CSC2420 Spring 2017: Algorithm Design, Analysis and Theory
Lecture 11

Allan Borodin

March 27, 2017

Announcements and todays agenda

- Announcements
  1. Assignment 2 was due today. I just saw some messages from Friday asking for some clarifications. I am pushiong back the due date to Wed. Please submit to me or Lalla.
  2. I have posted the first question for Assignment 3. I am planning to post more questions this week.
- Todays agenda
  1. By "popular demand" (i.e., one person asked me), a discussion of MapReduce
  2. Introduction to spectral methods and spectral graph theory

A discussion of MapReduce

I was asked at the end of the last class if I could say more about MapReduce models and algorithms. This is a topic of current interest, both in terms of its current practical application and popularity as a parallel programming paradigm, and as well as having some theoretical interest in the algorithm design and analysis community.

A discussion of MapReduce

I was asked at the end of the last class if I could say more about MapReduce models and algorithms. This is a topic of current interest, both in terms of its current practical application and popularity as a parallel programming paradigm, and as well as having some theoretical interest in the algorithm design and analysis community.

As stated last tine, the paradigm is implemented in the open source Hadoop. I would say that the initial and perhaps still main usage is with respect to data base applications.

The main advantage of MapReduce algorithms is that the programming environment is easy to understand.

A discussion of MapReduce

I was asked at the end of the last class if I could say more about MapReduce models and algorithms. This is a topic of current interest, both in terms of its current practical application and popularity as a parallel programming paradigm, and as well as having some theoretical interest in the algorithm design and analysis community.

As stated last tine, the paradigm is implemented in the open source Hadoop. I would say that the initial and perhaps still main usage is with respect to data base applications.

The main advantage of MapReduce algorithms is that the programming environment is easy to understand.

Since this is an area that I have not even remotely worked in (or lectured on), I am relying heavily on a few sources.

A few papers on MapReduce

- Zadeh abd Goel [2012] presnt a number of MapReduce algorithms to compute pairwise similarities (defined in different ways) between high dimensional sparse vectors (e.g. documents). These examples are relatively easy but interesting (and practical) examples of MapReducer.
- A high level CACM introduction by Ullman [2012]
- A more technical VLDB article by Afrati et al [2013] that presents a precise model in which both algorithms and limitations (.e. tradeoff results) are proved.
- I also mentioned a paper last class by Bahmani, Kumar and [2012] that converts a streaming algorithm to a MapReduce algorithm for the densest subgraph probelm
- There is a followup paper by Bahmani. Goel and Munagala [2014] on the densest subgraph problem focused entirely on MapReduce.
- There are also papers with precise models by Karloff et al [2010] and Beame et al [2013] that also feature algorithms and limitations.

The Map Reduce paradigm

As mentioned in last weeks lecture, MapReduce algorithms do computation on a large number of servers interconnected by a fast network. (There is no shared memory.) Each server performs computations (on the data they hold) and then exchange data. Map Reduce algorithms operate on (key,value) pairs in rounds (also called phases), each round consisting of three stages:

- Map: Transforms a (key,value) into one or several new (key,value) pairs.
- Shuffle: All the values associated with a given key are sent to the same (perhaps virtual) machine. This aspect is carried out automatically by the system and thereby allows the algorithm designer to avoid what can be complex implementation issues.
- Reduce: All values associated with a given key get batched into a multiset of (key,value) pairs

Measures of MapReduce complexity

Following the Goel and Munagala [2012] definitions of complexity measures for MapReduce computations, Zadeh and Goel study algorithms for similarity focusing on two hgih level complexity measures in a MapReduce round; namely *shuffle-size* and "reduce-key-complexity" defined as follows:

- Shuffle-size is the maximum number of outputs (i.e. the the number of (key,value) pairs) from a mapper.
- Reduce-key-complexity is the maximum time needed for a reducer to produce its output.

These measure depend on the algorithm but not on the details of the implementation (e.g., how many machines, number of mappers/reducers).

In general there will be a tradeoff between these measures and more generally, a tradeoff between the complesxity of a round and the number of rounds.

