15-859M: Randomized Algorithms Anupam Gupta
Lec 13/14: Metric Embeddings March 2, 2011

1 Metric Spaces

A metric space is a set V of points, with a distance function $d : V \times V \to \mathbb{R}_{\geq 0}$ that satisfies $d(x, x) = 0$ for all $x \in V$, *symmetry* (i.e., $d(x, y) = d(y, x)$), and the *triangle inequality* (i.e., $d(x, y) + d(y, z) \geq d(x, z)$ for all $x, y, z \in V$). Most of the computer science applications deal with finite metrics, and then n denotes the number of points $|V|$.

There are many popular problems which are defined on metric spaces:

- The Traveling Salesman Problem (TSP): the input is a metric space, and the goal is to find a tour $v_1, v_2, \ldots, v_n, v_{n+1} = v_0$ on all the n nodes whose total length $\sum_{i=1}^{n} d(v_i, v_{i+1})$ is as small as possible. This problem is sometimes defined on non-metrics as well, but most of the time we consider the metric version.

  The best approximation algorithm for the problem is a $(3/2 - \epsilon)$-approximation due to Oveis-Gharan, Saberi and Singh (2010). Their paper uses randomization to beat the 3/2-approximation of Cristofides (1976), and make progress on this long-standing open problem. The best hardness result for this problem is something like 1.005 due to Papadimitriou and Vempala.

- The k-Center/k-Means/k-median problems: the input is a metric space (V, d), and the goal is to choose some k positions F from V as "facilities", to minimize some objective function. In *k-center*, we minimize $\max_{v \in V} d(v, F)$, the largest distance from any client to its closest facility; here, we define the distance from a point v to a set S as $d(v, S) := \min_{s \in S} d(v, s)$. In *$k$-median*, we minimize $\sum_{v \in V} d(v, F)$, the total (or equivalently, the average) distance from any client to its closest facility. In *k-means*, we minimize $\sum_{v \in V} d(v, F)^2$, the average squared distance from any client to its closest facility. (Note: to see why these problems are called what they are, consider what happens for the 1-means/medians problem on the line.)

  The best algorithms for k-center give us a 2-approximation, and this is the best possible unless P=NP. The best k-median algorithm gives an $(3 + \epsilon)$-approximation, whereas the best hardness known for the version of the problem stated above is $(1 + 1/e)$ unless P=NP. For k-means, gap between the best algorithm and hardness results is worse for general metric spaces. For geometric spaces, better algorithms are known for k-means/medians.

- The k-server problem: this is a classic online problem, where the input is a metric space (given up-front); a sequence of requests $\sigma_1, \sigma_2, \cdots$ arrives online, each request being some point in the metric space. The algorithm maintains k servers, one each at some k positions in the metric space. When the request σ_t arrives, one of the servers must be moved to σ_t to serve the request. The cost incurred by the algorithm in this step is the distance moved by the server, and the total cost is the sum of these per-step costs. The goal is to give a strategy that minimizes the total cost of the algorithm.

The best algorithm for k-server is a $2k-1$-competitive deterministic algorithm due to Koutsoupias and Papadimitriou. Since k-server contains paging as a special case (why?), no deterministic algorithm can do better than k-competitive. It is a long-standing open problem whether we can do better than $2k-1$CC deterministically—but far more interesting is the question of whether randomization can help beat $2k-1$; the best lower bound against oblivious adversaries is $\Omega(\log k)$, again from the paging problem.

1.1 Approximating Metrics by Trees: Attempt I

A special kind of metric space is a *tree metric*: here we are given a tree $T = (V, E)$ where each edge $e \in E$ has a length ℓ_e. This defines a metric (V, d_T), where the distance $d_T(x, y)$ is the length of the (unique) shortest path between x and y, according to the edge lengths ℓ_e. In general, given any graph $G = (V, E)$ with edge lengths, we get a metric (V, d_G).

