1 Notes by Tim on Directly Sparsifying $L_G^\dagger$ via cut metrics and tree metrics

Theorem 1.1. *Effective Resistance embeds isometrically into a distribution over Tree Metrics*

Theorem 1.2. *Effective Resistance embeds isometrically into a distribution over cut metrics*

Theorem 1.3. *There exists a set of $O(n \log n/\varepsilon^2)$ cut metrics, whose weighted sum $(1+\varepsilon)$ approximates effective resistance.*

The first two theorems are equivalent to Effective Resistance being located inside L_1, which is known in the literature.

The third theorem is also implied by literature, since L_1 metrics can all be sparsified by $O(n/\varepsilon^2)$ cut metrics. This is not obvious, and was a paper in 2010.

In this note, we try to write $L^\dagger$ as a positive linear combination of rank-one matrices associated to cuts (we will call these rank-one matrices **cut matrices**, to be defined later). This is how we will prove Theorems 1.1 and 1.2. Running matrix chernoff on these rank-one matrices will lead us to Theorem 1.2.

I do not claim credit for the proof of Theorem 1.1 or Theorem 1.2, as they were known in the literature before.

We also prove, using rather naive methods, that

Conjecture 1.3.1. *There exists a set of $O(\log n/\varepsilon^4)$ tree metrics, whose sum $(1+\varepsilon)$ approximates effective resistance.*

This shows that Effective Resistance can be written as a small number of tree metrics, a result that was previously not known. However, this is NOT TRUE for cycles, and I have no reason to believe it will be true for any modification of this process! :D

We now turn our attention to a few open-ended conjectures:

Conjecture 1.3.2. *There exists a set of $O(\log n/\varepsilon^2)$ tree metrics, whose sum $(1+\varepsilon)$ approximates effective resistance.*

Conjecture 1.3.3. *There exists a set of $O(n \log n/\varepsilon)$ cut metrics, whose sum $(1+\varepsilon)$ approximates effective resistance.*

Conjecture 1.3.4. *There exists a set of $O(\log n/\varepsilon)$ tree metrics, whose sum $(1+\varepsilon)$ approximates effective resistance.*

Conjecture 1.3.2 would show that effective resistance has a structural result that it is approximately a small number of tree metrics.

It would be nice if I had fast algorithms to do any of these, but I don't.

2 Refinement of Theorem 1.1: Distribution of Tree Metrics that Average to Effective-Resitance

In this section, we explicitly find the distribution over trees such that the expected value of $L_T^\dagger$ is $L_G^\dagger$. This directly implies a distribution of trees whose expected value is effective resistance.

Algorithm 1 TreeGenerator(G)

---

1. Sample a uniformly random spanning tree in G.
2. For each edge of the tree, weight it with the number of edges crossing the cut induced by the tree. (That is, each edge in the tree induces a cut: count the number of edges in G that cross the cut. Weight each edge with that number)

Theorem 2.1. TreeGenerator(G) *generates a distribution on trees T such that the expected value of $L_T^\dagger$ is $L_G^\dagger$.*

We only show that $\boldsymbol{\chi_{uv}}^T \mathbb{E}\left[L_T^\dagger\right] \boldsymbol{\chi_{uv}} = \boldsymbol{\chi_{uv}}^T L_G^\dagger \boldsymbol{\chi_{uv}}$, where $\boldsymbol{\chi_{uv}}$ is the vector with 1 at vertex u, -1 at v, and 0 elsewhere. We claim without proof that if this holds for all $u, v \in V(G)$, and the nullspace of $\mathbb{E}\left[L_T^\dagger\right]$ and $L_G^\dagger$ are th same, then $\mathbb{E}_{L_T^\dagger}[=] L_G^\dagger$. That is, we claim without proof that it suffices to prove Lemma 2.2.

Lemma 2.2. TreeGenerator(G) *generates a distribution on trees T such that the expected value of $\boldsymbol{\chi_{uv}}^T \mathbb{E}_{L_T^\dagger}[\boldsymbol{\chi_{uv}}]$ is $\boldsymbol{\chi_{uv}}^T L_G^\dagger \boldsymbol{\chi_{uv}}$.*

Proof. (of Lemma 2.2). We split into two cases:

1. Edge uv is in G.
2. Edge uv is not in G.

We first deal with Case 1.