Cosine similarity

We consider one round algorithms for computing the cosine similarity between all pairs of high dimensional but sparse vectors. Here we are thinking of a document (resp. a twitter user) as a bit vector where each component indicates whether of not a given word is present in this document (resp. which other users a given user follows).

For the twitter example, they take as typical values: $N = 10^9$ as the number of all twitter users, $D = 10^7$ as the number of users being compared, and $L = 1000$, as the maximum number of other users being followed by any given user.

Alternatively, we can let $N = 10^9$ as the number of documents, $D = 10^7$ as the size of a dictionary of words, and $L = 20$ as the maximum number of words in say a tweet (rounding down 148 characters for simplicity).

Cosine similarity continued

Let $\#(w_i, w_j)$ denote the number of co-occurrences of w_i and w_j (resp. $\#w_i$ is the number of occurrences of w_i) in the collection of D documents/tweets.

Definition: The cosine similarity of (w_i, w_j) is defined as $\frac{\#(w_i,w_j)}{\sqrt{(\#w_i}\sqrt{(\#w_j}}$.
In a naive implementation, the shuffle-size would be $N \cdot L^2$ which is prohibitively large (i.e., 400 Billion for the example with $L = 20$).

That is, as presented in Zadeh and Goel,

Naive Mapper(t)

```
For all pairs $(w_1, w_2)$ in document $t$
    emit($(w_1, w_2, 1)$)
EndFor
```

Naive Reducer($(w_1, w_2), \langle r_1, \ldots, r_R \rangle$)

```
$s = \sum_{k=1}^{R} r_k$     % We have $r_i = 1$ for all $k$
Output (for a pair $(w_i, w_j)$) = $\frac{s}{\sqrt{(\#w_i}\sqrt{(\#w_j}}$
```

A more practical randomized MapReduce for cosine similarity

In practice, one would only be interested in similarities that exceed a certain threshold $\epsilon < 1$. In order to have the suffle size depend on D and L, we sample each occurence of a pair (w_1, w_2) with an appropriate probability. Choosing $R = \log D/\epsilon$, then for the stated $D = 10^7$ and $\epsilon = .1$, we have $R \approx 100$.

Randomized Mapper(t)

```
For all pairs $(w_1, w_2)$ in document $t$
    emit($(w_1, w_2, 1)$) with probability $\frac{R}{\sqrt{(\#w_i}\sqrt{(\#w_j}}$
EndFor
```

Corresponding Reducer($(w_1, w_2), \langle r_1, \ldots, r_R \rangle$)

```
$s = \sum_{k=1}^{R} r_k$     % We have $r_i = 1$ for all $k$
Output (for a pair $(w_i, w_j) = \frac{s}{R\sqrt{(\#w_i}\sqrt{(\#w_j}}$
```

Conclusion for the randomized MaReduce for cosine similarity

The Zadeh and Goel paper proves the following result:

Theorem: Ramdomized MapReduce for cosine similarity

- The output is an estimator for the cosine similarity
- It is "accurate with high probability" (i.e. we can obtain an (ϵ, δ) result by setting the probability to $O(R)$).
- The shuffle size is $O(DL\log(D)/\epsilon)$
- The Reduce-key-complexity is $O(\log D/\epsilon)$

Note that to output highly similar pairs, there can be $O(DL)$ such pairs so that these bound are within a $O(\log D)$ factor of optimality.

Approximating the densest subgraph problem in MapReduce

The densest subgraph problem is defined as follows:

Given a graph $G = (V, E)$, find a subset $V' \subseteq V$ so as to maximize $\frac{|e: u, v \in V'|}{|V'|}$; that is, to maximize the density (or equivalently the average degree) in a subgraph of G.

There is also a directed graph version of this problem. We will consider the undirected case.

Approximating the densest subgraph problem in MapReduce

The densest subgraph problem is defined as follows:

Given a graph $G = (V, E)$, find a subset $V' \subseteq V$ so as to maximize $\frac{|e:u,v \in V'|}{|V'|}$; that is, to maximize the density (or equivalently the average degree) in a subgraph of G.

There is also a directed graph version of this problem. We will consider the undirected case.