Tree metrics are especially nice because we can use the graph theoretic idea that it is "generated" by a tree to understand the structure of the metric better, and hence give better algorithms for problems on tree metrics. For instance:

- TSP on tree metrics can be solved exactly: just take an Euler tour of the points in the tree.
- k-median can be solved exactly on tree metrics using dynamic programming.
- k-server on trees admits a simple k-competitive deterministic algorithm.

So if all metrics spaces were well-approximable by trees (e.g., if there were some small factor α such that for every metric $M = (V, d)$ we could find a tree T such that

$$d(x, y) \leq d_T(x, y) \leq \alpha d(x, y) \tag{1}$$

for every $x, y \in V$, then we would have an α-approximation for TSP and k-median, and an αk-competitive algorithm for k-server on all metrics. Sadly, this is not the case: for the metric generated by the cycle graph C_n, the best factor we can get in (1) is $\alpha = n - 1$. This is what we would get if we just approximated the tree by a line.

So even for simple metrics like that generated by the cycle (on which we can solve these problems really easily), this approach hits a dead-end really fast. Pity.

1.2 Approximating Metrics by Trees: Attempt II

Here's where randomization will come to our help: let's illustrate the idea on a cycle. Suppose we delete a uniformly random edge of the cycle, we get a tree T (in fact, a line). Note that the distances in the line are at least those in the cycle.

How much more? For two vertices x, y adjacent in the cycle, the edge (x, y) still exists in the tree with probability $1 - 1/n$, in which case $d_T(x, y) = d(x, y)$; else, with probability $1/n$, x and y lie at distance $n - 1$ from each other. So the expected distance between the endpoints of an edge (x, y) of the cycle is

$$(1 - 1/n) \cdot 1 + 1/n \cdot n - 1 = 2(1 - 1/n) \cdot d(x, y)$$

And indeed, this also holds for any pair $x, y \in V$ (check!),

$$E_T[d_T(x, y)] \leq 2(1 - 1/n) \cdot d(x, y)$$

But is this any good for us?

Suppose we wanted to k-median on the cycle, and let F^* be the optimal solution. For each x, let f_x^* be the closest facility in F^* to x; hence the cost of the solution is:

$$OPT = \sum_{x \in V} d(x, f_x).$$

By the expected stretch guarantee, we get that

$$\sum_{x \in V} E_T[d_T(x, f_x)] < 2\,OPT.$$

I.e., the expected cost of this solution F^* on the random tree is at most $2\,OPT$. And hence, if OPT_T is the cost of the optimal solution on T, we get

$$E_T[OPT_T] \leq 2\,OPT$$

Great—we know that the optimal solution on the random tree does not cost too much. *And* we know we can find the optimal solution on trees in poly-time.

Let's say $F_T \subseteq V$ is the optimal solution for the tree T, where the closest facility in F_T to x is f_x^T, giving $OPT_T = \sum_x d_T(x, f_x^T)$. How does this solution F_T perform back on the cycle? Well, each distance in the cycle is less than that in the tree T, so the expected cost of solution F_T *on the cycle* will be

$$E\left[\sum_x d(x, f_x^T)\right] = \sum_x E[d(x, f_x^T)] \leq \sum_x E[d_T(x, f_x^T)] = E_T[OPT_T] \leq 2\,OPT.$$

And we have a randomized 2-approximation for k-median on the cycle!

1.3 Popping the Stack

To recap, here's the algorithm: pick a random tree T from some nice distribution. Find an optimal solution F_T for the problem, using distances according to the tree T, and output this set as the solution for the original metric.

And what did we use to show this was a good solution? That we had a distribution over trees such that

- every tree in the distribution had distances no less than that in the original metric, and
- the expected tree distance between any pair $x, y \in V$ satisfies $E_T[d_T(x, y)] \leq \alpha \cdot d(x, y)$ for some small α; here $\alpha = 2$.

And last but not least

- that the objective function was linear in the distances, and so we could use linearity of expectations.

