Derandomized dimensionality reduction with applications

Lars Engebretsen Piotr Indyk Ryan O'Donnell
MIT Laboratory for Computer Science
Cambridge, Massachusetts 02139-3594
E-mail: {enge,indyk,odonnell}@theory.lcs.mit.edu

December 26, 2001

Abstract

The Johnson-Lindenstrauss lemma provides a way to map a number of points in high-dimensional space into a low-dimensional space, with only a small distortion of the distances between the points. The proofs of the lemma are non-constructive: they show that a random mapping induces small distortions with high probability, but they do not construct the actual mapping. In this paper, we provide a procedure that constructs such a mapping deterministically in time almost linear in the number of distances to preserve times the dimension of the original space. We then use that result (together with Nisan's pseudorandom generator) to obtain an efficient derandomization of several approximation algorithms based on semidefinite programming.

1 Introduction

Dimensionality reduction is a general class of techniques which allow one to reduce the dimension of a multidimensional set of data points while preserving some important properties of the set. One of the most fundamental and useful dimensionality reduction techniques is given by the *Johnson-Lindenstrauss lemma (JL lemma)*, which states that any set of n points in l_2^d can be embedded into $l_2^{d'}$ with $d' = O(\log n/\epsilon^2)$ in such a way that the distances between all pairs of points are preserved up to a factor of $1 \pm \epsilon$. Since the original dimension could be as big as n, the JL lemma provides a way to reduce the size of the data set by several orders of magnitude while preserving its distance properties.

The JL lemma was introduced to Theoretical Computer Science in [LLR94] and has since found many algorithmic applications. It is by now a standard way to achieve polynomial [PRTV00, BOR99, GIV01] or even exponential [IM97, Das99] improvements to the complexity of approximate algorithms. Very recently it also became a topic of more applied investigation (e.g., see [IKM00]). In addtion, it has several structural implications; for example, together with Bourgain's lemma [LLR94] it implies that any n-point metric can be embedded into $l_2^{O(\log n)}$ with distortion $O(\log n)$.

Because of its importance, several proofs of this lemma exists in the literature, e.g., see [JL84, FM88, IM97, AV99, DG99, Ach01]; some of them simplify the original proof, others simplify the embedding or improve the $O()$ constant. Unfortunately, none of the existing proofs is constructive. More specifically, all the proofs proceed by showing that a *random* linear mapping (taken from a proper space of mappings) preserves the distances with probability strictly greater than 0. Therefore, the proofs do not lead to a deterministic algorithm for finding such an embedding, but only to a randomized procedure.

In this paper we address this issue and provide a deterministic algorithm which finds a mapping with the properties guaranteed by the JL lemma. The algorithm runs in $O(dm(\log n + 1/\epsilon)^{O(1)})$ time, where $m \leq n^2$ is the number of pairs of distances to preserve. For comparison, note that even *verifying* the quality of a random mapping (and therefore all Las Vegas algorithm for finding a good embedding) would take $O(dm)$ time, which is only slightly less than the running time of our algorithm. As a consequence of our result, we also obtain deterministic procedures for some implications of rhe JL lemma. In particular, since it is known how to embed an n-point metric into l_2 with distortion $O(\log n)$ deterministically (e.g., see [LLR94]), we obtain a deterministic polynomial time algorithm which finds such an embedding into l_2 with dimension $O(\log n)$.

In the second part of this paper, we show that by using the deterministic JL lemma one can efficiently derandomize several approximation algorithms based on semidefinite programming (SDP); in particular, we treat the MAXCUT algorithm of [GW95] and the 3-coloring algorithms of [KMS98]. It was already known that these algorithms could be derandomized in polynomial time; see [MR95]. However, the algorithm therein was extremely complex and had very high running time (around n^{30}). In contrast, our approach is conceptually much simpler, since it decouples the derandomization into the dimensionality reduction step (same for all problems) and enumeration step (problem-dependent but fairly simple). In addition, the running time of our algorithms (excluding the SDP part) is only $O(n^{3+o(1)})$[1]. The derandomization approach we present for the MAXCUT algorithm of [GW95] can be extended to derandomize several other SDP-based approximation algorithms for constraint satisfaction problems. Among other things, our approach works for all constraints that involve only three variables [Zwi98], including MAX 3-SAT [KZ97], MAX k-CUT [FJ97, GW01], and systems of linear equations mod p with two unknowns per equation [AEH01].

1.1 Our techniques

Our deterministic embedding result is obtained by using the method of conditional probabilities. In particular, we apply it to derandomize the proof of JL lemma given in [IM97]. Their proof (described briefly in Preliminaries) is very convenient for our purpose, since it provides us with exact expressions for the values of relevant conditional probabilities. In particular, computing the probability of a single "unsuccessful" event (i.e., that a given pair of points is mapped with large distortion) can be done by evaluating a 3-dimensional integral of a smooth function. This can be easily done in polynomial time (and with polynomially small error) via any numerical integration algorithm, e.g., the trapezoid method. However, in order to obtain a fast algorithm, we show that the integrated function can be well-approximated by a polynomial of polylogarithmic degree. In this way, we can find/update the probability of a single event in polylogarithmic time. Since at any step of the algorithm we consider $O(m)$ events and the algorithm performs $O(d \log^{O(1)} n)$ steps, the running time follows.

The derandomization of SDP-based approximation algorithms is done as follows. First, we compute the solution to the semidefinite program, getting a set of vectors in n-dimensional space. Then, we reduce the dimension to $O(\log n)$ or even $O(1)$ using the deterministic JL lemma; we can show that the reduction approximately preserves the quality of the solution. Finally, we round the vectors to obtain a feasible solution to the combinatorial problem. In the original algorithms [GW95, KMS98] this was done by choosing a random vector r and computing the dot

[1] To make the comparison with [MR95] fair, one should note that our method results in approximation factors $C_1(1-\epsilon)$ for MAXCUT and $C_2^{1+\epsilon}$ for 3-coloring, where C_1 and C_2 are the approximation factors of the randomized algorithms. On the other hand, the algorithms of [MR95] result in approximation factors $C_1(1-1/n)$ and $C_2(1-1/n)$; It is likely that they would be able to obtain our factors in something somewhat faster than in $O(n^{30})$ time.

products of r and the solution vectors. Here, we simulate the random choice of r by enumerating *all* potential vectors. This can be done efficiently, since (a) the dimensionality of the space is low and (b) the number of vectors to consider can be further reduced by showing that r does not have to be "truly random" and that "pseudorandom" vectors obtained via the generator of Nisan [Nis92] are good enough.

1.2 Previous work

In recent years, there has been several results on deterministic embeddings between various spaces. One such result [CCGG98, CCGGP98] is that any n-point metric space can be deterministically embedded into a probabilistic tree metric with low distortion; this improves earlier randomized results of [Bar96, Bar98]. As a consequence, several approximation algorithms based on such embeddings can be derandomized. However, the methods of [CCGG98, CCGGP98] are based on linear programming and cannot be applied in our context.

Another related result (obtained in [I00]) is an explicit embedding of l_2^d into $l_1^{2^{O(\log^2 d)}}$ with distortion $1+1/d^{\Theta(1)}$; earlier, only randomized embeddings of this type had been known. The main tool used in [I00] is the Nisan generator, also used in the enumeration part of our deterministic SDP-based approximation algorithms. However, applying this tool to the JL lemma itself would result in an algorithm with only quasi-polynomial time, $2^{O(\log^2 n)}$, which is much worse than that guaranteed by our algorithm.

2 Preliminaries

The general statement of the JL lemma is as follows.

Lemma 1. *Let $v_1 \dots v_m$ be a sequence of vectors in $\Re^d$ and let $\epsilon, F \in (0,1]$. Then there exists a linear mapping $A : \Re^d \to \Re^k$ where $k = O(\log(1/F)/\epsilon^2)$ such that, for at least a fraction $1-F$ of the m vectors,*

$$k|v_i|^2 \le |Av_i|^2 \le k(1+\epsilon)|v_i|^2.$$

Note that since the mapping A is linear, we can assume without loss of generality that all v_i's are unit vectors. Furthermore, we can get rid of the factor of k by taking $A' = \frac{1}{\sqrt{k}}A$.

