1

Randomization and derandomization

Department of Computer Science and Engineering and
Centre for Theoretical Studies
Indian Institute of Technology, Kharagpur 721302, India.

January 14, 2014

Keywords: expectation, conditional probability, Las Vegas, Monte Carlo, first moment, discrepancy, hypergraph, random sampling, ϵ-nets and ϵ-approximations, probabilistic method, expanding graph, derandomization.

Contents

1 The method of expectations or the first moment for hypergraph bicolouring

We start with a problem dealing with set systems $G(V, E)$, where we have a set V of n *elements* (or *vertices*), and the set E containing subsets $h \subseteq V$ of these elements where $|h| \geq r$. Such subsets $h \in E$ are called *hyperedges* and such a system $G(V, E)$ is called a *hypergraph*. [A hypergraph where each $h \in E$ satisfies $|h| = 2$, is an undirected *graph*.] Suppose we consider *sparse* hypergraphs where $|E| < 2^{r-1}$, where each hyperedge has at least r vertices. Here, it makes sense to assume that r is not a constant. If r is a constant then the number of hyperedges remains fixed even as the number n of vertices grows. The sparsity of the hypergraph is due to the upper bound of 2^{r-1} on the number of hyperedges in the hypergraph. Suppose we colour the vertices with two colours, say 0 and 1. How do we determine whether such a hypergraph has a *proper bicolouring*? [A *proper bicolouring* is such that no hyperedge is *monochromatic*, that is, has all vertices painted with the same colour.] An arbitrary hypergraph may or may not be properly bicolourable. However, we show that proper bicoluring is possible under the assumptions of sparsity as above.

The proof goes as follows and uses some analysis of *random bicolourings* of the set of vertices V. One way is to show that the probability of a proper bicoluring in the random process is non-zero. This method is termed the *probabilistic method*. The probability that a hyperedge is monochromatic is no more than 2×2^{-r} because all the vertices in the hyperedge could be either white or black. [There are at least r vertices in each hyperedge.] So, the probability that some hyperedge is monochromatic is no more than $|E|2^{-(r-1)} < 2^{r-1}2^{-(r-1)} = 1$. The probability that no hyperedge is monochromatic is therefore non-zero. So, there must be a proper bicolouring. See [4].

Another way is to show that the expected number of *monochromatic* hyperedges in the random process is less than one; there must be at least one bicolouring with no (or zero) monochromatic hyperedges. This is called the *first moment method*. Here, we consider the expected number of monochromatic hyperedges in a random bicolouring of the n vertices. Let I_i be 1 (0), depending on whether the ith hyperedge is monochromatic; such variables are called *indicator* variables. We consider the expectation $E(\Sigma_{i=1}^{|E|} I_i)$, which is the expectation of a sum of random indicator variables. It is known that this expectation is no more than the sum $\Sigma_{i=1}^{|E|} 2^{-(r-1)} < 2^{r-1}2^{-(r-1)}(= 1)$, of expectations of the indicator variables. [The expectation of each indicator variable is easily seen to be $2^{-(r-1)}$.] Since the expected number of monochromatic hyperedges is strictly less than 1, there must be a random colouring with an integral number of zero monochromatic hyperedges. See [4].

2 A simple Las Vegas method for proper bicolouring of hypergraphs

In a random bicolouring as in Section 1, a hyperedge is rendered monochromatic with probability $\frac{2}{2^r} = \frac{1}{2^{r-1}}$. Provided we assume that $|E| < 2^{r-2}$, we can show that the probability that some hyperedge is monochromatic is upper bounded by $|E|2^{r-1} < \frac{1}{2}$. So, the probability that no hyperedge is monochromatic exceeds $\frac{1}{2}$. Therefore, simply a random bicolouring of the vertices leads to a success rate of over 50%. Once we get a random bicolouring,

it is a straightforward task to verify by scanning all hyperedges whether the bicoluring is proper by checking each hyperedge for non-monochromaticity. So, a failure to get a proper random bicolouring can be followed by another attempt at proper random bicolouring until we finally get a proper bicoluring to settle with. In such a sequence of bicolouring attempts, all but the last step is successful. If there are t steps of failure then the probability with which these t failures can occur in a sequence is no more than $(\frac{1}{2})^t$, which diminishes exponentially with t, and the expected number of failures is $\Sigma_t t(\frac{1}{2})^t \leq 2$. See [3, 4].