We WILDLY ASSUME that TreeGenerator gives the same distribution as CutGenerator. Now we just show CutGenerator gives the right thing.

Split the set (Tree, edge) into equivalence classes, where: two (Tree, edge) objects are equivalent if and only if removing the edge results in the same two-tree forest. Call this equivalence class a megatree.

Now, for a given megatree, it is chosen with probability $cutsize/total - tree - count$ and is weighted $1/cutsize$. So overall, it contributes a weight of $1/total - tree - count$ per megatree.

Note that for every tree T that contains uv, there is a one to one correspondence to a megatree where T is one of the representatives of the megatree.

The expected distance from u to v in the megatree is exactly equal to the number of representatives T containing uv, which is exactly equal to the effective resistance (assuming a unit weight graph, possibly with multi-edges).

Now: The expected distance from u to v is therefore equal to exactly the probability a tree contains uv, which is exactly equal to the Effective Resistance. Note this holds even if u and v has multi-edges between them, and does NOT give leverage score.

I don't know how to prove Case 2. □

Proof. (of Theorem 2.1) Follows (without proof) from Lemma 2.2). □

Proof. (of Theorem 1.3.1 THERE IS A MISTAKE HERE AND THE THEOREM IS NOT TRUE (Breaks for cycles). We first take a graph with high leverage scores all around. Then the megatree is chosen with high probability (at least proportional to leverage score). A graph with high leverage score that spectrally approximates the original can be obtained via Spielman Srivastava.

Now, each megatree containing uv is sampled with probability equal to at least the leverage score of uv. (It is sampled with probability equal to cut-size times effective resistance of uv, and cut-size is at least the conductance from u to v).

Thus the probability any megatree is sampled is at least $> O(\log n/\varepsilon^2)$, as that is a uniform bound on leverage score in Spielman-Srivastava. This bound is VERY weak.

The weight that each megatree contributes to the distance from u to v is less than $1/cutsize$, which is less than $1/c_e$.

Therefore the weight on uv is a average of a bunch of random variables, each of which is less than $1/c_e$, and averages to ER. (Now assume unit weight graph). Concentration is the same as if a bunch of random variables were each less than 1, and averaged to leverage-score. Likewise, if a bunch of variables were weighted inverse lev score and averaged to 1.

How many samples would we need from here? Chernoff bound tells us that for each uv, we would need something like: $\varepsilon^2 s/L = \log n$ where L is the maximum value of the sample (assuming a 1 average), and s is the number of samples taken. Then $s = L \log n/eps^2$, which equals $\log^2 n/eps^4$.

This is a real shitty bound and I think I might have made a bunch of mistakes getting up to this point. Now you take the union bound over all pairs of points.

□

Corollary 2.2.1. TreeGenerator(G) *generates a distribution on tree metrics whose expected value is the effective resistance.*

3 Refinement of Theorem 1.2: Distribution of Cut Metrics whose Average is Effective Resistance

In this section, we find a distribution over cuts such that the expected value of such cuts is equal to the effective resistance.

Algorithm 2 CutGenerator(G)

1. Sample a uniformly random spanning tree in G.
2. Pick an edge uniformly at random from the tree. This will induce a cut.
3. Weight the cut with the inverse of the number of edges crossing the cut, times $(n-1)$.

Theorem 3.1. CutGenerator(G) *generates a distribution on cuts such that the expected value of these cuts is equivalent to effective resistance.*

This follows directly from Theorem 2.1, by splitting each tree metric into a cut metric. However, we'd like a matrix version of Theorem 3.1, so we can apply Matrix Chernoff to it and subsample a small number of cuts.

To do this, we want to find a rank one matrix associated to each cut.

Definition 3.2. *Let G have n vertices. For each cut C that divides the graph into vertex set S and T, define a vector $u_C \in \mathbb{R}^n$ such that*

$$u_C(v) = |T|/n$$

if $v \in S$, and

$$u_C(v) = -|S|/n$$

if $v \in T$.