Since our deterministic version of the above result is obtained by derandomizing the probabilistic proof from [IM97], we recall the basic components of this proof. Its basic idea is to choose the matrix A at random such that each of its coordinates is chosen independently from $N(0,1)$. By spherical symmetry of the normal distribution it follows that each coordinate of Av_i is also distributed according to $N(0,1)$. Therefore, for each $j = 1 \dots k/2$, the sum of squares of consecutive coordinates $Y_j = |(Av)_{2j-1}|^2 + |(Av)_{2j}|^2$ has exponential distribution with exponent $1/2$. Assuming k is even, the expectation of $L = |Av|^2$ is equal to $\sum_j E[Y_j] = k$. One can show that L is sharply concentrated around its mean (i.e., its value lies in $[1, 1+\epsilon]$ with probability $1-F$). Thus the expected number of v_i's whose length is approximately preserved is at least $(1-F)m$.

We mention that the above proof goes through even if the elements of A are normal variables "truncated" to lie in the interval $[-B, B]$ for $B = O(\sqrt{\log kdm})$. Moreover, it remains correct if the values of A_{ij} are not continuous but are represented using only $O(\log kdm)$ bits.

The Johnson-Lindenstrauss lemma approximately preserves vectors' *lengths*. Since our applications involve semidefinite programming, which is more concerned with vectors' *angles*, we record here a simple lemma relating the two (the proof is presented in the appendix):

Lemma 2. *Suppose that $v_1, \ldots, v_m \in \Re^d$ are unit vectors, and $A : \Re^d \to \Re^k$ is a linear mapping such that $|v|^2 \leq |Av|^2 \leq (1+\epsilon)|v|^2$ for all $v = v_i$ or $v_i - v_j$. Let w_i be the vector Av_i normalized to unit length. Then $w_i \cdot w_j \leq v_i \cdot v_j + \epsilon$ for all i, j.*

3 Deterministic dimensionality reduction

We will derandomize the algorithm described in Preliminaries using the method of conditional probabilities. For $i = 1 \ldots m$ we use E_i to denote the event "$|Av_i|^2 > k(1+\epsilon)$" and E_i' to denote the event "$|Av_i|^2 < k$". We know that $\Pr[E_i], \Pr[E_i'] \leq F/2$ for each i. We exploit the method of conditional probabilities to find A for which the number of unsatisfied events is at most $2Fm$.

The method of conditional probabilities proceeds by setting the random bits (in our case, the bits of A) one by one; each bit is set to the value which minimizes the expected number of unsatisfied events conditioned on that setting. In our case, the order in which we set the bits of the matrix is as follows: we start by setting the bits of A_{11}, $A_{12}, \ldots$ (i.e., the first row), then we set the bits of the elements of the second row of A, and so on. The bits of each element A_{ij} are set by first determining its sign, then its most significant bit, its second significant bit and so on. Note that each setting of bits corresponds to restricting A_{ij} to lie within a certain interval.

It remains to show how to compute the probabilities of events E_i and E_i' conditioned on partial settings of the bits. We focus on one E_i, since the remaining events can be treated similarly. Each event E_i is of the form "$|Av|^2 > t$" where $t = (1+\epsilon)k$. Note that if the rows $1 \ldots s$ are already fully set, then the values of $(Av)_1 \ldots (Av)_s$ are already determined, and thus it suffices to compute the probability of an event "$|A[s+1 \ldots k, \cdot] \cdot v|^2 > t$", where $t = (1+\epsilon)k - |A[1 \ldots s, \cdot] \cdot v|^2$. Therefore, without the loss of generality we can assume that we are setting the first row of the matrix A (say, the element A_{1j}). The value of $|Av|^2$ can be then expressed as a sum of $(v_j X_I + aX')^2 + Z$, where

- X is a random normal variable restricted to an interval I
- X' is a random normal variable, and a is the norm of the vector $v[j+1 \ldots d]$
- Z is a sum of $k-1$ squares of normal variables, i.e., has χ^2 distribution.

Therefore, we need to estimate $\Pr[(v_j X_I + aX')^2 + Z > t]$. To this end we use the following technical lemma (the proof is presented in the appendix):

Lemma 3. *Let X_I be a Gaussian truncated to $I \subseteq [-L, L]$, X' be $N(0,1)$ and Z have χ^2 distribution. Let $W_I = \frac{1}{\sqrt{2\pi}} \int_I e^{-x^2/2}\, dx$. Then, for any $\eta > 0$, it is possible to compute an approximation to*

$$\Pr[(v_j X_I + aX')^2 + Z > t]$$

with additive error

$$\begin{aligned} & \frac{t^{k/2}t^{T_1}}{2^{k/2}\Gamma(k/2)(T_1)!} \\ + & \left(1+\frac{t^{k/2}t^{T_1}}{2^{k/2}\Gamma(k/2)(T_1)!}\right)\left(\frac{2\eta^{2T_2+1}}{(T_2)!}+\frac{e^{-\eta^2/2}}{\eta}\right) \\ + & \left(1+\left(1+\frac{t^{k/2}t^{T_1}}{2^{k/2}\Gamma(k/2)(T_1)!}\right)\left(\frac{2\eta^{2T_2+1}}{(T_2)!}+\frac{e^{-\eta^2/2}}{\eta}\right)\right)\frac{L^{2T_3}}{W_I(T_3)!} \end{aligned}$$

by evaluating a polynomial with $O(T_1^3T_2^3T_3)$ terms.

Now we are ready to show how to implement the method of conditional probabilities in $\tilde{O}(dm)$ time. First, we show how to set the parameters T_1, T_2, T_3 such that the error guaranteed by Lemma 3 is smaller than a parameter $\alpha > 0$. In order to set T_1 observe that $t < 2k$, and therefore $T_1 = O(\log k + \log 1/\alpha)$ is sufficient to keep the first error term smaller than $\alpha/3$. To bound the second term, it is sufficient to set $\eta = O(\sqrt{\log 1/\alpha})$ and $T_2 = \eta$ to make the second term smaller than $\alpha/3$.

Finally, in order to bound the third term, we need to bound W_I from below. To this end, note that if we discretize all values of A with precision up to $\pm 1/D$, then the value of W_I is at least

$$|I| \cdot e^{-\eta^2/2}/\sqrt{2\pi} \geq 1/D \cdot e^{-\eta^2/2}/\sqrt{2\pi}.$$

Therefore, we need $T_3 = O(\log D + \eta^2)$ to make the third term smaller than $\alpha/3$. Thus, the total error term is at most α, while $T_1, T_2, T_3 = O(\log k + \log 1/\alpha + \log D)$.

Unfortunately, we still cannot apply the above estimations directly, since our algorithm does not generate the matrix A with real values, but rather a matrix $\overline{A}$ whose values are truncated to the interval $[-L, L]$ and discretized to be multiples of $1/D$. Therefore, we need to estimate the conditional probabilities over the space induced by $\overline{A}$ instead of A. Note that we can assume that with probability at least $1 - kdN(L) = 1 - O(kde^{-L^2/2})$ each coordinate of A is within $1/D$ from the corresponding coordinate of $\overline{A}$.

Let $\overline{X_I}, \overline{X'}$ and $\overline{Z}$ be the variables corresponding to X_I, X', Z but induced by $\overline{A}$ instead of A. Observe that with probability at least $1 - kdN(L)$ we have $|X_I - \overline{X_I}| \leq 1/D$, $|X' - \overline{X'}| \leq d/D$, $|Z - \overline{Z}| \leq dk/D$. Therefore, for $\delta = O(dk/D)$, we have

$$\begin{aligned} & \Pr[(v_jX_I + aX')^2 + Z > t + \delta] - kdN(L) \\ \leq\ & \Pr[(v_j\overline{X_I} + a\overline{X'})^2 + \overline{Z} > t] \\ \leq\ & \Pr[(v_jX_I + aX')^2 + Z > t - \delta] + kdN(L) \end{aligned}$$

Now we use the following fact:

Fact 1. *There is a constant C such that for any $\delta > 0$ we have*

$$\left|\Pr[(v_jX_I + aX')^2 + Z > t - \delta] - \Pr[(v_jX_I + aX')^2 + Z > t + \delta]\right| \leq C\delta$$

Therefore, the difference between the probabilities induced by A and $\overline{A}$ is at most $O(kd(e^{-L^2/2} + 1/D))$. By setting $D = O(\frac{\alpha}{kd})$ and $L = O(\sqrt{\log 1/\alpha + \log k + \log d})$ we make the error less than α.

Since our algorithm takes $O(kd \log D)$ steps, the sum of the probabilities $\Pr[E_i]$ at the end of the algorithm is at most $Fm(1 + O(\alpha kd \log D))$. By setting $\alpha = 1/\Theta(kd \log kd)$ we can make it at most $2Fm$.