3 Derandomizing using the method of conditional probabilities for properly bicolouring hypergraphs

In Section 2, we have extended the result of Section 1 of the existence of a proper bicolouring by actually designing a Las Vegas algorithm for actually generating the proper bicolouring in polynomial time. Now we will show how to design a deterministic polynomial time algorithm for proper bicolouring using the *method of conditional probabilities*. The number of hyperedges is strictly less than 2^{r-2} and the number of vertices in any hyperedge is at least r, as assumed in Sections 2 and 1. See the details in [3].

4 A simple Monte Carlo method for computing the minimum cut

We know that the minimum cut in an undirected graph $G(V, E)$ can be found out by $|V|$ applications of *maximum flow computation* from a vertex s to all vertices $t \in V \setminus \{s\}$. We may use either the Ford-Fulkerson algorithm [2] or the algorithms of Dinic, Edmonds-Karp or Malhotra et al. Here, we present a Monte Carlo algorithm that computes the minimum cut with high probability. The main idea is to *collapse* edges one by one; the two vertices of the collapsed edge are made into a single *supervertex*. The edges incident on the original (super)vertices are preserved. The process goes on until only two supervertices are left. The set of edges across these vertices is declared as the desired cut. Throughout, we maintain and process the *multigraph*, which can have multiple edges between a pair of supervertices. The methods for these operations and the data strcutures involved are not straightforward and surely afr from being trivial. These operational details may be worked out for the sake of completeness.

The question is whether the cut so computed is also a minimum cut. We can see that the cut obtained is certainly a cut of the original input graph. Why? We also observe that as long as we have collapsed only those edges that do not belong to a minimum cut C, we are done. How would a randomized algorithm avoid the collapse of each and every edge of the minimum cut C? Every time we run the same randomized algorithm on the very same initial graph as input, we may very well collapse different sets of edges. We now determine the (finite) probability with which we would certainly be able to avoid the collapse of the edges of the minimum cut C. Here, C is just any one of the possible minimum cuts. A (random) run of the algorithm (when successful), could deliver any one of the possibly many minimum cuts in teh graph.

We use the observation that the *degree* of a vertex cannot be smaller than the cardinality k of the minimum cut. Why? We also know that the number $e = |E|$ of edges cannot be less than $n\frac{k}{2}$. Why? The probability of selecting an edge of the cut C is $\frac{k}{|E|} \leq \frac{2}{n}$, which is rather small. Starting with this argument and continuing as we merge vertices, show that the minimum cut C is computed with high probability in polynomial time. See [7, 6] for detailed analysis as follows. The probability of choosing an edge that is not in the mincut C is at least $1 - \frac{2}{n}$. The graph now has only $n - 1$ vertices. So, the probability of again choosing an edge not in mincut C is

at least $1 - \frac{2}{n-1}$. The probability of selecting no edge of C in $n-2$ edge collapse steps is at least

$$(1 - \frac{2}{n})(1 - \frac{2}{n-1})...(1 - \frac{2}{n-i+1})..(1 - \frac{2}{3})$$

This is at least $\frac{2}{n^2}$ The probability of failure is therefore at most $1 - \frac{2}{n^2}$. The probability that $\frac{n^2}{2}$ such full algorithmic steps would fail to discover the mincut C is at most

$$(1 - \frac{2}{n^2})^{\frac{n^2}{2}}$$

This is at most $\frac{1}{e}$.

Note that the analysis hinged on the number of vertices remaining in each stage; this number reduced by unity for each of the $n-2$ stages of edge collapsing. Further, the analysis also requires the degree of a vertex to remain intact, even with multiedges. That is, if a supervertex has multiple edges incident on it then all these are counted in its degree so that the probability estimates remain valid in the analysis. These degrees therefore need to be maintained in counters in the data structures as the edge collapsing stages proceed. The total number of multiedges too does not remain bounded by $\binom{n}{2}$. Another issue is that of picking one of these large number of multiedges for collapsing, uniformly at random. Multiple edges may join the same pair of supervertices but they must be accounted as separate edges.

Further, some questions remain to be addressed. Can we further reduce the failure probability. How? To what extent? And at what cost in running time? Can any of these ideas be used in any attempts to design algorithms for similar problems, partciularly, to problems about cuts?

