S \text{ even}} Q(x) - \sum_{x : \in \{\pm 1\}^k : x_S \text{ odd}} Q(x) \right) \\ &= \frac{1}{2^k} \left(\Pr[x_S \text{ even}] - \Pr[x_S \text{ odd}] \right) \end{aligned}$$

where x_S is x restricted to the coordinates in S, and so if we take $\delta = 1 + 2^k\hat{Q}(S)$, the distribution induced by restricting k-clauses to the r-clauses specified by S is Q^δ_σ. Note that by the definition

of the distribution complexity, $\hat{Q}(T) = 0$ for any $1 \leq |T| < r$, and so the original and induced distributions are uniform over any set of $r-1$ coordinates. □

First consider the case $r = 1$. Restricting each clause to S for $|S| = 1$, induces a noisy 1-XOR-SAT distribution in which a random true literal appears with probability δ and random false literal appears with probability $2 - \delta$. The simple majority vote algorithm described above suffices: set each variable to $+1$ if it appears more often positively than negated in the restricted clauses of the formula; to -1 if it appears more often negated; and choose randomly if it appears equally often. Using $c\sqrt{t \log(1/\epsilon)}$ clauses for $c = O(1/|1-\delta|^2)$ this algorithm will give an assignment that agrees with σ (or $-\sigma$) on $n/2 + t\sqrt{n}$ variables with probability at least $1 - \epsilon$; using $cn \log n$ clauses it will recover σ exactly with probability $1 - o(1)$.

Now assume that $r \geq 2$. We describe how the parity distribution Q_σ^δ on r-constraints induces a bipartite block model. Let V_1 be the set of $2n$ literals of the given variable set, and V_2 the collection of all $(r-1)$-tuples of literals. We have $n_1 = |V_1| = 2n$ and $n_2 = |V_2| = \binom{2n}{r-1}$. We partition each set into two parts as follows: $A_1 \subset V_1$ is the set of false literals under σ, and B_1 the set of true literals. $A_2 \subset V_2$ is the set of $(r-1)$-tuples with an even number of true literals under σ, and B_2 the set of $(r-1)$-tuples with an odd number of true literals.

For each r-constraint $(l_1, l_2, \ldots, l_r)$, we add an edge in the block model between the tuples $l_1 \in V_1$ and $(l_2, \ldots, l_r) \in V_2$. A constraint drawn according to Q_σ^δ induces a random edge between A_1 and A_2 or B_1 and B_2 with probability $\delta/2$ and between A_1 and B_2 or B_1 and A_2 with probability $1 - \delta/2$, exactly the distribution of a single edge in the bipartite block model. Recovering the partition $\mathcal{P}_1 = A_1 \cup B_1$ in this bipartite block model partitions the literals into true and false sets giving σ (up to sign). Now the model in Defn. 2 is that of m clauses selected independently with replacement according to a given distribution, while in Defn. 1, each edge is present independently with a given probability. Reducing from the first to the second can be done by Poissonization; details given in the full version [18].

The key feature of our bipartite block model algorithm is that it uses $\tilde{O}(\sqrt{n_1 n_2})$ edges (i.e. $p = \tilde{O}((n_1 n_2)^{-1/2})$, corresponding to $\tilde{O}(n^{r/2})$ clauses in the planted CSP.

5 Comparison with spectral approach

As noted above, many approaches to graph partitioning problems and planted satisfiability problems use eigenvectors or singular vectors. These algorithms are essentially based on the signs of the top eigenvector of the centered adjacency matrix being correlated with the planted vector. This is fairly straightforward to establish when the average degree of the random graph is large enough. However, in the stochastic block model, for example, when the average degree is a constant, vertices of large degree dominate the spectrum and the straightforward spectral approach fails (see [26] for a discussion and references).

In the case of the usual block model, $n_1 = n_2 = n$, while our approach has a fast running time, it does not save on the number of edges required as compared to the standard spectral approach: both require $\Omega(n \log n)$ edges. However, when $n_2 \gg n_1$, eg. $n_1 = \Theta(n), n_2 = \Theta(n^{k-1})$ as in the case of the planted k-CSP's for odd k, this is no longer the case.

Consider the general-purpose partitioning algorithm of [28]. Let G be the matrix of edge probabilities: G_{ij} is the probability that the edge between vertices i and j is present. Let G_u, G_v denote columns of G corresponding to vertices u, v. Let σ^2 be an upper bound of the variance of an entry in the adjacency matrix, s_m the size of the smallest part in the planted partition, q the number of parts, δ the failure probability of the algorithm, and c a universal constant. Then the condition for the success of McSherry's partitioning algorithm is:

$$\min_{u,v \text{ in different parts}} \|G_u - G_v\|^2 > cq\sigma^2(n/s_m + \log(n/\delta))$$

In our case, we have $q = 4, n = n_1 + n_2, s_m = n_1/2, \sigma^2 = \Theta(p)$, and $\|G_u - G_v\|^2 = 4(\delta-1)^2 p^2 n_2$. When $n_2 \gg n_1 \log n$, the condition requires $p = \Omega(1/n_1)$, while our algorithm succeeds when $p = \Omega(\log n_1/\sqrt{n_1 n_2})$. In our application to planted CSP's with odd k and $n_1 = 2n, n_2 = \binom{2n}{k-1}$, this gives a polynomial factor improvement.