It is easy to verify that the whole derandomization procedure can be implemented to run in $\tilde{O}(dm)$ time. Thus we have proven the following theorem.

Theorem 1. *Let $v_1 \ldots v_m$ be a sequence of vectors in $\Re^d$ and let $\epsilon, F \in (0, 1]$. Then in $O(dm(\log n + 1/\epsilon)^{O(1)})$ deterministic time we can compute a linear mapping $A : \Re^d \to \Re^k$ where $k = O(\log(1/F)/\epsilon^2)$ such that*

$$k|v_i|^2 \leq |Av_i|^2 \leq k(1 + \epsilon)|v_i|^2$$

for all at least a fraction $1 - F$ of i's.

4 Derandomized rounding

In this section we show how to use the deterministic dimensionality reduction technique to derandomize several known approximation algorithms based on semidefinite programming, including the ones for MAXCUT and for coloring 3-colorable graphs. The vast majority of such algorithms do the following:

1. Solve (deterministically) a semidefinite program, obtaining n vectors in $\Re^n$
2. Round the vectors into integers, obtaining a combinatorial solution to the problem

Since the first step is already deterministic, it is sufficient to focus on the second step. The implementation of the rounding procedures depends on the actual problem; however, most share a common essential subprocedure: picking a random spherically symmetric vector r, and then selecting all solution vectors v such that $v \cdot r \geq c$ for some constant c. For example, the rounding procedure for MAXCUT in [GW95] picks one random vector r, and then selects all vectors v satisfying $v \cdot r \geq 0$, putting them on one side of the cut. To color 3-colorable graphs, [KMS98] repeatedly find large independent sets by considering all vectors v such that $v \cdot r \geq c$, where c is a constant depending on the graphs' maximum degree.

Our goal is to give a fast deterministic algorithm for performing such selection procedures. An obvious approach would be to enumerate "all" vectors r (e.g., all points in an ϵ-net for the n-dimensional sphere) and choose the one giving the "best" selection of vectors. While this would work, its running time would be exponential in the dimension.

In order to reduce the running time, we use the following two tools:

- Dimensionality reduction: we show that if we reduce the dimension of the solution vectors from n to the much smaller value k, the quality of solution does not degrade significantly. For example, one can reduce the dimension of MAXCUT vectors to a *constant* and still preserve the approximation factor up to arbitrary multiplicative factor of $1 + \epsilon$. For coloring, we can reduce the dimension to $O(\log n)$ and change the approximation factor only slightly. Then, enumerating "all" vectors in the new k-dimensional space takes time only polynomial in n (although the exponent of the polynomial could be very high)
- Small sample spaces: we show that instead of choosing r at random from the set of "all" k-dimensional vectors, we can choose it from a much smaller subset while almost preserving the expected quality of the solution. In particular, we use the fact that the two key operations used

in rounding (namely generating a random vector from the k-dimensional normal distribution, and computing the dot product of two k-dimensional vectors) can be done using $\log^{O(1)} k$ space. Therefore, we can use Nisan's generator [Nis92] in order to generate the necessary $k^{O(1)}$ pseudorandom bits from a truly random seed of length $\log^{O(1)} k$. This reduces the number of potential vectors r to $2^{\log^{O(1)} k}$, which is sublinear if $k = \Theta(\log n)$.

Before going on, let us briefly explain what we mean when we speak of enumerating "all" vectors in k-dimensional space. In a typical randomized selection procedure, one picks a spherically symmetric vector $r \in \Re^k$ according to the k-dimensional normal distribution. Since the usual model of randomness involves getting independent random 0/1 bits, generating a vector from the true normal distribution is not, strictly speaking, possible. Implicit in previously published randomized selection procedures is an algorithm for generating an *approximately* normally distributed vector r, using a reasonable number of random bits — say $k^{O(1)}$. When we enumerate "all" vectors in k-dimensional space, we are actually enumerating all possible $2^{k^{O(1)}}$ vectors r' given by this approximating algorithm. Also implicit in such algorithms is the idea that using a vector which is only approximately normally distributed does not introduce much error. Since we are going to need accurate estimates of the number of random bits needed and the amount of error introduced, we give the following technical result, whose proof is deferred to the appendix.

Lemma 4. *For every dimension k and error parameter $\delta > 0$, there is a distribution X' which can be sampled using $O(\log(k/\delta))$ random bits, $O(\log(k/\delta))$ space, and $O(\frac{1}{\delta}\log^2(k/\delta))$ time, and is close to the 1-dimensional normal distribution in the following sense. Let $v, w \in \Re^k$ be unit vectors, and let $c \in \Re$. Consider the randomized algorithm $\mathcal{A}_{v,c}$ which picks a random vector r in $\Re^k$ by choosing each coordinate independently from the 1-dimensional normal distribution, and then outputs 1 or 0 depending on whether $v \cdot r \geq c$. If $\mathcal{A}_{v,c}$ instead uses the approximate distribution X', its output distribution is the same within additive error $\pm\delta$. A similar statement holds for the algorithm $\mathcal{B}_{v,w}$, which outputs 1 or 0 depending on whether $sgn(v \cdot r) = sgn(w \cdot r)$.*

In the following sections, we first show how to use the first tool to derandomize the MAXCUT algorithm (since we reduce the dimension to a constant, using small sample spaces to reduce the running time is not necessary). Then we proceed with the coloring algorithm, which uses both the dimensionality reduction and small sample space techniques.

4.1 Derandomization of the Goemans-Williamson MAXCUT algorithm

To get a deterministic version of the Goemans-Williamson (GW) MAXCUT algorithm which runs in small polynomial time, it suffices to use just the first technique mentioned, namely dimensionality reduction. In this section we show how we can reduce the dimension of the solution space to a constant without losing much quality; at this point, the naive derandomization of the randomized selection procedure is still fast.

Recall the GW algorithm, which takes a graph $G = (V, E)$ containing n vertices and m edges[2] and tries to find a large cut. First the GW algorithm uses semidefinite programming to maximize the objective function $\sum_{(i,j)\in E}(1 - v_i \cdot v_j)/2$ over the set of all unit vectors $v_1, \ldots, v_n$ in $\Re^n$. Then the algorithm picks a random spherically symmetric vector r, and returns the cut consisting of all vertices i such that $v_i \cdot r \geq 0$. [GW95] shows that the expected number of edges cut is within a factor of $\rho \approx 0.87$ of the largest cut.

Our rounding procedure will return a cut which is within a factor $\rho - O(\epsilon)$ of optimal, and will run in deterministic time $2^{O(\log^2(1/\epsilon)/\epsilon^3)}m$.

[2]For simplicity, we consider here only the unweighted case; the weighted case can be treated in a similar way.

The main step is to use the derandomized Johnson-Lindenstrauss algorithm from Theorem 1 to reduce the dimension of the optimal v_i's to $k = O(\log(1/\epsilon)/\epsilon^2)$ while preserving at least a $1-\epsilon$ fraction of the dot products $v_i \cdot v_j$, up to a factor of $1+\epsilon$. We show that the dimensionality reduction can change the value of the objective function by at most a factor of $1 - O(\epsilon)$. Once all the vectors live in k-dimensional space, the second step is to derandomize the selection procedure by considering all possible vectors that could arise in the randomized choice of a vector from the k-dimensional normal distribution.

In the following we describe both steps in more detail.

Step 1: dimensionality reduction. We invoke Theorem 1 with parameters $F = \epsilon$, allowable distortion $(1+\epsilon)$ and set of vectors $V \cup V'$, where:

- V is a *multiset* containing m/n copies of each vector v_i, and
- V' is the set of all vectors $v_{ij} = v_i - v_j$, for $(i,j) \in E$.

Note that $|V| = |V'| = m$.

The derandomized Johnson-Lindenstrauss procedure returns a mapping A of $\Re^n$ into $\Re^k$ ($k = O(\log(1/\epsilon)/\epsilon^2)$) such that:

- for at least a fraction $1-2F$ of v_i's we have $1 \le |Av_i|^2 \le 1+\epsilon$
- for at least a fraction $1-2F$ of vectors v_{ij} we have $|v_{ij}|^2 \le |Av_{ij}|^2 \le (1+\epsilon)|v_{ij}|^2$

Let E' denote the set of edges (i,j) such that the above good preservation events happen for all of v_i, v_j, and v_{ij}. E' consists of at least a $1-6F$ fraction of the edges.