5 The method of random sampling for a set system or hypergraph

The following discussion is based on [1]. Take any *set system* (hypergraph) $G(V, S)$ where $V = \{v_1, \cdots, v_n\}$ and $S = \{e_1, \cdots, e_m\}$; here, $e_i \subseteq V$ for all $1 \leq i \leq m$. Given any integer $1 \leq r \leq n$, we wish to find a subset $N \subseteq V$ that intersects every e_i of size greater than $\frac{n}{r}$. We can assume that $|e_i| > \frac{n}{r}$, for any i, and $m > 1$.

5.1 Binomial random sampling of the set of vertices

Let $p = \frac{cr(\log m)}{n}$, for some large enough constant c. Sample the set V by Binomial distribution, that is, construct the set N by including in it v_i with probability p. Then, $N \cap e_i = \phi$ with probability less than $(1-p)^{\frac{n}{r}}$ for a single value of i. The probability that N does not intersect some e_i is less than $m(1-p)^{\frac{n}{r}}$. We can make this probability smaller than any constant. Why? [The probability can be shown to be bounded by $\frac{1}{m^{c-1}}$, which can be made smaller than any given limiting constant as m grows beyond a certain value for each valid choice of the constant c.] So, the sample N of expected size $np = cr \log m$ intersects every set e_i of size $\frac{n}{r}$; note also that the random sample N is of size $O(r \log m)$ with high probability. Therefore, we have a way of getting random samples such as N of size $O(r \log m)$ with high probability, so that N intersects all the sets e_i of size greater than $\frac{n}{r}$.

This was a simple randomized construction of such a set N (with high probability), that intersects all the sets of size not lesser than $\frac{n}{r}$. Can we construct such a set N by a deterministic method? Below, we discuss a greedy deterministic method, and later we consider another method, which is a derandomization of the above randomized sampling technique.

5.2 ϵ-nets and ϵ-approximations

Before we take up the deterministic methods for generating such sample sets as N, we introduce some standard terminology and definitions. For any set system or hypergraph $G(V, S)$, a set $N \subseteq V$ is called an ϵ-net for $G(V, S)$ if $N \cap e_i \neq \phi$, for any $e_i \subseteq S$ with $|e_i| \geq \epsilon|V|$. The set N in Section 5.1 is an $\frac{1}{r}$-net for $G(V, S)$ with high probability. We now consider ϵ-approximations as defined in [1], pages 169-170. As a simple exercise, show that an ϵ-approximation is also an ϵ-net but the converse is not necessarily true. As a reading exercise, study the notions of such *nets* and *approximations* and their significance. See Section 4.1 of [1].

Suppose e_i is not known to us but we have an ϵ-approximation A. Given the set A, suppose we can test and identify all elements of e_i in A quickly. We do not know e_i but we can certainly find $e_i \cap A$ in time proportional to the size of A. Can we then estimate the cardinality $|e_i|$ of e_i? Do we require the assumption that $|e_i| > \epsilon|V|$? In this way, what advantage do we have with respect to an ϵ-net? What can we do with an ϵ-net N? We can test if N has any element in e_i by looking at each element of N, thereby being able to get such an element in N if the cardinality of e_i is above a threshold. What happens if we do not find any vertex of e_i in N?

As an exercise, work out the proofs of Lemma 4.1 and 4.2 on page 171 of [1].

5.3 Deterministic construction of ϵ-nets

We show that one greedy and deterministic way of generating an ϵ-net N is as follows. Find a vertex v_i contained in most sets $e_j \subseteq S$. Remove this vertex from further consideration and add it to N. Then, remove all sets containing v_i from future consideration. Repeat these steps until all hyperedges from S are removed. Show that this algorithm can be made to run in $O(mn)$ time. Does N turn out to be quite small as required? Let m_k be the number of hyperedges remaining after k iterations. Clearly, $m_0 = m$. Sticking to the assumption that each set e_i has at least $\frac{n}{r}$ elements, we now have m_k sets left after the kth iteration with at least $\frac{n}{r}$ elements. We can select any of the $n - k$ remaining vertices in the next iteration. We have a distribution of $n - k$ distinct vertices in at least $m_k \times \frac{n}{r}$ instances over the m_k sets. So, the most frequent vertex of the m_k sets must be in at least $\frac{m_k \times \frac{n}{r}}{n-k} \geq \frac{m_k}{r}$ sets. Why? [What is the expected number of vertices in these sets?] Thus, $m_{k+1} \leq m_k(1 - \frac{1}{r})$. So, $m_k \leq m(1 - \frac{1}{r})^k$. For a large enough constant $c > 0$, and any $k \geq cr \log m$, we have $m_k < 1$, and therefore $m_k = 0$. In other words, picking any sufficiently large number $k \geq cr \log m$ of vertices we can ensure that we hit all the hyperedges that have at least $\frac{n}{r}$ vertices. So, we can deterministically compute a $\frac{1}{r}$-net N of size $O(r \log m)$, greedily as above.