The densest subgraph problems can be solved in polynomial time by a flow based algorithm as described in Lawler's 1976 text and improved in Gallo et al [1989]. There is also an LP duality based optimal method given in Charikar [2000] that is the starting point for the MapReduce $(1+\epsilon)$ approximation algorithm due to Bahmani, Goel and Munagala [2014].

The $(1+\epsilon)$ approximation follows a MapReduce $2(1+\epsilon)$ approximation by Bahmani, Kumar and Vassilvitskii [2012] based on the Charikar greedy 2-approximation. (Note that these papers use approximation ratios ≥ 1.)

Some comments on the densest subgraph problem

One immediate application is that the densest subgraph can be used to identify a “community” in the web or a social network. There are other applications as well in biological networks.

Some comments on the densest subgraph problem

One immediate application is that the densest subgraph can be used to identify a "community" in the web or a social network. There are other applications as well in biological networks.

The k-densest subgraph problem asks for the densest subgraph V' of size $|V'| = k$. As far as I know the best approximation known for the k-densest subgraph problem is $O(n^{\frac{1}{3}+\delta})$.

Some comments on the densest subgraph problem

One immediate application is that the densest subgraph can be used to identify a "community" in the web or a social network. There are other applications as well in biological networks.

The k-densest subgraph problem asks for the densest subgraph V' of size $|V'| = k$. As far as I know the best approximation known for the k-densest subgraph problem is $O(n^{\frac{1}{3}+\delta})$.

Obviously if one can (approximately) solve the k-densest subgraph problem then one can (approximately) solve the "densest subgraph with at least (resp. at most) k vertices. But the converse is not apparentlly true. Andersen and Chellapilla [2009] show the following:

- There is a 3-approximation algorithm for the "at least k vertices" variant.
- If there is a γ approximatin for the "at most k vertices" variant then there is an $\frac{\gamma^2}{8}$ approximation for the k-densest subgraph problem

The Charikar LP duality based method

We will use the notation and sketch the development in Bahmani et al [2014]. In addition to the new MapReduce approximation ratio, this development is interesting as it nicely integrates the Plotkin, Shmoys and Tardos [1991] framework for approximating fractional packing and covering problems which in turn is desribed in Arora et al [2012] as an application of the multiplicative weights paradigm.

The Charikar LP duality based method

We will use the notation and sketch the development in Bahmani et al [2014]. In addition to the new MapReduce approximation ratio, this development is interesting as it nicely integrates the Plotkin, Shmoys and Tardos [1991] framework for approximating fractional packing and covering problems which in turn is desribed in Arora et al [2012] as an application of the multiplicative weights paradigm.
The Charikar LP is as follows:

Maximize $\sum_e y_e$ subject to
$y_e \leq x_v \quad \forall e \in E, e$ incident on $v \in V$
$\sum_v x_v \leq 1 \quad \forall v \in V$
$x_v, y_e \geq 0 \quad \forall v \in V, e \in E$

The Charikar LP duality based method

We will use the notation and sketch the development in Bahmani et al [2014]. In addition to the new MapReduce approximation ratio, this development is interesting as it nicely integrates the Plotkin, Shmoys and Tardos [1991] framework for approximating fractional packing and covering problems which in turn is desribed in Arora et al [2012] as an application of the multiplicative weights paradigm.
The Charikar LP is as follows:

Maximize $\sum_e y_e$ subject to
$y_e \leq x_v \quad \forall e \in E, e$ incident on $v \in V$
$\sum_v x_v \leq 1 \quad \forall v \in V$
$x_v, y_e \geq 0 \quad \forall v \in V, e \in E$

Charikar shows

- For any $S \subseteq V$, the value of the LP v is at least as large as the value of the largest densest subgraph.
- Given a feasible solution of the LP with value v, one can efficiently construct an $S \subseteq V$ such that the value of the largest densest subgraph is at least v. That is, the LP can be optimally "rounded".

The dual of the Charikar LP

Bahmani et al [2014] do not using the Charikar rounding but instead consider the dual of the LP (formulated as a decision problem) which is a maximum concurrent multi-commodity flow (MCMF) problem::

Minimize D subject to
$\alpha_{eu} + \alpha_{ev} \geq 1 \quad \forall e = (u, v) \in E$
$\sum_{e:e \text{ incident on } v} \alpha_{ev} \leq D \quad \forall v \in V$
$\alpha_{ev} \geq 0 \quad \forall e \in E, v \in V$

By duality, D is at least the value of the optimal primal LP (= the optimal value of the densest subgraph).