Note that TSP, k-median, k-server, and many other metric problems have cost functions that are linear in the distances, so as long as the metrics we care about can be "embedded into random trees" with small α, we can translate algorithms on trees for these problems into (randomized) algorithms for general metrics! This approach gets used all the time, and is worth remembering. (BTW, note that this general approach does not work for non-linear objective functions, like k-center, or k-means.)

But can we get a small α in general? In the next section, we show that for any n-point metric with *aspect ratio* $\frac{\max d(x,y)}{\min d(x,y)} = \Delta$, we can get $\alpha = O(\log n \log \Delta)$; and we indicate how to improve this to $O(\log n)$, which is the best possible!

2 Embeddings into Distributions over Trees

In this section, we prove the following theorem using tree embeddings (and then, in the following section, we improve it further to $O(\log n)$).

Theorem 1 *Given any metric (V, d) with $|V| = n$ and aspect ratio Δ, there exists a efficiently sampleable distribution $\mathcal{D}$ over spanning trees of V such that for all $u, v \in V$:*

1. *For all $T \in$ Support$(\mathcal{D})$, $d_T(u, v) \geq d(u, v)$, and*
2. *$\mathbb{E}_{T\sim\mathcal{D}}[d_T(u, v)] \leq O(\log n \log \Delta)\ d(u, v)$.*

To prove this theorem, we will use the idea of a *low diameter decomposition.* Given a metric space (V, d) on $|V| = n$ points and a parameter $r \in \mathbb{R}_+$, a *(randomized) low-diameter decomposition* is an efficiently sampleable probability distribution over partitions of V into $S_1 \uplus S_2 \uplus S_3 \uplus \cdots \uplus S_t$ such that

1. *(Low Radius/Diameter)* For all S_i, there exists $c_i \in V$ such that for all $u \in S_i$, $d(c_i, u) \leq r/2$. Hence, for any $u, v \in S_i$, $d(u, v) \leq r$.
2. *(Low Cutting Probability)* For each pair u, v, $\Pr[u, v \text{ lie in different } S_i's] \leq \beta \, \frac{d(u,v)}{r}$ with $\beta = O(\log n)$.

We'll show how to construct such a decomposition in the next section (next lecture), and use such a decomposition to prove Theorem 1.

Consider the following recursive algorithm, which takes as input a pair (U, i) where $U \subseteq V$ is a set of vertices of diameter at most 2^i, and returns a rooted tree (T, r).

```
TreeEmbed$(U, i)$:
    1. Apply the low-diameter decomposition to $(U, d)$ with the parameter $r = 2^{i-1}$
       to get the partition $S_1, \ldots, S_t$.
    2. Recurse: Let $(T_j, root_j) \leftarrow$ TreeEmbed$(S_j, i-1)$. As a base case, when $S_i$
       is a single point, simply return that point.
    3. For every tree $T_j$ with $j > 1$, add the edge $(root_1, root_j)$ with length $2^i$.
       This is a new tree which we denote $T$.
    4. Return the tree/root pair $(T, root_1)$.
```

Recall that since the low diameter decomposition is randomized, this algorithm defines a distribution over trees over U. To build the tree for V, we first rescale so that for all $u, v \in V$, $d(u, v) \geq 1$ and $d(u, v) \leq \Delta \approx 2^\delta$. We define the distribution $\mathcal{D}$ as the one obtained by calling **TreeEmbed**(V, δ).

Lemma 2 *For all $x, y \in V$, $d_T(x, y) \geq d(x, y)$ for all $T \in$ support$(\mathcal{D})$.*

Proof: Fix x and y, and let i be such that $d(x, y) \in (2^{i-1}, 2^i]$. Consider the invocation of **TreeEmbed**(U, i) such that $x \in U$. First, we examine the case in which $y \in U$. By the definition of the low diameter decomposition, since $d(x, y) > 2^{i-1}$, x and y will fall into separate parts of the partition obtained in Step 1, and so we will have $d_T(x, y) \geq 2^i$, the length of the edge placed between different subtrees. In the case in which $y \notin U$, then it must be that x and y have been separated at a higher level i' of the recursion, are consequently separated by a higher subtree edge, and hence $d_T(x, y) \geq 2^{i'} > 2^i$. $\square$