6 Random sampling for geometric/searching applications

The following discussion is based on Chapter 5 (pages 173-175) of [5]. We have n objects in the set N, and subsets of N can be the *defining elements* of *configurations*. Let $\Pi = \Pi(N)$ be the set of all configurations and $\sigma \in \Pi$ be one such configuration. For instance, imagine a one-dimensional space (the real line) with n distinct points and the pairs of n points as configurations, which are actually linear intervals. If we fix a constant $r < n$, we may take a *random sample* R of r elements, selected out of the n elements in N. [Sampling is done without repetitions; each time an element is selected independently and randomly.] Here, the cardinality $d(\sigma)$ of the set $D(\sigma)$ of *triggers* or *defining elements* of any of these configurations σ is exactly 2. Each configuration σ may contain (intersect) a set $L(\sigma)$ of $l(\sigma)$ elements from N. These are called *stoppers* of the configuration σ; the number of stoppers is is also called the *conflict size* of the configuration σ. We wish to derive an upper bound on the probability that we get a configuration with a large number of stoppers but no triggers (except for the two extreme defining triggers of the configuration). Such configurations from Π, are called *active configurations*

over the random sample R. We show that every active configuration of Π over the random sample R ($r = |R|$), would have conflict size $O(\frac{n}{r}\log r)$ with probability at least $\frac{1}{2}$. Let $p(\sigma, r)$ be the conditional probability that R has no point in conflict with σ, given that R contains the defining elements of σ. We claim that $p(\sigma, r)$ follows

$$p(\sigma, r) \leq (1 - \frac{l(\sigma)}{n})^{r-d(\sigma)} \tag{1}$$

The intuitive justification is as follows. The interval being of conflict size $l(\sigma)$, the probability of choosing a conflicting point is at least $\frac{l(\sigma)}{n}$. Since we select $r - d(\sigma)$ points without conflicts, the probability required is upper bounded as in Inequality 1. [For a rigorous derivation, see page 175 in [5].] However, since $1-x \leq exp(-x)$ where $exp(x) = e^x$, we have

$$p(\sigma, r) \leq exp(-\frac{l(\sigma)}{n}(r - d(\sigma))) \tag{2}$$

Since $d(\sigma) \leq 2$, putting $l(\sigma) \geq c(n \ln s)/(r-2)$ for some $c > 1$ and $s \geq r$, we get

$$p(\sigma, r) \leq exp(-c \ln s) = \frac{1}{s^c} \tag{3}$$

Now an active configuration σ over the random sample R must be such that all its defining points must be in R. In other words, $\sigma \in \Pi(R)$. Let this probability be $q(\sigma, r)$. The probability that σ is an active configuration due to random sample R with conflict set at least as big as $\frac{cn \ln s}{r-2}$ is therefore no more than

$$p(\sigma, r)q(\sigma, r)$$

The probability that some active configuration has such a "long" conflict set is no more than the sum of probabilities for all such "long" configurations $\sigma \in \Pi(R)$

$$\sum_{\sigma \in \Pi : l(\sigma) > \frac{cn \ln s}{r-2}} p(\sigma, r)q(\sigma, r) \leq \sum_{\sigma \in \Pi : l(\sigma) > \frac{cn \ln s}{r-2}} q(\sigma, r)/s^c \leq \frac{1}{s^c} \sum_{\sigma \in \Pi} q(\sigma, r) \tag{4}$$

Now the last summation in the Inequality 4 is the expectation $E(\pi(R))$ and $\pi(R) = |\Pi(R)| = O(r^2)$. So, choosing $c > 2$ we can ensure that the probability of having a long active configuration in $\sigma \in \Pi(R)$ is less than $\frac{1}{2}$ for a random sample R.