If we now set w_i to be the normalization of Av_i to unit length, Lemma 2 lets us conclude that $w_i \cdot w_j \le v_i \cdot v_j + \epsilon$ for all $(i,j) \in E'$. Hence we now have a set of unit vectors in $\Re^k$ such that

$$\sum_{(i,j)\in E'} (1 - w_i \cdot w_j)/2 \ge \sum_{(i,j)\in E'} (1 - v_i \cdot v_j - \epsilon)/2 \ge \sum_{(i,j)\in E} (1 - v_i \cdot v_j)/2 - m(\epsilon/2 + 6F)$$

It is well known that in the optimal solution, $\sum_{(i,j)\in E}(1 - v_i \cdot v_j)/2 \ge m/2$. Therefore

$$\sum_{(i,j)\in E'} (1 - w_i \cdot w_j)/2 \ge (1 - 13\epsilon) \sum_{(i,j)\in E} (1 - v_i \cdot v_j)/2 \ge (1 - 13\epsilon)OPT.$$

Step 2: rounding.

Suppose we now round as in [GW95], forming the cut by picking a random spherically symmetric vector $r \in \Re^k$ and splitting the vertices i according to $\mathrm{sgn}(w_i \cdot r)$. Then the expected value of the cut is $\sum_{(i,j)\in E} \arccos(w_i \cdot w_j)/\pi \ge \sum_{(i,j)\in E'} \arccos(w_i \cdot w_j)/\pi$. We can't pick r from the true normal distribution, but by Lemma 4 we can use X' to get the probabilities close enough. I.e., the expected value of the cut using X' is at least

$$\left(\sum_{(i,j)\in E'} \arccos(w_i \cdot w_j)/\pi\right) - m\delta \ge \rho\left(\sum_{(i,j)\in E'} (1 - w_i \cdot w_j)/2\right) - m\delta$$
$$\ge \rho(1 - 13\epsilon)OPT - m\delta \ge \rho(1 - 13\epsilon - 2\delta)OPT$$

where $\rho \approx 0.87$, the first inequality using the analysis of [GW95], the last inequality using $OPT \ge m/2$. Now if we take $\delta = \epsilon$, we end up with a $(\rho - O(\epsilon))$-approximation algorithm, as desired.

Recall that sampling X' requires $O(\log(k/\delta))$ random bits and $O(\frac{1}{\delta}\log^2(k/\delta))$ time. With $k = O(\log(1/\epsilon)/\epsilon^2))$ and $\delta = \epsilon$, we see that this is $R := O(\log(1/\epsilon))$ random bits and $T := O(\log^2(1/\epsilon)/\epsilon)$ time per coordinate. Enumerating over all choices of random bits for each coordinate leads to a total time of $(2^R T)^k = 2^{O(\log^2(1/\epsilon)/\epsilon^3)}$. Hence we get the claimed deterministic time bound for the whole algorithm.

We mention in passing that using the small sample techniques explained in the next section, this running time may be reduced to $2^{\log^{O(1)}(1/\epsilon)}m$.

4.2 Derandomization of the Karger-Motwani-Sudan coloring algorithm

In [KMS98], Karger, Motwani, and Sudan give an algorithm for coloring 3-colorable graphs on n vertices, using $\tilde{O}(n^{1/4})$ colors. The main part of this result involves showing how to color such a graph using $\tilde{O}(\Delta^{1/3})$ colors in such a way that only $n/4$ edges have both endpoints the same color. (Here Δ is the maximum degree of the graph.) Using [Wig83] it is straightforward to show that such "semicolorings", using $\tilde{O}(\Delta^a)$ colors, lead to fast proper coloring algorithms using $\tilde{O}(n^{a/(1+a)})$ colors.

In order to find a semicoloring using a small number of colors, [KMS98] take the given 3-colorable graph $G = (V, E)$ with maximum degree Δ, and solve an associated semidefinite program. This yields a set of unit vectors $\{v_i\}_{i \in V} \subseteq \Re^n$ with the property that if $(i, j) \in E$, then $v_i \cdot v_j \leq -1/2$. At this point, [KMS98] describe two randomized rounding techniques which give a semicoloring.

The first, simpler, technique involves selecting $t = 2 + \log_3 \Delta$ random spherically symmetric vectors $r_1, \ldots, r_t$. Each vertex v_i is then given a t-bit color, dictated by the values of $\mathrm{sgn}(v_i \cdot r_j)$. This leads to $O(\Delta^{\log_3 2})$ colors, and fewer than $n/4$ monochromatic edges in expectation.

The second technique involves selecting just one random spherically symmetric vector r, and then considering all v_i such that $v_i \cdot r \geq c$, where $c = \sqrt{\frac{2}{3}\Delta}$. Their analysis shows that the subgraph induced by these vertices has $\tilde{\Omega}(n/\Delta^{1/3})$ more vertices than edges in expectation. This leads to an independent set of roughly the same size. Repeatedly extracting such independent sets gives a semicoloring using $\tilde{O}(\Delta^{1/3})$ colors.

We start by derandomizing the first, simpler, technique. As mentioned, there are two steps involved: dimensionality reduction, and small sample spaces.

Step 1: dimensionality reduction.

As in the MAXCUT algorithm, after we solve the coloring semidefinite program yielding $v_1, \ldots, v_n \in \Re^n$, we reduce the dimension of the space in which these vectors lie. Specifically, we invoke Theorem 1 with parameters $F = 1/n$, distortion parameter ϵ, and set of vectors $\{v_i\}_{i \in V} \cup \{v_i - v_j\}_{i,j \in V}$. The result is a new set of vectors in dimension $k = O(\log n/\epsilon^2)$, in which *every* vector's length is approximately preserved. Normalize these vectors to have unit length, yielding $w_1, \ldots, w_n \in \Re^k$. By Lemma 2, $w_i \cdot w_j \leq v_i \cdot v_j + \epsilon$. The key property of the semidefinite program is that when $(i, j) \in E$, $v_i \cdot v_j \leq -1/2$. Hence the unit vectors w_i have the property that if $(i, j) \in E$, then $w_i \cdot w_j \leq -1/2 + \epsilon$.

At this point, we might try to derandomize the rounding procedures from [KMS98] by again trying "all" rounding vectors $r \in \Re^k$. However, each of the k coordinates of r requires random bits, and so the time taken would be at least 2^k. This is polynomial since $k = O(\log n/\epsilon^2)$, but it could be an extremely large polynomial. We would like to do better. To this end, we show how small sample spaces can decrease the number of rounding vectors r we need to try.

Step 2: small sample spaces.

Consider the algorithm $\mathcal{A}_{v,c}$ from Lemma 4. It uses a random spherically symmetric vector r and calculates the dot product of r with v, returning 1 iff $v \cdot r \geq c$. Notice that this calculation can be performed in a very small amount of space if the coordinates of r are presented in an online fashion. Namely, it only needs space which is polynomial in the size of one coordinate of v or r. Hence, if the coordinates of r are generated online using the algorithm for sampling from distribution X', then $\mathcal{A}_{v,c}$ becomes an algorithm using $R := O(k \log(k/\delta))$ random bits and $S := \log^{O(1)}(k/\delta)$ total space.

Hence, the results of Nisan ([Nis92]) allow us to build a pseudorandom generator using small-length seed which fools every algorithm $\mathcal{A}_{v,c}$. Specifically, the length of the seed is $O(S \log R)$, which is still $\log^{O(1)}(k/\delta)$. The additive error that Nisan incurs in the output distribution is only $2^{-S} \leq \delta$. Since we already incur an error of δ for using X' instead of the true k-dimensional normal distribution, this is not a problem.

Summarizing, we now have a fast algorithm (time $O(\frac{1}{\delta}\log^2(k/\delta))$) for generating a nearly spherically symmetric vector r using only $\log^{O(1)}(k/\delta)$ random bits, such that for all unit vectors v and numbers c, the probability that $v \cdot r \geq c$ is within $\pm 2\delta$ of the desired probability.

At this point, we can enumerate over all possible seeds in time $2^{\log^{O(1)}(k/\delta)}$, which is sublinear in n when ϵ and δ are constant, since $k = O(\log n/\epsilon^2)$.

Let's see why this is enough to give a fast derandomization of the first, simple rounding procedure of [KMS98]. Suppose we pick the rounding vectors of the simple procedure one at a time, at each step partitioning the vertices into two sets and thus adding a bit to the color label of each vertex. Let $(i,j) \in E$. When r is picked from the true normal distribution, the probability that both $w_i \cdot r$ and $w_j \cdot r$ have different signs is equal to $\arccos(w_i \cdot w_j)/\pi$. By Lemma 2, $w_i \cdot w_j \leq -1/2 + \epsilon$. Hence, $\arccos(w_i \cdot w_j)/\pi \geq \arccos(-1/2+\epsilon)/\pi \geq 2/3 - O(\epsilon)$. We conclude that the probability a particular edge becomes cut given the new color bit assigned is at least $2/3 - O(\epsilon)$.