Lemma 3 *For all $x, y \in V$, $E_{T \sim \mathcal{D}}[d_T(x, y)] \leq d(x, y) \cdot O(\log \Delta \, \log n)$*

Proof: We begin with two easy subclaims. Suppose $(T_U, root) \leftarrow$ **TreeEmbed**(U, i):

1. Claim 1: $d_{T_U}(root, x) \leq 2^{i+1}$ for all $x \in U$. By induction, x lies in some piece S_j of the partition having diameter at most 2^{i-1} and hence inductively is at distance at most 2^{i-1+1} from its root $root_j$. That root is connected to the root $root$ by an intertree edge of weight 2^i, giving us 2^{i+1} in total.
2. Claim 2: If $x, y \in U$, then $d_{T_U}(x, y) \leq 2 \cdot 2^{i+1}$. From the previous claim, each x and y is at distance at most 2^{i+1} from $root$, distances are symmetric, and the triangle inequality applies.

We now have from the definition:

$$\begin{aligned} d_T(x,y) &\leq \sum_{i=\delta}^{0} \Pr[(x,y) \text{ first separated at level i}] \cdot 4 \cdot 2^i \\ &\leq \sum_{i=\delta}^{0} \beta \, \frac{d(x,y)}{2^{i-1}} \cdot 2^i \cdot 4 \\ &= (\delta + 1)\, 8\beta \, d(x,y) \end{aligned}$$

where the first inequality follows from our subclaims, the second follows from the property of the low diameter decomposition. Setting $\beta = O(\log n)$ and $\delta = O(\log \Delta)$ completes the proof. $\square$

The two lemmas above prove Theorem 1. How do we implement these low diameter decompositions? And how can we get the promised $O(\log n)$? Keep reading...

3 Low Diameter Decompositions

Recall the definition of a *(randomized) low-diameter decomposition* from above: given a metric (V, d) and a bound r, we want a partition with pieces of radius at most $r/2$, and want vertices to be separated with "small" probability $\beta \frac{d(x,y)}{r}$ (i.e., proportional to their distance, and inversely proportional to r).

Before we proceed, think about how you'd get such a decomposition for a line metric, or a tree metric, with $\beta = O(1)$; moreover, you cannot hope to get subconstant $\beta = o(1)$ for even the line. So the theorem says that general graphs lose a factor $O(\log n)$ more, which is not bad at all! (And this factor is existentially optimal, we will show a tight example.)

3.1 Algorithm I: Geometric Region Growing

To make our life easier, we'll assume that all distances in the metric are at least r/n^2. (We can enforce this via a pre-processing without much effort, I'll come back to it.)

The algorithm just picks a "truncated" geometric distance R, carves out a piece of radius R around some vertex, and repeats until the metric is eaten up.

```
**Geom-Regions**$(V, d, r)$:
    1. Choose $R \sim \texttt{Geom}(\frac{4 \ln n}{r})$; if $R > r/2$, then set $R \leftarrow r/2$.
    2. Pick an arbitrary vertex $u \in V$, and set $S_1 \leftarrow \{v \in V \mid d(u,v) \leq R\}$.
    3. Return $\{S_1\} \cup$ **Geom-Regions**$(V \setminus S_1, d, r)$.
```

Clearly, the radius bound is maintained by the fact that $X \leq r/2$ with probability 1.

What's the chance that x, y lie in separate parts? So let's view this process as picking a vertex u and starting with a ball of radius zero around it; then we flip a coin with bias $p = r/(4 \ln n)$, increasing the radius by one after each tails, until either we see a heads or we reach $r/2$ tails, when we cut out the piece. And then we pick another vertex, and repeat the process.