For a reading exercise, study Section 5.1, pages 176-180, from [5] where generalizations to two and higher dimensions are considered. The configurations spaces considered have *bounded valence*. It is shown that a result similar to that derived above holds for such spaces (see Theorem 5.1.2 [5]).

This derivation is also worked out in [6].

7 Examples illustrating the probabilistic method

See Example 5.3 on page 102 of [6]. This example illustrates the use of the probabilistic argument about the existence of a *solution* to a system by virtue of the fact that a random 0-1 vector satisfies the system with a finite non-zero probability. As a reading exercise, study this example from pages 102-103. Also study the proof of Theorem 5.1 from page 103, in [6].

7.1 The existence of expanding graphs

We prove the existence of an *expander* or *expanding bipartite graph* as in [6]. Here, the graph is a *random graph* with $2n$ vertices and at most dn edges where d is an integer constant. The edges run from vertices in the set L of n vertices to the set R of n vertices, where each vertex in L is connected to at most d vertices in R. The notation for such a bipartite graph is $G(L, R, E)$, where E is the edge set. The random graph is constructed uniformly and randomly selecting d vertices from the set R for each vertex of L, with repetitions allowed. This may give several edges between a pair of vertices but we count only one copy for such repetitions. So, we may have vertices with degrees in the range $[1, d]$. Now consider a set $S \subset L$ where $s = |S|$. These s vertices can connect to at most ds vertices in R. However, due to repeated edges and due to common neighbours of vertices of S in R, the actual neighbourhood set for S in R may have much less than ds vertices. We wish to have the property that any subset S of L, of cardinality s, $1 \leq s \leq \alpha n$, must have a neighbourhood set in R of cardinality more than cs, where $c > 1$ is a constant. This is the reason why we call such graphs *expanders*. Naturally, we expect that $c < d$. Now we show the existence of an expanding graph with $c = 2$, $d = 18$ and $\alpha = \frac{1}{3}$.

The main idea is to over estimate the failure probability $\mathcal{E}(s)$, of a set S in achieving a neighbourhood set of cardinality more than cs in the set R. So, we must ensure (for such failures) that all the ds random neighbours of vertices in S, are concentrated within only some cs vertices of R. This failure probability can be bounded by

$$\binom{n}{s}\binom{n}{cs}(\frac{cs}{n})^{ds}$$

Then, we can sum up such failure probabilities for all s in the range $1 \leq s \leq \alpha n$ to get the overall failure probability. Once we show that the cummulative failure probability is strictly less than unity, we can say that the success probablity is non-zero, which means that there is indeed an expander (a random graph), with the properties we are looking for. See the detailed analysis in class handouts and in Section 5.3 of [6].

7.2 Yet another expanding graph

We can keep L as it is with n vertices and make R bigger with $2^{\log n^2}$ vertices. The first property we require is an *expander* property that each subset of $\frac{n}{2}$ vertices of L must have at least $2^{\log n^2} - n$ neighbours in R. An additional second property required is that no vertex in R must have more than $12\log n^2$ neighbours in L. The random graph is again similarly constructed as in Section 7.1, with the difference that $d = 2^{\log n^2}(4\log^2 n)/n$. The failure probability in the case of this expanding graph with respect to the first (expander) property is bounded by

$$\binom{n}{\frac{n}{2}}\binom{2^{\log^2 n}}{n}(\frac{2^{\log^2 n} - n}{2^{\log^2 n}})^{\frac{dn}{2}}$$

This failure probability is strictly less than 50%. See Theorem 5.7 of [6] for this claim about the first property as well as the second property.

8 Discrepancy upper bounds and derandomization using the method of conditional expectations

Given a set system (hypergraph) $G(V, S)$, with n vertices and m hyperedges, we define

$$\chi(e_i) = \sum_{v \in e_i} \chi(v)$$

where $\chi(v) \in \{1, -1\}$ is the colour assigned to vertex v. The maximum of $|\chi(e_i)|$ over all $e_i \in S$, is called the *discrepancy* of the system under the given colouring. The minimum discrepancy over all possible colourings $D(S)$ is called the *discrepancy* of the system.

8.1 An upper bound for combinatorial discrepancy using tail bounds

If we perform a random colouring χ for the vertices, uniformly and randomly from $\{1, -1\}$, then we say that e_i is *bad* if $|\chi(e_i)| > \sqrt{2|e_i| \ln(2m)}$. Using Chernoff's bound, we can show that Prob [e_i is bad]$< \frac{1}{m}$ as follows.