Outline of the Bahmani et al algorithm

Basdd on the algorithm of Young [1995], the Arora et al multiplicative weights solving of the decision MCMF is framed in terms of the following linear covering problem subject to a convex set:

The Covering Problem

Does there exist an $\mathbf{x} \in P$ such that $A\mathbf{x} \geq 1$ where A is an $r \times s$ matrix and P is convex set in $\mathbb{R}^s$: $A\mathbf{x} \geq 0$ for all $\mathbf{x} \in P$.

It is assume that there is an "oracle" for maximizing a linear combination of the linear constraints. The running time of the algorithm is measured in terms of *width* ρ of the problem where $\rho = \max_i \max_{\mathbf{x} \in P} \mathbf{a}_i \mathbf{x}$.

Outline of the Bahmani et al algorithm

Basdd on the algorithm of Young [1995], the Arora et al multiplicative weights solving of the decision MCMF is framed in terms of the following linear covering problem subject to a convex set:

The Covering Problem

Does there exist an $\mathbf{x} \in P$ such that $A\mathbf{x} \geq 1$ where A is an $r \times s$ matrix and P is convex set in $\mathbb{R}^s$: $A\mathbf{x} \geq 0$ for all $\mathbf{x} \in P$.

It is assume that there is an "oracle" for maximizing a linear combination of the linear constraints. The running time of the algorithm is measured in terms of *width* ρ of the problem where $\rho = \max_i \max_{\mathbf{x} \in P} \mathbf{a}_i \mathbf{x}$.

For the MCMF decision problem, the goal is to decide if the linear constraints $\alpha_{eu} + \alpha_{ev} \geq 1$ can be satisifed subject to the convex set $P(D)$ constraint:

$\sum_{e:e \text{ incident on } v} \alpha_{ev} \leq D \quad \forall v \in V$

$\alpha_{ev} \geq 0 \quad \forall e \in E, v \in V$

For the densest subgraph problem, the width corresponds to the maximum degree d_{max} in the input graph G.

Outline of the Bahmani et al algorithm continued

In solving the densest subgraph problem using this framework, the number of multiplicative weigths iterations (corresonding to rounds in MapReduce) will depend on the width.

Bahmani et al introduce a *width modulation* technique whereby extra capacity constraints $\alpha_{ev} \leq q$ ($\forall e \in E, v \in V$) are added to the convex set constraints which reduces the width to $2q$.

Outline of the Bahmani et al algorithm continued

In solving the densest subgraph problem using this framework, the number of multiplicative weigths iterations (corresonding to rounds in MapReduce) will depend on the width.

Bahmani et al introduce a *width modulation* technique whereby extra capacity constraints $\alpha_{ev} \leq q$ ($\forall e \in E, v \in V$) are added to the convex set constraints which reduces the width to $2q$.

As they explain, there is a tradeoff between the running time (rounds) where small width helps and having an efficient rounding scheme where large witdh helps.

Outline of the Bahmani et al algorithm continued

In solving the densest subgraph problem using this framework, the number of multiplicative weigths iterations (corresonding to rounds in MapReduce) will depend on the width.

Bahmani et al introduce a *width modulation* technique whereby extra capacity constraints $\alpha_{ev} \leq q$ ($\forall e \in E, v \in V$) are added to the convex set constraints which reduces the width to $2q$.

As they explain, there is a tradeoff between the running time (rounds) where small width helps and having an efficient rounding scheme where large witdh helps.

The result of this non trivial development is a MapReduce algorithm satisfying $O(\log n/\epsilon)$ rounds, with shuffle complexity $O(|E|)$ and reduce-size-complexity $O(d_{max})$ in each round.

Some other MapReduce algorithms

As previously mentioned, for the densest subgraph problem, Bahmani et al [2012] adapted the Charikar “reverse greedy” algorithm to derive an $O(\log n)$ round MapReduce algorithm with approximation $2(1+\epsilon)$.