Consider the first time when one of these lies in the current ball. Note that either this ball will eventually contain both of them, or will separate them. And to separate them, it must make a cut within the next $d(x,y)$ steps. The chance of this is at most the chance of seeing a heads from a bias-p coin in $d(x,y)$ steps, *plus* the chance that a $\texttt{Geom}(p)$ r.v. sees more than $(2 \ln n)/p$ tails in a row. Using a naive union bound for the former, we get

$$d(x,y) \cdot p + (1-p)^{(2 \ln n)/p} \leq \frac{d(x,y)}{r} \cdot 2 \ln n + \frac{1}{n^2}.$$

We now use the fact that all distances are at least r/n^2 to claim that $1/n^2 \leq \frac{d(x,y)}{r}$ and hence the probability of x, y separated is at most $(4 \ln n + 1) d(x,y)/r$, which proves the second property of the decomposition.

Finally, the loose ends: to enforce the minimum-distance condition that $d(x,y) \geq r/n^2$, just think of the metric as a complete graph with edge-lengths $\ell_{xy} = d(x,y)$, contract all edges (x,y) with $d(x,y) < r/n^2$, and recompute edge lengths to get the new metric $d' \leq d$. Running the decomposition **Geom-Regions**$(V, d', r/2)$ on this shrunk metric, and then unshrinking the edges, will ensure that each pair is separated with probability either 0 (if it has length $< r/n^2$), or probability at most $(8 \ln n + 2) d(x,y)/r$. And finally, since the output had radius at most $r/4$ according to d', any path has at most n nodes and its length can change by at most $n \cdot r/n^2 \leq r/4$ for $n \geq 4$, the new radius is at most $r/2$!.

Another advantage of this shrinking preprocessing: a pair x, y is separated only when $r \leq d(x,y) \cdot n^2$, and it is separated for sure when $r \geq d(x,y)$. Using this observation in the calculation from the previous section can change the $O(\log n \log \Delta)$ to just $O(\log n \min(\log n, \log \Delta))$. But to get the ultimate $O(\log n)$ guarantee, we'll need a different decomposition procedure.

3.2 Algorithm II: The CKR Decomposition

Theorem 4 (The Better Decomposition) *There exists an efficiently sampleable probability distribution $\mathcal{D}$ over partitions with parts having radius at most $r/2$ such that*

$$\Pr[u, v \textit{ separated by the partition}] \leq \frac{4\, d(u,v)}{r} \log \left(\frac{|Ball(u, r/2)|}{|Ball(u, r/8)|} \right)$$

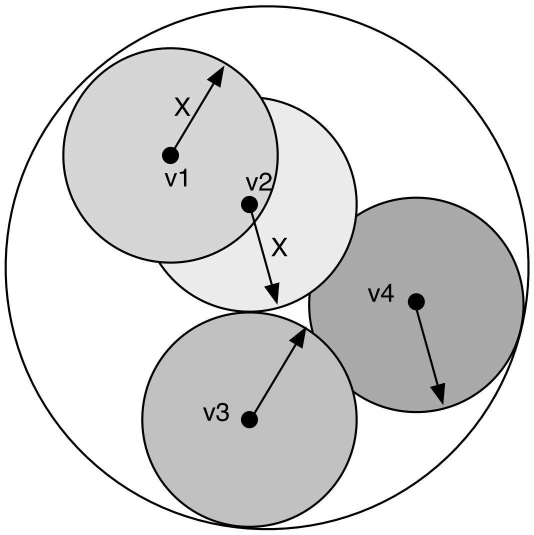
where $Ball(x, r) = \{y : d(x, y) \leq r\}$.

The procedure for the decomposition is a little less intuitive, but very easy to state:

CKR Decomposition(V, d, r):

1. Choose $R \in_R [\frac{r}{4}, \frac{r}{2}]$ uniformly at random.
2. Choose a random permutation $\pi \colon V \to V$ uniformly at random.
3. Consider the vertices one by one, in the order given by π. When we consider w, we assign all the yet-unassigned vertices v with $d(w, v) \leq R$ to w's partition.

For example, suppose the ordering given by π is v_1, v_2, v_3, v_4. The figure below illustrates the coverage when the vertices are visited by this process.



This construction directly implies the low-radius property, restated in the following claim.