[We use the bound as in Lemma A.5 of Appendix A in [1]. The sum X of n mutually independent random variables x_i, uniformly distributed in $\{+1, -1\}$ is such that Prob $[X \geq \Delta] < e^{-\Delta^2/2n}$, for any $\Delta > 0$. Substituting the value for Δ as in the required upper bound for discrepancy, we get a probability upper bound of $\frac{1}{2m}$. Since the random variable X, which in our case is $\chi(e_i)$ can be be positive or negative, we take twice this probability upper bound and get the limiting probability $\frac{1}{m}$.]

Since there are m hyperedges, we deduce using the *union bound* that some e_i is bad with probability strictly less than $m \times \frac{1}{m} = 1$. So, no e_i is bad with nonzero probability. Therefore, we conclude that there is a colouring such that $|\chi(e_i)| \leq \sqrt{2|e_i| \ln(2m)}$, for all $e_i \in S$. This means that the discrepancy of the set system is no more than $\sqrt{2n \ln(2m)}$. This result is an example of the use of tail bounds and the probabilistic method for establishing an upper bound on a combinatorial property of a set system (hypergraph).

8.2 A relaxed upper bound and a Las Vegas algorithm for generating a bicolouring with bounded discrepancy

The result in Section 8.1 simply establishes the existence of a colouring with bounded discrepancy. However, it would be nice to compute a colouring with some provable upper bound in polynomial time. We now design a Las Vegas algorithm that runs in polynomial time and generates a colouring with bounded discrepancy. We however relax the discrepancy upper bound achieveable as follows, by redefining *bad* colouring. We now say that e_i is *bad* if $|\chi(e_i)| > \sqrt{3|e_i| \ln(2m)}$. Using Chernoff's bound again, we can easily show that Prob [some e_i is bad]$< \frac{1}{\sqrt{m}}$. Why? [For a specific hyperedge e_i, find Prob[e_i is bad] and multiply by m to get an upper bound for Prob[some e_i is bad].] Show that this observation leads to a Las Vegas algorithm which runs in polynomial time. [Show that the desired bounded discrepancy colouring can be obtained with probability at least $1 - \frac{1}{m^{k/2}}$ if k independent rounds of random colouring are performed.]

8.3 Derandomizing using the method of conditional expectations for generating a bicolouring with bounded discrepancy

Let B_i be the indicator random variable that the ith hyperedge is bad, as defined above in Section 8.2. We define random variable $B = \sum_{i=1}^{m} B_i$ which gives the number of bad hyperedges in a bicolouring. We observe that

$$EB = \sum_{i=1}^{m} EB_i < \frac{1}{\sqrt{m}}$$

when we use the second definition of *bad* hyperedges as in Section 8.2.

This bound helps us keep all expectations within limited precision, thereby enabling computation of expectations whose values can be stored and processed in a reasonable model of computation. Binomial coefficients

of probabilities of various combinations of bicolouring vertices too need to be computed within reasonable precision, amounting to a $O(\log(m+n))$ bits, as we will observe below. We colour vertices $v_1, v_2, \cdots, v_n$ one by one, deterministically with colours $c_1, c_2, \cdots, c_n$, where c_i is either 1 or -1, so that

$$E[B|\chi(v_1) = c1] \leq E[B|\chi(v_1) = -c1] \tag{5}$$

Observe that

$$EB = E_{\chi(v_1)}(E[B|\chi(v_1)]) \geq E[B|\chi(v_1) = c1] \tag{6}$$

Similarly, optimizing over $x \in \{1, -1\}$ for

$$E[B|\chi(v_1) = c_1, \chi(v_2) = c_2, \cdots, \chi(v_{k-1}) = c_{k-1}, \chi(v_k) = x] \tag{7}$$

we can find c_k.

This would enforce

$$E[B|\chi(v_1) = c_1, \chi(v_2) = c_2, \cdots, \chi(v_{k-1}) = c_{k-1}] \tag{8}$$
$$= E_{\chi(v_k)}[B|\chi(v_1) = c_1, \chi(v_2) = c_2, \cdots, \chi(v_{k-1}) = c_{k-1}, \chi(v_k)] \tag{9}$$
$$\geq E[B|\chi(v_1) = c_1, \chi(v_2) = c_2, \cdots, \chi(v_{k-1}) = c_{k-1}, \chi(v_k) = c_k] \tag{10}$$

So, we have

$$E[B|\chi(v_1) = c_1, \chi(v_2) = c_2, \cdots, \chi(v_n) = c_n] \leq \frac{1}{\sqrt{m}} \tag{11}$$

There is nothing to choose when we have already coloured all the n vertices and therefore the expectation above has to be an integral. Why? This integer must be zero because it is less than unity. So, we have a deterministic way of determining a colouring with bounded discrepancy, where no hyperedge is *bad* for the colouring.