We don't actually have truly normal rounding vectors. However Lemma 4 tells us that we only lose an additive factor of $O(\delta)$ in this estimation. So by taking $\delta = \epsilon$, we conclude that the probability a random output of our pseudorandom generator causes a particular edge to be cut is at least $2/3 - O(\epsilon)$. Hence if we start with m uncut edges, rounding with a random output of the generator gives us at most $(1/3 + O(\epsilon))m$ uncut edges in expectation. We can enumerate all outputs of the generator in sublinear time, and thus find a particular vector such that rounding with it leads to at most $(1/3 + O(\epsilon))m$ uncut edges.

Now we simply repeat this process. After $t := \log_{3-O(\epsilon)} \Delta + O(1)$ rounds we will go from m edges down to less than $n/4$, and hence have a semicoloring. This leads to $2^t = O(\Delta^{\log_{3-O(\epsilon)} 2})$ colors. I.e., we have a fast derandomization of the first rounding scheme of [KMS98].

In order to derandomize the second rounding procedure of [KMS98], we have to be a little more careful. This is because the analysis depends on very small quantities, which are overwhelmed when the error parameter δ is a constant. In the simple rounding procedure, the color classes are essentially assigned one bit at a time. In the improved rounding procedure, all bits of a color class are found at once. If we take the intermediate road of assigning color classes in blocks of constantly many bits, then we can get arbitrarily close to the desired $O(\Delta^{1/3})$-semicoloring, in such a way that the error parameter δ does not overwhelm the probabilities. We leave the details to the appendix.

References

[Ach01] D. Achlioptas. Database-friendly random projections. *Principles of Database Systems*, 2001.

[AEH01] G. Andersson, L. Engebretsen, and J. Håstad. A new way of using semidefinite programming with applications to linear equations mod p. *Journal of Algorithms*, 39(2):162–204, 2001.

[AV99] R. I. Arriaga and S. Vempala. Algorithmic theories of learning. *Foundations of Computer Science*, 1999.

[Bar96] Y. Bartal. Probabilistic approximation of metric spaces and its algorithmic applications. *Foundations of Computer Science*, 1996.

[Bar98] Y. Bartal. On approximating arbitrary metrices by tree metrics. *Symposium on Theory of Computing*, 1998.

[BOR99] A. Borodin, R. Ostrovsky, Y. Rabani. Subquadratic approximation algorithms for clustering problems in high dimensional spaces. *Symposium on Theory of Computing*, 1999.

[CCGG98] M. Charikar and C. Chekuri and A. Goel and S. Guha. Rounding via trees: deterministic approximation algorithms for group Steiner trees and k-Median. *Symposium on Theory of Computing*, 1998.

[CCGGP98] M. Charikar and C. Chekuri and A. Goel and S. Guha and S. Plotkin. Approximating a finite metric by a small number of tree metrics. *Foundations of Computer Science*, 1998.

[Das99] S. Dasgupta. Learning mixtures of Gaussians. *Foundations of Computer Science*, 1999.

[DG99] S. Dasgupta and A. Gupta. An elementary proof of the Johnson-Lindenstrauss lemma. *ICSI technical report TR-99-006, Berkeley, CA*, 1999.

[Fel62] William Feller. *An Introduction to Probability Theory and Its Applications*, volume 1. John Wiley & Sons, New York, second edition, 1962.

[FM88] P. Frankl and H. Maehara. The Johnson-Lindenstrauss lemma and the sphericity of some graphs *Journal of Combinatorial Theory B*, 44(1988), pp. 355-362.

[FJ97] Alan Frieze and Mark Jerrum. Improved approximation algorithms for MAX k-CUT and MAX BISECTION. *Algorithmica*, 18:67–81, 1997.

[GIV01] A. Goel, P. Indyk, and K. Varadarajan. Reductions among high dimensional proximity problems. *Symposium on Discrete Algorithms*, 2001.

[MR95] S. Mahajan and H. Ramesh. Derandomizing semidefinite programming based approximation algorithms. *Foundations of Computer Science*, 1995.

[GR94] I.S. Gradsteyn and I.M. Ryzhik. *Table of Integrals, Series, and Products.* Academic Press, Boston, fifth edition, January 1994.

[GW95] Michel X. Goemans and David P. Williamson. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *Journal of the ACM*, 42(6):1115–1145, November 1995.

[GW01] Michel X. Goemans and David P. Williamson. Approximation algorithms for MAX-3-CUT and other problems via complex semidefinite programming. *Symposium on Theory of Computing*, 2001.

[I00] P. Indyk. Stable distributions, pseudorandom generators, embeddings and data stream computation *Foundations of Computer Science*, 2000.

[IKM00] P. Indyk and N. Koudas and S. Muthukrishnan. Identifying representative trends in massive time series datasets using sketches. *International Conference on Very Large Databases (VLDB)*, 2000.

[IM97] Piotr Indyk and Rajeev Motwani, Approximate nearest neighbors: towards removing the curse of dimensionality. *Symposium on Theory of Computing*, 1997.

[JL84] W.B. Johnson and J. Lindenstrauss. Extensions of Lipshitz mapping into Hilbert space. *Contemporary Mathematics*, 26:189–206, 1984.

[KMS98] David Karger, Rajeev Motwani and Madhu Sudan. Approximate graph coloring by semidefinite programming. *Journal of the ACM*, 45(2):246–265, March 1998.

[KZ97] H. Karloff and U. Zwick. A 7/8-approximation algorithm for MAX 3SAT? *Foundations of Computer Science*, 1997.

[LLR94] N. Linial, E. London, and Y. Rabinovich. The geometry of graphs and some of its algorithmic applications. *Foundations of Computer Science*, pages 577–591, 1994.

[Nis92] Noam Nisan, Pseudorandom sequences for space bounded computations. *Combinatorica*, 12:449–461, 1992.

[PRTV00] C. Papadimitriou and P. Raghavan and H. Tamaki and S. Vempala. Latent semantic indexing: a probabilistic analysis. *Symposium on the Principles of Database Systems*, 1998.

[Wig83] Avi Wigderson. Improving the performance guarantee for approximate graph coloring. *Journal of the ACM*, 30:729–735, 1983.

[Zwi98] U. Zwick. Approximation algorithms for constraint satisfaction problems involving at most three variables per constraint. *Symposium on Discrete Algorithms*, 1998

A Proof of Lemma 2

Proof of Lemma 2. This lemma states that if A approximately preserves the lengths of all v_i's and $(v_i - v_j)$'s, then it also does not shrink the angle between vectors by more than a small additive amount.

Note that $w_i \cdot w_j = (Av_i \cdot Av_j)/|Av_i||Av_j| \leq Av_i \cdot Av_j$, since v_i and v_j are unit vectors. Hence it suffices to show that $Av_i \cdot Av_j \leq v_i \cdot v_j + \epsilon$. Using the relation $|x-y|^2 = |x|^2 - 2x \cdot y + |y|^2$, we get:

$$\begin{aligned} 2Av_i \cdot Av_j &= |Av_i|^2 + |Av_j|^2 - |Av_i - Av_j|^2 \\ &\leq (1+\epsilon)|v_i|^2 + (1+\epsilon)|v_j|^2 - |A(v_i - v_j)|^2 \\ &= 2 + 2\epsilon - |A(v_i - v_j)|^2 \\ &\leq 2 + 2\epsilon - |v_i - v_j|^2 \\ &\leq 2 + 2\epsilon - (|v_i|^2 - 2v_i \cdot v_j + |v_j|^2) \\ &= 2\epsilon + 2v_i \cdot v_j \end{aligned}$$

and we are done. □

B Proof of Lemma 3

Proof of Lemma 3. Let the probability densities of X_I, X' and Z, respectively, be f_{X_I}, $f_{X'}$ and f_Z, respectively. Then the above probability can be written

$$\int_{x \in I} \int_{x'=-\infty}^{\infty} \Pr[Z > t - (v_j x + ax')^2] f_{X_I}(x) f_{X'}(x')\, dx\, dx'$$