Some other MapReduce algorithms

As previously mentioned, for the densest subgraph problem, Bahmani et al [2012] adapted the Charikar “reverse greedy” algorithm to derive an $O(\log n)$ round MapReduce algorithm with approximation $2(1+\epsilon)$.

It may seem inconsistent that greedy style algorithms (which seem inherently sequential) can be utilized to derive MapReduce algorithms with a relatively small number of passes.

But, in fact, the Bahmani et al algorithm is representative of a number of such algorithms. See Chierichetti et al [2010] for Max-Cover, Lattanzi et al [2011] for vertex cover and maximal matching problems, and Ene et al [2011] for the k-center problem.

Kumar et al [2013] provide a unifying idea in how natural greedy algorithms can be modified so as to be implemented in the Streaming and MapReduce models. As they explain, the basic idea is to be able to identify a large subset of data items that can be removed in a round without significantly changing the desired objective value.

Some other MapReduce algorithms

As previously mentioned, for the densest subgraph problem, Bahmani et al [2012] adapted the Charikar "reverse greedy" algorithm to derive an $O(\log n)$ round MapReduce algorithm with approximation $2(1+\epsilon)$.

It may seem inconsistent that greedy style algorithms (which seem inherently sequential) can be utilized to derive MapReduce algorithms with a relatively small number of passes.

But, in fact, the Bahmani et al algorithm is representative of a number of such algorithms. See Chierichetti et al [2010] for Max-Cover, Lattanzi et al [2011] for vertex cover and maximal matching problems, and Ene et al [2011] for the k-center problem.

Kumar et al [2013] provide a unifying idea in how natural greedy algorithms can be modified so as to be implemented in the Streaming and MapReduce models. As they explain, the basic idea is to be able to identify a large subset of data items that can be removed in a round without significantly changing the desired objective value.
How would you modify the Charikar reverse greedy algorithm?

A brief introduction to spectral methods

- Like other topics in the course, spectral methods and in particular spectral graph theory (and, in particular, spectral graph algorithms) is really a topic in itself.
- Spectral methods are becoming more and more important with applications to many areas of research.
- When we say *spectral method*, we mean algorithmic methods relying on the eigenvalues and eigenvectors of a matrix. In particular, we will just highlight some results relating to matrices coming from undirected graphs.
- One of the most active and influential researchers in this area is Dan Spielman. His Fall, 2015 course notes on spectral graoh theory can be found at http://www.cs.yale.edu/homes/spielman/561/. I have posted a tutorial by Dan Spielman on the course web page.
- I will just briefly introduce some terminology and give a glimpse of some applications of spectral graph theory. Spielman's course notes and tutorial will, of course, provide many further applications.

Spectral graph theory

- For undirected graphs, the adjacency matrix $A(G)$ of a graph G is a real symmetric matrix.
- A non-zero (column) vector x is an eigenvector of A with eigenvalue λ if $Ax = \lambda x$.
- The spectrum of A or a graph G refers to the set of eigenvalues of A (resp $A(G)$).
- When A is a real symmetric matrix, then all the eigenvalues are real and there is an orthonormal basis of R^n consisting the eigenvectors of A. That is, the eigenvectors are orthogonal to each other and each normalized to length $= 1$.
- The question is what useful information about a graph can the spectrum provide?

The Laplacian

- In spectral graph theory, it is often better to consider the Laplacian of a graph which is defined as $L(G) = D(G) - A(G)$ where $D(G)$ is the diagonal matrix whose entries are the degrees of the vertices.
- In particular if G were d regular, then any eigenvector of $A(G)$ with eigenvalue λ is an eigenvector of $L(G)$ with eigenvalue $d - \lambda$ and vice versa.
- The nice property of the Laplacian $L(G)$ is that it is a positive semi-definite matrix which implies that all its eigenvalues are non-negative.
- Furthermore, G is connected if and only if $\lambda = 0$ is an eigenvalue of $L(G)$ with multiplicity 1. More generally, G has k connected components iff 0 is an eigenvalue of multiplicity k.
- Why is this interesting? Ordering so that $\lambda_1 \leq \lambda_2 \ldots \leq \lambda_n$, we can think of the two smallest eigenvalues being close iff the graph is "close" to being disconnected iff there is a "sparse cut".