We argue now that the determination of all the conditional expectations in the above deterministic procedure are possible in polynomial time with requisite precision. See [1] for details.

9 The use of Chebyshev inequality for Monte Carlo $2n + o(n)$ Randomized Selection

The design of a randomized selection algorithm with expected linear running time is presented in several texts. The design and analysis of such an algorithm may be taken up an exercise. Here we wish to use the technique of *random sampling* to demonstrate an $2n + o(n)$ randomized algorithm that does selection correctly with high probability in an unsorted array S of n elements. Our exposition is based on [6]. The algorithm uses optimal sorting twice, on $O(n^{\frac{3}{4}})$ sized unsorted arrays using $O(n^{\frac{3}{4}} \log n) = o(n)$ time. It also scans the input unsorted array twice. The novel use of Chebyshev inequality is an important feature of the algorithm's correctness arguments.

Sorting the whole array S requires $O(n \log n)$ time, and must therefore be avoided. Instead, we take a sample R of $n^{\frac{3}{4}}$ draws, done uniformly and randomly from elements in S. The multiset R is allowed to have repetitions thus. Certainly, we can sort R in $o(n)$ time and look for elements a and b in R with ranks $l = kn^{-\frac{1}{4}} - \sqrt{n}$ and $h = kn^{-\frac{1}{4}} + \sqrt{n}$. [The choice of the $\sqrt{n}$ term is dependant on the variance of an integral random variable X, which will be used in the application of Chebyshev's inequality. The other term is the *mean* of the *random variable* X.] We write $a = R_{(l)}$ and $b = R_{(h)}$. We can also define ranks $r_S(a)$ and $r_S(b)$ of a and b in S and actually compute these ranks by scanning the whole of S once.

The random variable X is the number of draws in R that are at most $S_{(k)}$, the kth smallest element of S. Since there are $n^{\frac{3}{4}}$ draws and the probability of selecting an element which is at most $S_{(k)}$ is $\frac{k}{n}$, the expectation $\mu_X = (\frac{k}{n})n^{\frac{3}{4}}$ and the variance $\sigma_X^2 = n^{\frac{3}{4}}(\frac{k}{n})(1 - \frac{k}{n}) \leq \frac{n^{\frac{3}{4}}}{4}$.

Depending upon the value of k, we define P in one of three different ways in the selection algorithm. For the first case where $n^{-\frac{1}{4}} \leq k \leq n - n^{-\frac{1}{4}}$, the set P is defined as the set of all elements of S in the range $[a, b]$.

For the second case where $k \leq n^{\frac{1}{4}}$, the expectation of X is at most 1 and $h \geq \sqrt{n}$. The set P is defined as the set of all elements in S that are less than $b = R_{(h)}$. Failure occurs when $S_{(k)} > b$. This means that X is at least $\sqrt{n}$ away from the expectation of X, yielding a Chebychev bound of at most $O(n^{-\frac{1}{4}})$ on the failure probability.

For the third case where $n - n^{\frac{1}{4}} \leq k$, the expectation $kn^{-\frac{1}{4}}$ of X is at least $n^{\frac{3}{4}}$. So, $l = kn^{-\frac{1}{4}} - \sqrt{n}$ is at least $n^{\frac{3}{4}} - \sqrt{n}$ and $a = R_{(l)}$. Choosing the set P therefore as the set of all elements in S that are more than a, we fail when $S_{(k)} < a$. This means that X is $\sqrt{n}$ less than the expectation $kn^{-\frac{1}{4}} \geq n^{\frac{3}{4}}$ of X, making it again possible to bound the error probability by $O(n^{-\frac{1}{4}})$ using Chebychev's inequality.