In fact, previous spectral approaches to planted CSP's or random k-SAT refutation worked for even k using $n^{k/2}$ constraints [23, 9, 14], while algorithms for odd k only worked for $k = 3$ and used considerably more complicated constructions and techniques [13, 22, 8]. In contrast to previous approaches, our algorithm unifies the algorithm for planted k-CSP's for odd and even k, works for odd $k > 3$, and is particularly simple and fast.

We now describe why previous approaches faced a spectral barrier for odd k, and how our algorithm surmounts it. The previous spectral algorithms for even k constructed a similar graph to the one in the reduction above: vertices are $k/2$-tuples of literals, and with edges between two tuples if their union appears as a k-clause. The distribution induced in this case is the stochastic block model. For odd k, such a reduction is not possible, and one might try a bipartite graph, with either the reduction described above, or with $\lfloor k/2 \rfloor$-tuples and $\lceil k/2 \rceil$-tuples (our analysis works for this reduction as well). However, with $\tilde{O}(k/2)$ clauses, the spectral approach of computing the largest or second largest singular vector of the adjacency matrix does not work.

Consider M from the distribution $M(p)$. Let u be the n_1 dimensional vector indexed as the rows of M whose entries are 1 if the corresponding vertex is in A_1 and -1 otherwise. Define the n_2 dimensional vector v analogously. The next propositions summarize properties of M.

Proposition 1. $\mathbb{E}(M) = (\delta - 1)puv^T$.

Proposition 2. *Let M_1 be the rank-1 approximation of M drawn from $M(p)$. Then $\|M_1 - \mathbb{E}(M)\| \leq 2\|M - \mathbb{E}(M)\|$.*

The above propositions suffice to show high correlation between the top singular vector and the vector u when $n_2 = \Theta(n_1)$ and $p = \Omega(\log n_1/n_1)$. This is because the norm of $\mathbb{E}(M)$ is $p\sqrt{n_1 n_2}$; this is higher than $O(\sqrt{pn_2})$, the norm of $M - \mathbb{E}(M)$ for this range of p. Therefore the top singular vector of M will be correlated with the top singular vector of $\mathbb{E}(M)$. The latter is a rank-1 matrix with u as its left singular vector.

However, when $n_2 \gg n_1$ (eg. k odd) and $p = \tilde{O}((n_1 n_2)^{-1/2})$, the norm of the zero-mean matrix $M - \mathbb{E}(M)$ is in fact much larger than the norm of $\mathbb{E}(M)$. Letting $x^{(i)}$ be the vector of length n_1 with a 1 in the ith coordinate and zeroes elsewhere, we see that $\|Mx^{(i)}\|_2 \approx \sqrt{pn_2}$, and so $\|M - \mathbb{E}(M)\| = \Omega(\sqrt{pn_2})$, while $\|\mathbb{E}(M)\| = O(p\sqrt{n_1 n_2})$; the former is $\Omega((n_2/n_1)^{1/4})$ while the latter is $O(1)$). In other words, the top singular value of M is much larger than the value obtained by the vector corresponding to the planted assignment! The picture is in fact richer: the straightforward spectral approach succeeds for $p \gg n_1^{-2/3} n_2^{-1/3}$, while for $p \ll n_1^{-2/3} n_2^{-1/3}$, the top left singular vector of the centered adjacency matrix is asymptotically uncorrelated with the planted vector [20]. In spite of this, one can exploit correlations to recover the planted vector below this threshold with our resampling algorithm, which in this case provably outperforms the spectral algorithm.

Acknowledgements

S. Vempala supported in part by NSF award CCF-1217793.

References

[1] E. Abbe, A. S. Bandeira, and G. Hall. Exact recovery in the stochastic block model. *arXiv preprint arXiv:1405.3267*, 2014.

[2] D. Achlioptas and F. McSherry. Fast computation of low rank matrix approximations. In *STOC*, pages 611–618, 2001.

[3] Q. Berthet and P. Rigollet. Complexity theoretic lower bounds for sparse principal component detection. In *COLT*, pages 1046–1066, 2013.

[4] A. Blum. Learning boolean functions in an infinite attribute space. *Machine Learning*, 9:373–386, 1992.