Sparse cuts

- Recall that a cut in a graph $G = (V, E)$ is a partition of the vertices V into S and $V \setminus S$ (or equivalently the cut set of edges , that is, $cut(S) = \{e = (u, v) | u \in S, v \in V \setminus S\}$).
- We previously discussed min cuts in a graph and how they can be optimally computed using the the max flow-min cut theorem and (say) a Ford Fulkerson based algorithm. (For edge weighted graphs, Ford Fulkerson computes a cut of minimum weight.)
- Our goal now is to produce "balanced sparse cuts". That is, we want to view the size of a cut relative to the sizes of S and $V \setminus S$. Such balanced cuts have applications to algorithms that work by decomposing a graph into roughly equal parts.
- The *conductance* $\phi(S)$ of a set S is defined as: $\frac{|cut(S)|}{\min\{vol(S), vol(V \setminus S)\}}$ where $vol(S) = \sum_{u \in S} degree(u)$.

Conductance

- Sometimes conductance is defined as $\frac{|cut(S)|}{|S| \cdot |V \setminus S|}$. These quantities are within a factor of 2.
- The conductance $\phi(G)$ of a graph is the $\min_{S:|S| \leq n/2} \phi(S)$. Computing the conductance of a graph is a well studied formulation of the *sparsest cut problem*. It is NP-hard and the best known approximation for about 15 years was the Leighton Rao $O(\log |V|)$ for about 15 years and then improved by Arora, Leighhton and Rao to $O(\sqrt{\log |V|})$.

Cheeger's Inequality

Cheeger's inequality has been called *the most important result in spectral graph theory*.

- To state this result it is useful to consider the following normalized adjacency and Laplacian matrices:
  $A' = D^{-1/2}AD^{-1/2}$ and $L' = D^{-1/2}LD^{-1/2}$
- Here $D^{-1/2}$ is the diagonal matrix with diagonal entries $d_{i,i} = degree(v_i)^{-1/2}$
- Letting $\{\alpha_i\}$ (resp. λ'_i) denote the eigenvalues of A' (resp. L'), it follows that
  $1 \geq \alpha_1 \geq \alpha_2 \ldots \geq \alpha_n \geq -1$ and $0 = \lambda'_1 \leq \lambda'_2 \ldots \leq \lambda'_n \leq 2$.

Cheeger's inequality continued

- The *spectral gap* is the difference bewteen α_1 and α_2 (or between λ'_1 and λ'_2).
- The spectral gap is closely related to conductance as well as the graph expansion properties and random walk properties.

Cheeger's inequality

$\lambda'_2/2 \leq \phi(G) \leq \sqrt{2\lambda'_2}$

- The spectral gap is also closely related to the important concept of *expander graphs*.
- Intuitively, expander graphs $G = (V, E)$ satisfy the property that for all (not too large) subsets $S \subset V$, the size of the neighbourhood of S is sufficiently larger than the size of S.

Expander graphs and applications

- Expander graphs have many applications (e.g. in coding theory, random walks, error probability amplification and derandomization).
- There are various combinatorial parameterized definitions and we will soon present two specific definitions.
- Expansion is (with high probability) a property of random graphs and in a sense expander graphs are often surrogates for random graphs.
- There is a considerable amount of research on the construction of *explicitly defined* expander graphs of small degree.
- We will see that, algebraically, expander graphs can also be characterized as graphs having a suitable spectral gap, and also equivalently as graphs having rapid (i.e. $O(\log n)$) mixing time to equilibrium in a random walk.

Two specific combinatorial expander definitions

Two expander definitions that occur are the following:

An $(n, d, c))$ node expander

A (n, d, c) node expander is an n node d-regular biparititie multi-graph $G = (X, Y, E)$ with $|X| = |Y| = n/2$ such that any subset $S \subseteq X$ satsifies —Neighbourhood of $S| \ \geq (1 + c(1 - \frac{2|S|}{n})|S|$.