Lemma 5 (Low Radius) *The output of the algorithm has the property that for all S_i, there exists $c_i \in V$ such that for all $u \in S_i$, $d(c_i, u) \leq r/2$.*

The real work is in showing that for each pair (u, v), it is separated with small probability. Before proving this, let us state two definitions useful for the proof. For the analysis only: suppose we re-number the vertices $w_1, w_2, \ldots, w_n$ in order of the distance from the closer of u, v.

- **(Settling)** At some time instant in this procedure, one (or both) of u or v gets assigned to some w_i. We say that w_i *settles* the pair (u, v).
- **(Cutting)** At the moment the pair is settled, if only one vertex of this pair is assigned, then we say that w_i *cuts* the pair (u, v).

According to these definitions, each pair is settled at exactly one time instant in the procedure, and it may or may not be cut at that time. Of course, once the pair is settled (with or without being cut), it is never cut in the future.

Now to bound the separation probability. Consider w_j, and let $d(w_j, u) = a_j$ and $d(w_j, v) = b_j$. Assume $a_j < b_j$ (the other case is identical). If w_j cuts (u, v) when the random values are R and π, the following two properties must hold:

1. The random variable R must lie in the interval $[a_j, b_j]$ (else either none or both endpoints of e would get marked).
2. The node w_j must come before $w_1, \ldots, w_{j-1}$ in the permutation π.

Suppose not, and one of them came before w_j in the permutation. Since all these vertices are closer to the pair than w_j is, then for the current value of R, they would have settled the pair (either capturing one or both of the endpoints) at some previous time point, and hence w_j would not settle—and hence not cut—the pair (u, v).

With these two properties, we establish

$$\begin{aligned}\Pr[\text{pair } (u,v) \text{ is separated}] &= \sum_j \Pr[w_j \text{ cuts the pair } (u,v)] \\ &\le \sum_j \Pr[R \in [a_j, b_j] \text{ and } w_j \text{ comes before } w_1, \ldots, w_{j-1} \text{ in } \pi] \\ &\le \sum_j \frac{d(u,v)}{(r/2 - r/4)} \cdot \frac{1}{j} \le \frac{4\, d(u,v)}{r} \cdot H_n = \frac{d(u,v)}{r}\, O(\log n)\end{aligned}$$

But we wanted to do better than that! No worries, the fix is easy, but clever. First, note that if $d(u, v) \ge r/8$ then the probability of separating u, v is at most $1 \le 8\, d(u, v)/r$. So suppose $d(u, v) < r/8$. Now, for w_j to cut (u, v), it is not enough for $R \in [a_j, b_j]$ and w_j comes before all w_i for $i < j$. It also must be the case that w_j be at most $r/2$ from the closer of the pair (say u) to even reach one of the vertices, let alone separate then. And at least $r/4$ from the further one (say v) so that some setting of R would have a chance to separate the two. So the distance of w_j from u must be at most r, and at least $r/4 - d(u,v) \geq r/8$, and the same for its distance from v. If we restrict the harmonic sum in the final expression over just the vertices that satisfy these bounds, we get the bound

$$\frac{d(u,v)}{r/4}\left(\frac{1}{|B(u,r/8)|+1} + \frac{1}{|B(u,r/8)|+2} + \cdots + \frac{1}{|B(u,r)|}\right),$$

and hence the bound in Theorem 4.

Theorem 6 (FRT 2003) *Using the decomposition procedure from Theorem 4 in the* ***TreeEmbed*** *algorithm, we get that for all $x, y \in V$:*

$$E_T[d_T(x,y)] \leq d(x,y) \cdot O(\log n)$$

The proof for the **TreeEmbed** algorithm remains essentially unchanged, except for the final calculations:

$$\begin{aligned}
E_T[d_T(x,y)] &\leq \sum_{i=\delta}^{0} \Pr[(x,y) \text{ separated at level i}] \cdot 4 \cdot 2^i \\
&\leq \sum_{i=\delta}^{0} \frac{4\,d(x,y)}{2^{i-1}} \cdot \log\left(\frac{|B(x,2^i)|}{|B(x,2^{i-3})|}\right) \cdot 2^i \cdot 4 \\
&= 32\,d(x,y) \sum_{i=0}^{\delta} (\log(|B(x,2^i)|) - \log(|B(x,2^{i-3})|)) \\
&= 32\,d(x,y)(\log(|B(x,2^\delta)|) + \log(|B(x,2^{\delta-1})|) + \log(|B(x,2^{\delta-2})|) \\
&\leq 96\,d(x,y)\log n
\end{aligned}$$

where the last equality follows from observing that we have a telescoping sum.

Citations: The $O(\min(\log\Delta, \log n)\log n)$ construction was due to Yair Bartal (1996); this substantially improved on the first non-trivial guarantee of $\exp(\sqrt{\log n \log\log n})$ due to Alon, Karp, Peleg and West (1992). The low-diameter decomposition is also from Bartal. The $O(\log n)$ algorithm is by Fakcharoenphol, Rao, and Talwar (2003), based on the improved decomposition scheme due to Calinescu, Karloff and Rabani (2000).

4 Lower Bounds

Let us show two lower bounds: first, that no randomized low-diameter decomposition can achieve better than $\beta = O(\log n)$ for general metrics. And that no random tree embeddings can do better than $\alpha = O(\log n)$ either.

4.1 Lower Bounds for Decompositions

First, given a graph $G = (V, E)$ with unit length edges, if we apply a β decomposition with parameter r to the graph metric d_G, we will cut each edge with probability β/r. The expected number of cut edges will be $\beta m/r$. So, for each r the probabilistic method says there exists a diameter-r partition that cuts at most $\beta m/r$ edges.

Let G be a graph with n nodes and cn edges (with $c > 1$), where the girth of the graph (the length of the shortest simple cycle) is at least $g = c' \log n$ (for constant $c' < 1$). Such graphs are known to exist, this can be shown by the probabilistic method.

Now, if we set $r = \frac{c'}{3} \log n = g/3$ and consider any diameter-r partition: we claim no set S in this partition can induce a cycle. Indeed, since every cycle is of length g, two furthest points in the cycle would be $g/2 = \frac{c'}{2} \log n > r$ distance from each other. So all sets induce a forest, which means the number of internal edges is at most $n - 1 < n$. This means at least $(c-1)n$ edges are cut.

Cool. For every diameter-r partition, at least $(c-1)n$ edges are cut because of the large girth property. But there exists one that cuts at most $\beta\, m/r = \beta\, cn/r$ edges, because we have a good decomposition algorithm. So now we put the two facts together.

$$(c-1)n \leq \beta \frac{cn}{r} \implies \beta \geq \frac{c-1}{c} r = \Omega(\log n).$$

4.2 Lower Bounds for Random Tree Embeddings

Suppose there is a distribution $\mathcal{D}$ that achieves expected stretch α for the large-girth graphs above. Let's use this to obtain a low-diameter decomposition with cutting parameter $\beta = O(\alpha)$; this will mean $\alpha = \Omega(\beta) = \Omega(\log n)$.

Sample a tree T from the distribution, pick an arbitrary vertex v, pick a random value $R \in_R [0, r/2)$. Delete all edges that contain points at distance exactly in $\{R, r/2 + R, r + R, 3r/2 + R, \ldots\}$ from v. The remaining forest has components with radius at most $r/2$, and diameter r in the tree. Since distances on the original graph are only smaller, the diameter of each part will only be less in the original graph.

Moreover, given the tree T, a pair will be separated with probability at most $\frac{d_T(x,y)}{r/2}$. Taking expectations, the total probability of x, y separated is at most

$$E_T[\frac{2\, d_T(x,y)}{r}] \leq 2\alpha\, \frac{d(x,y)}{r}.$$

So we have a decomposition scheme with parameter 2α. And combining this with the previous lower bound on any decomposition scheme, we get $\alpha = \Omega(\log n)$.