*For cut C, define its **cut matrix** M_C to equal $u_C u_C^T$. Note that M_C is orthogonal to the all-ones vector, and has rank one.*

Lemma 3.3. *Let χ_{ij} equal the vector with 1 at i and -1 at j and 0 everywhere else. Then $\chi_{ij}^T M_C \chi_{ij}$ is equal to the distance between i and j in the cut metric.*

We state the above lemma without proof, and I believe it is a straightforward verification.

Theorem 3.4. CutGenerator(G) *generates a distribution on cuts C such that the expected value of M_C is $L_G^\dagger$.*

This follows directly from theorem 2.1. Here, we need to check that the sum of matrices M_C, where C ranges over all cuts induced by removing an edge from T, is equal to $L_T^\dagger$. This exercise is left to the reader :).

4 Matrix Chernoff for M_C, and 'Proof' of Theorem 1.3

Here, we run Matrix Chernoff on cut matrices M_C. I'll just state a couple of results without proof.

Recall that in Spielman Srivastava, it is proven that if we have a distribution of matrices weighted by the leverage score, which is conductance times effective resistance in the case of

sparsifying L_G, and we weight each matrix L_e by $(n-1)$ times the inverse effective resistance, then we get a distribution over L_e, such that sampling and scaling $n \log n/\varepsilon^2$ samples from this distribution gives a $(1+\varepsilon)$ approximation of L_G.

Now, we show the analog of this process when we write $L_G^\dagger = \sum_C a_C M_C$. Here, a_C turns out to be the number of two-tree spanning forests with no edge crossing cut C, divided by the total number of spanning trees of G. This is a known result from folklore, or can alternately be derived from Theorem 3.1.

4.1 Analog of c_e

Recall $\sum_e c_e L_e = L_G$, and analagously $\sum_C a_C M_C = L_G^\dagger$. Thus, we can loosely say that a_C is the analog of c_e.

4.2 Analog of Effective Resistance

Recall that Effective Resistance in Spielman-Srivastava equals:

$$\text{Trace}(L_G^{\dagger/2} L_e L_G^{\dagger/2})$$

The analog of this for cut matrices, therefore, would be:

$$\begin{aligned}
&\text{Trace}(L_G^{1/2} M_C L_G^{1/2}) \\
&\text{Trace}(L_G^{1/2} u_C u_C^T L_G^{1/2}) \\
&= \text{Trace}(u_C^T L_G u_C) \\
&= u_C^T L_G u_C,
\end{aligned}$$

where u_C was defined in Definition 3.2. Here, it is not too difficult to show that this quantity equals the number of edges in G crossing cut C.

4.3 Analog of Leverage Score

The analog of conductance times effective resistance, is $a_C \cdot u_C^T L_G u_C$, which turns out to equal the number of spanning trees with exactly one edge crossing cut C.

I haven't rigorously worked through the analogies above, and this is a crucial step I haven't done yet.

Theorem 4.1. *Sampling and rescaling $n \log n/\varepsilon^2$ values of M_C via* CutGenerator(G)*, and adding the results, gives us $L_G^\dagger$ within $(1+\varepsilon)$ approximate factor, with high probability.*

The above theorem follows from our computation of leverage score, Matrix Chernoff, and the way CutGenerator is defined. I have omitted details in the proof, and these details certainly need checking.

This proves Theorem 1.3, and gives a sampling method for generating an empirical sample of cut metrics whose average is effective resistance.

5 Open Ended Questions

In CutGenerator, you pick a random spanning tree and sample each edge with $\frac{1}{n-1}$ probability to get a cut. If you just took the batch of $(n-1)$ cuts associated with the tree, you would get TreeGenerator. We need $n \log n/\varepsilon^2$ cuts from CutGenerator to approximate $L_G^\dagger$ with high probability. Perhaps we might only need $\log n/\varepsilon^2$ trees from TreeGenerator? This gives us Conjecture 1.3.2. Sadly, I have no idea how to proceed with this! I would believe it is false, but it is unclear how to systematically provide counter-examples.

Incidentally, I believe we can use the above framework to embed any n-item metric in l_1, into $l_1^{n \log n/\varepsilon^2}$ with $(1+\varepsilon)$ distortion. It's possible we can get rid of ε^2 and replace it with ε, kind of like how JS'17 does it.

Note that in general, l_1 isometrically embeds into $l_1^{n(n-1)/2}$, and $n(n-1)/2$ is tight.