An $(n, d, c))$ edge expander

(N, d, c) edge expander is an n node, d-regular multi-graph $G = (V, E)$ such that any subset $S \subseteq V$ with $|S| \leq n/2$ has at least $cd|S|$ edges between S and $V \setminus S$.

- In general, one wants small degree d and a constant $c > 0$.
- Most random d-regular bipartite garphs are such expanders but we usually need explicitly constructed exapnders (which are known).

Expanders and the spectral gap

Given this type of edge expander graph, we have the following seminal relation with the spectral gap (due to Noga Alon). Here we let $\{\lambda_i\}$ be the eigenvalues of the adjacency matrix $A(G)$.

Relating expansion and spectral gap

If G is a (n, d, c) edge expander then $\lambda_1 = d$, then
$\frac{1}{2}(1 - \lambda_2/d) \leq c \leq \sqrt{2(1 - \lambda_2/d)}$

Random walks and node expanders

- To study random walks, it is convenient to now normalize the adjacency matrix to form $P = A(G)/d$.
- Since G is bipartite there is no stationary distribution so to make the process aperiodic, define $Q = (I + P)/2$ meaning that with probability $1/2$, the process stays in the same state. Q now represents a doubly stochastic Markov process with a uniform stationary distribution $\{\pi_j\}$.
- The eigenvalues $\{\lambda'_i\}$ now satisfy $\lambda'_i = \frac{1+\lambda_i/d}{2}$ so that $1 = \lambda'_1 \geq \lambda'_2 \ldots \lambda'_n = 0$. Now suppose we have an expander with $\lambda'_2 \leq 1 - \frac{\epsilon}{2d}$.

Fast convergence

$\max_j \frac{|q_j^t - \pi_j|}{\pi_j} \leq n^{1.5}(\lambda'_2)^t$ where
q_j^t is the probability of being in state j at time t.

- This implies convergence in $O(\log n)$ steps. See Motwani and Raghavan, Section 6.7.2.

Application to probability correctness amplification

- Consider a polynomial time RP (1-sided error) set (e.g. the composite numbers, represented in binary or decimal) or BPP (2-sided error) set.
- Suppose the algorithm has error probability c (e.g. $c = 1/4$) using n random bits.
- For an RP (resp. BPP) set we can amplifiy the error bound to c^k by doing k independent trial and hence using kn bits.
- Suppose we have an explicit expander with constant degree (say degree $d = 8$). Consider a random walk on an (exponential size) expander where the nodes correspond to n bit strings.
- Since the stationary distribution is the uniform distribution, the idea is to do such a random walk and sample some $O(k)$ nodes, sampling every b steps (for some appropriate b)
- Starting at a random node (using n bits), we will only need $n + O(k)$ bits to obtain enough trials and the desired c^k error.

$\#P$ counting problems

- Recall that an NP set L can be defined by $L = \{x | R(x, y)\}$ where R is a polynomial time verification algorithm and y is a polynomial length certificate. (Similarly, RP sets are those where the fraction of certificates is some constant $c > 0$.)
- A $\#P$ counting problem $\#L$ is one that can be defined as the number of certficates for an NP set L.
- For example, $\#SAT$ is the counting problem that outputs the number $\#x$ of satsifying formulas for an input formula x encoding a CNF formual F.
- Clearly if $\#L$ is polynomial time computable, then so is L so we certainly do not expect counting problems coresponding to NP complete sets to be computable in polynomial time.

$\#P$ complete counting problems

- Clearly $\#SAT$ is a $\#P$ complete problem in the sense that that any NP counting problem can be reduced to this problems.
- But even if L is polynomial time computable, it does not show that $\#L$ is polynomial time.
- For example, given a proper DNF formula F, it is immediately clear that F is satisfiable and given a bipartite graph G, we can efficiently determine if G has a perfect matching. However, both #DNF-SAT and #bipartite-matching are $\#P$ complete counting problems.