Since $\Pr[Z > z] = 1$ as soon as $z < 0$, we can rewrite the above integral as $1 - I_1$, where

$$I_1 = \iint_{(x,x') \in A} \Pr[Z \leq t - (v_j x + ax')^2] f_{X_I}(x) f_{X'}(x')\, dx\, dx',$$

and $A = \{(x, x') : x' \in I \wedge (v_j x + ax')^2 \leq t\}$. Let us now start by evaluating

$$\Pr[Z \leq t - (v_j x + ax')^2] = \int_0^{t-(v_j x + ax')^2} f_Z(z)\, dz$$

By Proposition 1, we can approximate

$$f_Z(z) = \frac{z^{k/2-1} e^{-z/2}}{2^{k/2}\Gamma(k/2)} = \frac{(z/2)^{k/2-1} e^{-z/2}}{2\Gamma(k/2)}$$

within an additive error of

$$E_1 = \frac{t^{k/2-1} t^{T_1}}{2^{k/2}\Gamma(k/2)(T_1)!}$$

by a polynomial with T_1 terms. Thus,

$$\int_0^{t-(v_j x + ax')^2} f_Z(z)\, dz$$

can be expressed as a polynomial in $\sqrt{t - (v_j x + ax')^2}$ with $O(T_1)$ terms, within an additive error of tE_1. Let us denote this polynomial by P. It then remains to calculate

$$I_2 = \iint_{(x,x') \in A} P\left(\sqrt{t - (v_j x + ax')^2}\right) f_{X_I}(x) f_{X'}(x')\, dx\, dx'.$$

Since $\iint_{(x,x')\in A} f_{X_I}(x) f_{X'}(x')\,dx\,dx' \le 1$, the above integral approximates I_1 within an additive error of tE_1, i.e., $|I_1 - I_2| \le tE_1$. We now compute an approximation to I_2. Let us rewrite I_2 as

$$\begin{aligned} I_2 &= \int_{x\in I} f_X(x)\,dx \int_{x'=(-\sqrt{t}-v_jx)/a}^{(\sqrt{t}-v_jx)/a} P\Big(\sqrt{t-(v_jx+ax')^2}\Big) f_{X'}(x')\,dx' \\ &= \int_{x\in I} I_3(x) f_X(x)\,dx. \end{aligned}$$

Substitute $v_jx + ax' = y$ in the inner integral $I_3(x)$. This gives integration in the interval $|y| \le \sqrt{t}$, i.e.,

$$I_3(x) = \frac{1}{a}\int_{y=-\sqrt{t}}^{\sqrt{t}} P\Big(\sqrt{t-y^2}\Big) f_{X'}\big((y-v_jx)/a\big)\,dy. \tag{1}$$

Since $f_{X'}$ decreases rapidly, it is enough to integrate over the y such that the argument to $f_{X'}$ is close to zero. More precisely, let

$$Y = \Big\{y \;\Big|\; y^2 \le \sqrt{t}\Big\} \cap \Big\{y \;\Big|\; \frac{|y-v_jx|}{a} \le \eta\Big\}.$$

Since $f_{X'}$ is the probability density for the normal distribution, the error we introduce by computing

$$I_4(x) = \frac{1}{a}\int_{y\in Y} P\Big(\sqrt{t-y^2}\Big) f_{X'}\big((y-v_jx)/a\big)\,dy$$

instead of I_3 is bounded by the maximum value of P in the interval times the weight of tails of the normal distribution, which gives that

$$\begin{aligned} |I_3(x) - I_4(x)| &\le \frac{1}{a}\big(1+tE_1\big)\int_{|y-v_jx|/a\ge\eta} f_{X'}\big((y-v_jx)/a\big)\,dy \\ &= \big(1+tE_1\big)\int_{|s|\ge\eta} f_{X'}(s)\,ds \\ &\le \big(1+tE_1\big)\frac{e^{-\eta^2/2}}{\eta}. \end{aligned}$$

Moreover, for $y \in Y$, $f_{X'}$ can be approximated within additive error $E_2 = \eta^{2T_2}/(T_2)!$ by a polynomial with T_2 terms by Proposition 2. Let us denote this polynomial by Q. Since $0 < P(\cdot) < 1$, up to an additive error of tE_1, this implies that we can compute an approximation to $I_4(x)$ by computing

$$I_5(x) = \frac{1}{a}\int_{y\in Y} P\Big(\sqrt{t-y^2}\Big) Q\big((y-v_jx)/a\big)\,dy.$$

The error introduced by this approximation is

$$\begin{aligned} |I_4(x) - I_5(x)| &\le (1+tE_1)\int_{|s|\le\eta} \big|f_{X'}(s) - Q(s)\big|\,ds \\ &\le (1+tE_1)\cdot 2\eta E_2, \end{aligned}$$

which implies that

$$|I_3(x) - I_5(x)| \le \big(1+tE_1\big)\left(2\eta E_2 + \frac{e^{-\eta^2/2}}{\eta}\right).$$

We now argue that $I_5(x)$ can be expressed as a polynomial with few terms. Note that

$$P\left(\sqrt{t-y^2}\right) = P_1(y) + \sqrt{t-y^2}P_2(y),$$

where P_1 and P_2 are polynomials with $O(T_1^2)$ terms and degree $O(T_1)$. Also, Q contains $O(T_2^2)$ terms ans has degree $O(T_2)$ when the powers of $(y - v_j x)/a$ have been expanded. Thus,

$$I_5(x) = \frac{1}{a}\int_{y\in Y}\Big(P_1(y)Q(x,y) + \sqrt{t-y^2}P_2(y)Q(x,y)\Big)dy.$$

By Proposition 3,

$$\int_{y\in Y} P_1(y)Q(x,y)\,dy$$

is a polynomial with $O(T_1^2T_2^2)$ terms since $P_1(y)Q(x,y)$ has $O(T_1^2T_2^2)$ terms; by By Proposition 4,

$$\int_{y\in Y} \sqrt{t-y^2}P_2(y)Q(x,y)\,dy$$

is a polynomial with $O(T_1^3T_2^3)$ terms since $P_2(y)Q(x,y)$ has $O(T_1^2T_2^2)$ terms and degree $O(T_1T_2)$. Thus, $I_5(x)$ can be expressed as a polynomial in x with $O(T_1^3T_2^3)$ terms; let us denote this polynomial by R. Finally, we can compute

$$I_6 = \int_{x\in I} R(x)f_X(x)\,dx,$$

since f_X is a probability distribution,

$$|I_2 - I_6| \le \int_{x\in I}\big|I_3(x) - I_5(x)\big|f_X(x)\,dx \le \max\big|I_3(x) - I_5(x)\big|.$$

To compute an approximation to I_6, note that since

$$f_{X_I}(x) = \frac{e^{-x^2/2}/\sqrt{2\pi}}{W_I},$$

we can approximate f_{X_I} within additive error

$$E_3 = \frac{L^{2T_3}}{W_I(T_3)!}$$

by a polynomial with T_3 terms by Proposition 2. Since $0 \le R(\cdot) \le 1$ within an additive error $\max\big|I_3(x) - I_5(x)\big|$, this implies that I_6 can be approximated within additive error

$$\Big(1 + \max\big|I_3(x) - I_5(x)\big|\Big)E_3$$

by integrating a polynomial with $O(T_1^3T_2^3T_3)$ terms, let us denote the resulting polynomial by I_7.

To sum up, $|I_1 - I_2| \leq \epsilon_1$, $|I_2 - I_6| \leq \epsilon_2$, and $|I_6 - I_7| \leq \epsilon_3$, where

$$\epsilon_1 = \frac{t^{k/2} t^{T_1}}{2^{k/2}\Gamma(k/2)(T_1)!},$$
$$\epsilon_2 = \big(1+\epsilon_1\big)\left(\frac{2\eta^{2T_2+1}}{(T_2)!} + \frac{e^{-\eta^2/2}}{\eta}\right),$$
$$\epsilon_3 = \big(1+\epsilon_2\big)\frac{L^{2T_3}}{W_I(T_3)!}.$$

Therefore, we can compute an approximation to

$$\Pr[(v_j X_I + aX')^2 + Z > t]$$

with additive error $\epsilon_1 + \epsilon_2 + \epsilon_3$ by evaluating a polynomial with $O(T_1^3 T_2^3 T_3)$ terms. □

C Proof of Lemma 4

Definition 1. *$\phi(x)$ will denote the normal density function, i.e., $\phi(x) = \frac{1}{\sqrt{2\pi}} e^{-x^2/2}$.*

Definition 2. *$N(c)$ will denote the tail of the normal cumulative distribution function, i.e., $N(c) = \int_c^\infty \phi(x)\,dx$*

We begin by describing X', which we try to make η-close to the 1-dimensional normal distribution (we specify η in terms of δ later). By Proposition 5, we can pick an $M = O(\sqrt{\log(1/\eta)})$ such that $2N(M) \leq \eta$; i.e., such that the probability that a normal variable's magnitude exceeds M is at most η. Define $P \subseteq \Re$ to be the set of integer multiples of 2θ which are no more than M in magnitude (we will also specify θ in terms of δ later). Let X' be the random variable taking values in P given by the following randomized algorithm:

- Generate a random real t between 0 and 1 by using $2\log(1/\theta)$ random bits
- Proceed through the points $p_1, p_2, \ldots$ in P
  - At point p_i, calculate an approximation f_i to $\phi(p_i)$ within $\pm\theta$
  (simply use standard numerical techniques)
  - Subtract $2\theta f_i$ from t
  - If t becomes negative, stop and output p

Notice that this algorithm can be carried out in $O(\log(1/\theta))$ space and $O(|P|\log(1/\theta))$ time.