We now see how the set P may miss out $S_{(k)}$ in the first case. We show that the probability of this happening is $O(n^{-\frac{1}{4}})$. This can happen in one of two ways, based on the premise above defining l and a, and h and b; either (i) $S_{(k)} < a$, that is $X < l = kn^{-\frac{1}{4}} - \sqrt{n}$, implying $kn^{-\frac{1}{4}} - X \geq \sqrt{n}$ (for the left end of P), or (ii) $S_{(k)} > b$, that is $X > h = kn^{\frac{1}{4}} + \sqrt{n}$ or $X - kn^{-\frac{1}{4}} \geq \sqrt{n}$ (for the right end of P). The failure occurs therefore with Prob $[|X - kn^{-\frac{1}{4}}| > \sqrt{n}]$= Prob $[|X - kn^{-\frac{1}{4}}| > (2n^{\frac{1}{8}})\frac{n^{\frac{3}{8}}}{2}] \leq \frac{1}{(2n^{\frac{1}{8}})^2} = O(n^{-\frac{1}{4}})$ by Chebyshev's inequality for random variable X.

The above failure probability includes also the case where $S_{(k)}$ is not there in P but $|P| > 4n^{\frac{3}{4}}$. For the failure case where $S_{(k)} \in P$ but $|P| > 4n^{\frac{3}{4}}$, we need to see cases where $a < S_{(k)}$ or $b > S_{(k)}$. In no case do we wish $|P|$ to become so big as $4n^{\frac{3}{4}}$. We play safer replacing k by $k^- = k - 2n^{\frac{3}{4}}$ and $k^+ = k + 2n^{\frac{3}{4}}$, respectively. So, in the first case we use $a < S_{(k^-)}$, which implies the requirement $a < S_{(k)}$. Going by the algorithm and its definitions of l and a, we have $k^-n^{-\frac{1}{4}} - X < \sqrt{n}$ or $X + \sqrt{n} > k^-n^{-\frac{1}{4}} = kn^{-\frac{1}{4}} - 2\sqrt{n}$. So, $|kn^{-\frac{1}{4}} - X| > 3\sqrt{n}$. Now, we can use Chebyshev inequality again on X to show that this failure has probability no more than $O(n^{-\frac{1}{4}})$.

10 Using the local lemma for designing Las Vegas algorithms

We consider the k-sat problem, where we show that there is always a satisfying truth assignment provided each of the m clauses has exactly k literals, and each of the n variables appears in at most $2^{\frac{k}{50}}$ literals. We also consider computing such a truth assignment in (randomized) expected polynomial time. The probability that a random truth assignment would not satisfy a clause is $p = 2^{-k}$. The number of clauses sharing a variable with any clause would be no more than $d = k2^{\frac{k}{50}}$. So, the local lemma applies since $ep(d + 1) \leq 1$, thereby ensuring a satisfying truth assignment for the m clause k-sat formula. This discussion is based on the exposition in [6].

To compute a satisfying truth assignment, we work in two stages: the first one being a Las Vegas process with an expected constant number of rounds. The second one, a deterministic polynomial exhaustive enumeration step. In each step certain variables are fixed.

On fixing $\frac{k}{2}$ variables too, in some clause (in the first round), may not be satisfy the clause. Such a clause is called a *dangerous* clause, which *survives* into the second stage; some others clauses too *survive*, whose at most $\frac{k}{2}$ variables are fixed (but some other variable(s) is (are) shared with a dangerous clause). The *surviving* clauses are also satisfiable (by a truth assignment of the *deferred* variables), as can be shown using the local lemma again. Why? Explain.

Finding a truth assignment for the surviving clauses in the second stage is possible in polynomial time, assuming a favourable first stage execution, whereby, certain dependency conditions between surviving clauses are satisfied.

References

[1] B. Chazelle, The discrepancy method: Randomness and complexity, Cambridge University Press, 2000.

[2] T. H. Cormen, C. E. Leiserson, R. L. Rivest, Introduction to algorithms, Second Edition, Prentice-Hall India, 2003.

[3] Udi Manber, Introduction to algorithms: A creative approach, Addision-Wesley, 1989.

[4] M. Molloy and B. Reed, Graph colouring and the probabilistic method, Springer, 2002.

[5] Ketan Mulmuley, Computational Geometry: An Introduction Through Randomized Algorithsm, Prentice Hall, 1994.

[6] R. Motwani and P. Raghavan, Randomized algorithms, Cambridge University Press, 1995.

[7] J. Kleinberg and E. Tardos, Algorithm design, Pearson education, 2006.