Approximating $\#P$ complete counting problems

- Given the hardness of $\#P$ complete problems, what we can hope for is to compute an estimate of $\#x$ for an input instance x. We want an estimate to fall in the range $[(1-\epsilon)\#x, (1+\epsilon)\#x]$ for every input instance x.
- For a randomized algorithm we would want such an estimate to be obtained with probability error some $\delta < 1$.
- Given that we can encode an ϵ in $\log 1/\epsilon$ bits, we might hope for such an algorithm to have time bounded by a polynomial is $n, \log 1/\epsilon$ and for randomized algorithms also in time $\log 1/\delta$. But it turns out that this would imply $P = \#P$ (or BPP $= \#P$ for randomized algorithms).
- Instead we will be happy to get algorithms with run time polynomial in $n, (1/\epsilon,$ and $\log 1/\delta$. Such a randomized algorithm are called an FPRAS algorithm.

Approximate counting

- Unlike the BPP= P question, it turns out that there are some counting problems (e.g. volume estimnation of a convex body in n dimensions) for which randomization provably helps.
- When the underlying decision problem L is in P, there is a natural randomized approach, which we can call "basic Monte Carlo sampling".
- Namely, sample from the space of all possible inputs and let the fraction of good inputs (i.e. those in the set L) be an estimate of the fraction of all inputs that are good
- Here then is the "abstract estimation problem": Let f be a Boolean function over a universe U such that $f(U)$ can be efficiently computed for any $u \in U$. Assume that U can be sampled uniformly at random. We want to estimate the size of $G = \{u|f(u) = 1\}$.

The natural estmation approach and its limitation

- Let $Y_i = 1$ iff $f(u_i) = 1$ where u_i is the i^{th} sampled input. Choose m random samples and then estimate $|G|$ by $Z = |U| \sum_i Y_i/m$.
- Let $\rho = |G|/|U|$. Then the basic Monte Carlo estimation is an FRPAS if $m \geq \frac{4}{\epsilon^2 \rho} \ln \frac{2}{\delta}$.
- So what is the limitation?

What happens when ρ is small or when it is hard to sample uniformly

- Whe ρ is small (as it can be for say DNF-SAT or bipartite matchings) or when it is not clear how to sample uniformly (e.g. trying to sample from the set M_k of all size k matchings and then using this sampling to recursively estimate the number m_k of size k matchings.
- When the minimum degree is at least $n/2$ here is the approach. The idea is to sample uniformly from $M_k \cup M_{k-1}$. This will give estimates of $r_k = m_k/m_{k-1}$
- Noting that $m_1 = |E|$, the desired estimate is $m_n = m_1 \Pi_{i=2}^{n} r_i$.
- To do the sampling, one needs to construct a doubly stochastic matrix (as in the amplification analysis) such that the resulting Markov process is rapidly mixing.

Applications of the smallest eigenvalue

- We return to the normalized adjacency matrix $A'(G)$ with eigenvalues $\{\alpha_i\}$.
- It can be shown that G is bipartitie iff $\alpha_1 = -\alpha_n$.
- Recalling that the eigenvalues of A' are in [1,-1], the matrix $I + A'$ has eigenvalues in [0,2].
- A graph $G = (V, E)$ is "close to bipartite" if the smallest value of $I + A'$ is close to 0.
- Another way to think about being close to bipartite is to have a large maximum $cut(S)$ relative to $|E|$.
- The best approximation for this NP hard problem is the same $\approx .878$ achieved by the same kind of SDP we saw for Max-2-Sat. This ratio is the best possible assuming the UGC.
- The obvious greedy algorithm for max cut (or the naive random algorithm) gives a 1/2 approximation and it remained an open problem to beat 1/2 by a "combinatorial algorithm". Trevisan uses a spectral based algorithm that acheievs ratio .531 which was then improved by Sato to ,614. Can we achieve a combinatorial 3/4 approximation for Max-Cut by a combinational algorithm as known

Continued spectral applications

- Steuer gives some evidence for and against the UGC. The evidence against is an improved UGC algorithm that exploitts the entire spectrum (of eigenvalues).
- More classical results go back to Hoffman who related the independence number $\alpha(G)$ and chromatic number $\chi(G)$ of a graph to the spectrum.
- Namely, for $\{\lambda_i\}$ again being the eigenvalues for the adjcency matrix, $\alpha(G) \leq \frac{-\lambda_n}{d_{max}-\lambda_n}$ and $\chi(G) \geq 1 - \frac{\lambda_1}{\lambda_n}$.