Lemma 5. *Under this algorithm, the probability that p_i is chosen is equal to $\int_{p_i-\theta}^{p_i+\theta} \phi(x)\,dx + O(\theta^2)$.*

Proof. p_i is selected iff the initial choice of t is in the range $[\sum_{j=1}^{i-1} 2\theta f_j, \sum_{j=1}^{i} 2\theta f_j)$, an interval of width $2\theta f_i$. Since t is uniform except with error $O(2^{-2\log(1/\theta)}) = O(\theta^2)$, we conclude that p_i is selected with probability $2\theta f_i + O(\theta^2)$.

Since $f_i = \phi(p_i) + O(\theta)$, it follows that $2\theta f_i + O(\theta^2) = 2\theta\phi(p_i) + O(\theta^2)$. Finally, Proposition 7 tells us that $2\theta\phi(p_i)$ differs from $\int_{p_i-\theta}^{p_i+\theta} \phi(x)\,dx$ by at most $O(\theta^3)$. □

Lemma 6. *The random variable X' can be written as $X + D + U$, where:*

- *X is a random variable drawn from the true normal distribution*

- *D ("discretization" error) is a random variable* dependent *on X with $|D| \leq \theta$ always*
- *U ("unlikely events" error) is a random variable* dependent *on X which is 0 with probability at most $1 - \eta - O(M\theta)$*

Proof. Consider the following hypothetical way of picking an x' according to the distribution X'. First, draw X from the true normal distribution, yielding x. Then $|x| > M$ with probability at most η. This accounts for the η chance that U is nonzero, and we now assume that $|x| \leq M$. Next, round x to the nearest multiple of 2θ, yielding x'. This gives us the discretization error D, with $|D| \leq \theta$.

At this point, we have x' selected with probability $\int_{x'-\theta}^{x'+\theta} \phi(x)\,dx$. By the previous lemma we know this is correct within an additive error of $\pm O(\theta^2)$. Over all possible choices of $x' \in P$, this leads to a total additive error of $|P|O(\theta^2) = O(M\theta)$. This can be accounted for using the $O(M\theta)$ chance that U is nonzero. □

Finally, we are in a position to prove Lemma 4.

Proof of Lemma 4. Consider the two algorithms, $\mathcal{A}_{v,c}$ and $\mathcal{B}_{v,w}$, using the distribution X'. They generate $r' \in \Re^k$ by taking k i.i.d. instances of X' for the coordinates of r'. Then r' is distributed as $X + D + U$, where:

- X is a true k-dimensional normal random vector
- D is a random vector dependent on X with $|D| \leq \theta\sqrt{k}$ always
- U is a random vector dependent on X which is 0 with probability at least $1 - k(\eta + O(M\theta))$

The probability that $v \cdot X \geq c + \theta\sqrt{k}$ is $N(c + \theta\sqrt{k})$, since v is a unit vector. By Proposition 6, $N(c) - N(c + \theta\sqrt{k}) < \theta\sqrt{k}$. Since $U = 0$ except with probability $k(\eta + O(M\theta))$, the probability that $v \cdot (X + U) \geq c + \theta\sqrt{k}$ is at least $N(c) - \theta\sqrt{k} - k(\eta + O(M\theta))$. Finally, if $v \cdot (X + U) \geq c + \theta\sqrt{k}$ then $v \cdot (X + D + U) \geq c$, because $v \cdot D \geq -\theta\sqrt{k}$ since v has unit length and $|D| \leq \theta\sqrt{k}$. Hence we conclude that the probability that $v \cdot r' \geq c$ is at least $N(c) - \theta\sqrt{k} - k(\eta + O(M\theta))$.

A similar argument shows $\Pr[v \cdot r' \geq c] \leq N(c) + \theta\sqrt{k} + k(\eta + O(M\theta))$, and so $\mathcal{A}_{v,c}$ has the correct output probability within $\pm[k(\eta + O(M\theta)) + \theta\sqrt{k}]$.

The calculation for $\mathcal{B}_{v,w}$ is similar. We handle U as before, and for D, we note that it has no effect on the sign of $v \cdot (X + D)$, so long as $|v \cdot X| > \theta\sqrt{k}$. Since the distribution of $v \cdot X$ is 1-dimensional normal, this fails only with probability $N(-\theta\sqrt{k}) - N(\theta\sqrt{k}) \leq 2\theta\sqrt{k}$. Hence $\mathcal{B}_{v,w}$ has the correct output probability within $\pm[k(\eta + O(M\theta)) + 2\theta\sqrt{k}]$.

In conclusion, we have that both algorithms have the correct output distribution up to an additive error of $[k(\eta + O(M\theta)) + O(\theta\sqrt{k})]$. Thus if we take $\eta = \delta/k$, and $\theta = \delta/(k\sqrt{\log(k/\delta)})$, this quantity is $N(c) \pm O(\delta)$.

Plugging the values of η and θ back into randomness, space, and time bounds, we indeed get $O(\log(k\sqrt{\log(k/\delta)}/\delta) = O(\log(k/\delta))$ randomness and space, and $O(\frac{1}{\delta}\log^2(k/\delta))$ time. □

D A deterministic $O(\Delta^{1/3+O(\epsilon)})$-semicoloring

In this section with give a rounding procedure that can be applied to the [KMS98] algorithm to produce an $O(\Delta^{1/3+O(\epsilon)})$-semicoloring. We also show how to derandomize this rounding procedure. Before starting the rounding procedure we apply dimensionality reduction in the same way as for the simpler rounding procedure described in the main body of this paper. This implies that for any edge (i_1, i_2) in the graph, the corresponding unit vectors w_{i_1} and w_{i_2} satisfy $w_{i_1} \cdot w_{i_2} \leq -1/2 + \epsilon$.

D.1 The rounding procedure

Suppose t is a large constant parameter. We will have a number of *rounds*, in each of which we partition the vertices into $t+1$ classes. We do this using t independent selection procedures: We pick t random vectors such that the coordinates in the vectors are i.i.d. $N(0,1)$ and for each such vector r_j we form the class of vertices $C_j = \{i\colon w_i \cdot r_j \geq c\}$, where c is a global constant that will be chosen later. If a vertex falls into more than one class, we pick one for it arbitrarily. If a vertex is not put into any class, we put it in the "leftover", $(t+1)$st class.

Suppose the t random vectors were drawn from the true normal distribution. The probability that both vertices in an edge (i_1, i_2) fall entirely into the class C_j is

$$\Pr[\{w_{i_1} \in C_j\} \cap \{w_{i_2} \in C_j\}] = \Pr[\{w_{i_1} \cdot r_j \geq c\} \cap \{w_{i_2} \cdot r_j \geq c\}] \leq \Pr[(w_{i_1} + w_{i_2}) \cdot r_j \geq 2c].$$

The latter probability is $N(2c/\|w_{i_1} + w_{i_2}\|^2)$, thus

$$\Pr[\{w_{i_1} \in C_j\} \cap \{w_{i_2} \in C_j\}] \leq N\big(2c/(2 + 2w_{i_1} \cdot w_{i_2})\big) \leq N\big(2c/(1+2\epsilon)\big).$$

The probability that both vertices fall in the leftover class can be bounded by

$$\Pr[\{w_{i_1} \in C_{t+1}\} \cap \{w_{i_2} \in C_{t+1}\}] \leq \Pr[\{w_{i_1} \in C_{t+1}\}] = \big(1 - N(c)\big)^t,$$

where the last bound follows since a vertex is put in the leftover class if does not qualify for any of the first t classes. Hence

$$\Pr[i_1 \text{ and } i_2 \text{ fall into same class}] \leq tN\big(2c/(1+2\epsilon)\big) + \big(1 - N\big(c\big)\big)^t,$$

and we call this latter probability p. We now argue that c and t can be selected such that p is small.