[5] A. Bogdanov and Y. Qiao. On the security of goldreich's one-way function. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*, pages 392–405. 2009.

[6] R. B. Boppana. Eigenvalues and graph bisection: An average-case analysis. In *FOCS*, pages 280–285, 1987.

[7] A. Coja-Oghlan. Graph partitioning via adaptive spectral techniques. *Combinatorics, Probability & Computing*, 19(2):227, 2010.

[8] A. Coja-Oghlan, C. Cooper, and A. Frieze. An efficient sparse regularity concept. *SIAM Journal on Discrete Mathematics*, 23(4):2000–2034, 2010.

[9] A. Coja-Oghlan, A. Goerdt, A. Lanka, and F. Schädlich. Certifying unsatisfiability of random 2k-sat formulas using approximation techniques. In *Fundamentals of Computation Theory*, pages 15–26. Springer, 2003.

[10] A. Daniely, N. Linial, and S. Shalev-Shwartz. More data speeds up training time in learning halfspaces over sparse vectors. In *NIPS*, pages 145–153, 2013.

[11] A. Daniely and S. Shalev-Shwartz. Complexity theoretic limitations on learning dnf's. *CoRR*, abs/1404.3378, 2014.

[12] S. Decatur, O. Goldreich, and D. Ron. Computational sample complexity. *SIAM Journal on Computing*, 29(3):854–879, 1999.

[13] U. Feige and E. Ofek. Easily refutable subformulas of large random 3cnf formulas. In *Automata, languages and programming*, pages 519–530. Springer, 2004.

[14] U. Feige and E. Ofek. Spectral techniques applied to sparse random graphs. *Random Structures & Algorithms*, 27(2):251–275, 2005.

[15] V. Feldman. Attribute efficient and non-adaptive learning of parities and DNF expressions. *Journal of Machine Learning Research*, (8):1431–1460, 2007.

[16] V. Feldman. Open problem: The statistical query complexity of learning sparse halfspaces. In *COLT*, pages 1283–1289, 2014.

[17] V. Feldman, E. Grigorescu, L. Reyzin, S. Vempala, and Y. Xiao. Statistical algorithms and a lower bound for planted clique. In *STOC*, pages 655–664, 2013.

[18] V. Feldman, W. Perkins, and S. Vempala. Subsampled power iteration: a unified algorithm for block models and planted csp's. *CoRR*, abs/1407.2774, 2014.

[19] V. Feldman, W. Perkins, and S. Vempala. On the complexity of random satisfiability problems with planted solutions. In *STOC*, pages 77–86, 2015.

[20] L. Florescu and W. Perkins. Spectral thresholds in the bipartite stochastic block model. *arXiv preprint arXiv:1506.06737*, 2015.

[21] M. L. Fredman, J. Komlós, and E. Szemerédi. Storing a sparse table with 0 (1) worst case access time. *Journal of the ACM (JACM)*, 31(3):538–544, 1984.

[22] J. Friedman, A. Goerdt, and M. Krivelevich. Recognizing more unsatisfiable random k-sat instances efficiently. *SIAM Journal on Computing*, 35(2):408–430, 2005.

[23] A. Goerdt and M. Krivelevich. Efficient recognition of random unsatisfiable k-sat instances by spectral methods. In *STACS 2001*, pages 294–304. Springer, 2001.

[24] M. Kearns. Efficient noise-tolerant learning from statistical queries. *JACM*, 45(6):983–1006, 1998.

[25] S. B. Korada, A. Montanari, and S. Oh. Gossip pca. In *SIGMETRICS*, pages 209–220, 2011.

[26] F. Krzakala, C. Moore, E. Mossel, J. Neeman, A. Sly, L. Zdeborová, and P. Zhang. Spectral redemption in clustering sparse networks. *PNAS*, 110(52):20935–20940, 2013.

[27] L. Massoulié. Community detection thresholds and the weak ramanujan property. In *STOC*, pages 1–10, 2014.

[28] F. McSherry. Spectral partitioning of random graphs. In *FOCS*, pages 529–537, 2001.

[29] E. Mossel, J. Neeman, and A. Sly. A proof of the block model threshold conjecture. *arXiv preprint arXiv:1311.4115*, 2013.

[30] R. O'Donnell and D. Witmer. Goldreich's prg: Evidence for near-optimal polynomial stretch. In *Conference on Computational Complexity*, 2014.

[31] R. Servedio. Computational sample complexity and attribute-efficient learning. *Journal of Computer and System Sciences*, 60(1):161–178, 2000.

[32] S. Shalev-Shwartz, O. Shamir, and E. Tromer. Using more data to speed-up training time. In *AISTATS*, pages 1019–1027, 2012.

[33] V. Vu. A simple svd algorithm for finding hidden partitions. *arXiv preprint arXiv:1404.3918*, 2014.