By Proposition 8, $N(2c/(1+2\epsilon)) = N(c)e^{-(3-O(\epsilon))c^2/2}$. Thus, we can set

$$t = \frac{3}{N(c)} \ln \frac{1}{N(c)} = 3ce^{c^2/2}\left(\ln \frac{1}{N(c)}\right)\big(1 + o(1)\big)$$

to get $p = O(e^{-(3-O(\epsilon))c^2/2} \log^4(1/N(c)))$.

It follows that if we make c large enough, and partition into $t+1$ classes according to $t = 3(1/N(c)) \ln(1/N(c))$ independent random vectors, the number of edges which remain in the same class goes from m down to at most pm in expectation. If we now do $\log_p(1/2\Delta)$ rounds, the number of uncut edges decreases to at most $m/2\Delta \leq n/4$, and we have a semicoloring as desired. Since we split into t classes in each round, the total number of color classes in the end is $t^{\log_p(1/2\Delta)} = (1/2\Delta)^{\log_p t} \leq (1/2\Delta)^{1/3+O(\epsilon)}$.

Now, recall that we don't have true normal distributions. Hence the probabilities $N(c)$ and $N((2-4\epsilon)c)$ in the above discussion should really be replaced with $N(c)-2\delta$ and $N(2c/(1+2\epsilon))+2\delta$. Hence for the same proof to go through, we need δ to be much smaller than both $N(c)$ and $1/t$.

All of this can be achieved if we pick c and δ to be constants in the correct order. Specifically, take c to be a sufficiently large constant so that the approximation $N((2-4\epsilon)c) \approx N(c)^{4-O(\epsilon)}$ is correct within an additive constant error dependent on ϵ. This forces t to be an even larger constant. Then we can take δ to be an extremely small constant compared to $N(c)$ and $1/t$, and we're done.

D.2 The derandomization

It remains to give the derandomization and calculate its running time. As the constants c, t and δ all depend implicitly on ϵ, our $O(\cdot)$ notation will hide a (very big) dependence on ϵ. In every round we select $t = O(1)$ vectors independently using the pseudorandom generator, with error parameter $\delta = \Omega(1)$. This requires $O(t \log^{O(1)}(k/\delta)) = \log^{O(1)} k$ truly random bits. We can enumerate over all of these random bits in time $2^{\log^{O(1)} k}$. Since $k = O(\log n)$, this is sublinear in n. Once enumerated, we can pick a set of t vectors which does better than the expected value; i.e., leaves no more than a p fraction of the remaining uncut edges uncut. Repeating this for all $O(\log n)$ rounds gives us the desired semicoloring in sublinear time.

E Some useful propositions

Proposition 1. *Let $f(x) = e^{-x}$. In the interval $x \in [0, k]$, $f(x)$ can be approximated within additive error $k^T/T!$ by a polynomial with T terms and degree $T - 1$.*

Proof. We approximate f by a truncated MacLaurin series,

$$f(x) = \sum_{n=0}^{T-1} \frac{(-1)^n x^n}{n!} + R_T$$

Since the series is alternating and the absolute values of the terms form a strictly decreasing sequence, $|R_T| < k^T/T!$. □

Proposition 2. *Let $f(x) = e^{-x^2/2}$. In the interval $x \in [-k, k]$, $f(x)$ can be approximated within additive error $k^{2T}/2^T T!$ by a polynomial with T terms and degree $2T - 2$.*

Proposition 3. *For any non-negative integer m, $\int x^m \, dx = \frac{1}{m+1} x^{m+1}$.*

Proposition 4. *Consider $\int x^m \sqrt{a - x^2} \, dx$ in the interval $[-\sqrt{a}, \sqrt{a}]$ for any non-negative real a and any non-negative integer m. When m is odd, the integral can be written as $\sqrt{a - x^2}$ times a polynomial in x with $O(m)$ terms. When m is even, it can be written as the sum of $\frac{a}{2} \arcsin \frac{x}{\sqrt{a}}$ and $\sqrt{a - x^2}$ times a polynomial in x with $O(m)$ terms.*

Proof. Let $I_m = \int x^m \sqrt{a - x^2} \, dx$ and apply some of the results in [GR94, §2.26] with $b = 0$, $c = -1$, $\Delta = 4ac - b^2 = -4a$, and $n = 0$. For $m \geq 2$ we get [GR94, equation (2.260.1), p 98]

$$\begin{aligned} I_m &= \int x^m \sqrt{a - x^2} \, dx \\ &= -\frac{x^{m-1}(a - x^2)^{3/2}}{m + 2} + \frac{(m-1)a}{m+2} \int x^{m-2} \sqrt{a - x^2} \, dx \\ &= -\frac{x^{m-1}(a - x^2)^{3/2}}{m + 2} + \frac{(m-1)a}{m+2} I_{m-2}. \end{aligned}$$

The base cases are

$$I_1 = \int x \sqrt{a - x^2} \, dx = -\frac{(a - x^2)^{3/2}}{3}$$

for odd m. For even m, we get the base case

$$I_0 = \int \sqrt{a - x^2}\, dx = \frac{x\sqrt{a - x^2}}{2} + \frac{a}{2} \int \frac{dx}{\sqrt{a - x^2}}$$

by [GR94, equation (2.262.1), p 99]. The last integral above is easily evaluated; it is

$$\int \frac{dx}{\sqrt{a - x^2}} = \arcsin \frac{x}{\sqrt{a}}.$$

by [GR94, equation (2.261), p 99]. Thus,

$$I_0 = \frac{x\sqrt{a - x^2}}{2} + \frac{a}{2} \arcsin \frac{x}{\sqrt{a}}.$$

□

Proposition 5. *Let $\phi(x) = \frac{1}{\sqrt{2\pi}} e^{-x^2/2}$ be the density function for the $N(0,1)$ distribution. Then $\phi(x)(\frac{1}{x} - \frac{1}{x^3}) < \int_x^\infty \phi(t)\, dt < \frac{1}{x}\phi(x)$.*

Proof. See e.g. [Fel62]. □

Proposition 6. *Let $\phi(x) = \frac{1}{\sqrt{2\pi}} e^{-x^2/2}$ be the density function for the $N(0,1)$ distribution. Then, for any x and any non-negative δ, $\int_x^{x+\delta} \phi(t)\, dt < \delta/\sqrt{2\pi}$.*

Proof. By the mean value theorem, there exists a $\xi \in [x, x + \delta]$ such that $\int_x^{x+\delta} \phi(t)\, dt = \phi(\xi)\delta$. Since $\phi(x) \leq \phi(0) = 1/\sqrt{2\pi}$, the result follows. □

Proposition 7. *Let $\phi(x) = \frac{1}{\sqrt{2\pi}} e^{-x^2/2}$ be the density function for the $N(0,1)$ distribution. Then, for any x and any non-negative δ, $\int_{x-\delta}^{x+\delta} \phi(t)\, dt = 2\delta\phi(x) + O(\delta^3)$.*

Proof. Let $N(x) = \int_x^\infty \phi(t)\, dt$. Then $\int_{x-\delta}^{x+\delta} \phi(t)\, dt = N(x - \delta) - N(x + \delta)$. By expanding the latter expression in a Taylor series around x and using the fact that $N' = -\phi$, we obtain

$$\int_{x-\delta}^{x+\delta} \phi(t)\, dt = -2N'(x)\delta - \frac{N'''(\xi_1)}{6}\delta^3 - \frac{N'''(\xi_2)}{6}\delta^3 = 2\delta\phi(x) + \frac{\phi''(\xi_1)}{6}\delta^3 + \frac{\phi''(\xi_2)}{6}\delta^3$$

for some $\xi_1 \in [x - \delta, x]$ and some $\xi_2 \in [x, x + \delta]$. Since ϕ'' is bounded, the result follows. □

Proposition 8. *Let $\phi(x) = \frac{1}{\sqrt{2\pi}} e^{-x^2/2}$ be the density function for the $N(0,1)$ distribution and $N(x) = \int_x^\infty \phi(t)\, dt$ Then*

$$\frac{\int_x^\infty \phi(t)\, dt}{\int_{ax}^\infty \phi(t)\, dt} \geq \sqrt{2}\left(1 - \frac{1}{x^2}\right) e^{(a^2-1)x^2/2}.$$