CMSC 858F: Algorithmic Lower Bounds: Fun with Hardness Proofs
Fall 2014
Cubic hardness and all-pair shortest paths

Instructor: Mohammad T. Hajiaghayi
Scribe: Karthik Abinav S

September 11, 2014

1 Overview

In the next two lectures, we look at lower bounds conjectured on two important and well-known problems. One is the All-Pairs-Shortest-Path(APSP) problem which is believed to be truly cubic(i.e. there is no exact algorithm for this problem which runs in time $O(n^{3-\epsilon})$ for a constant $\epsilon > 0$). The second problem considered is the $3-SUM$ problem which is conjectured to be truly quadratic(i.e. there is no exact algorithm that runs in time $O(n^{2-\epsilon})$ for a constant $\epsilon > 0$). This lecture will focus on the cubic hardness and the APSP problem.

2 Problem Definition

Definition 1 (All Pair Shortest Path Problem) *Given a directed(or undirected) graph G(V,E) which has n vertices and m edges with the absolute edge weights in the set $\{0, 1, \ldots, M\}$, the problem is to compute the shortest distance between vertex x and y for all pairs of vertices x and y.*

This problem is a very important and central problem in graph theory. There is a well-known $O(n^3)$ algorithm for this problem due to Floyd and Warshall. The algorithm is as follows

Data: A graph G(V,E) given as an adjacency matrix G(i,j)
Result: A resulting matrix dist such that dist(i,j) gives the shortest distance between the vertices i and j
$\forall i, j \in V$, assign dist(i,j) as G(i,j) ;
for *k=1 to n* **do**
  for *i=1 to n* **do**
    for *j =1 to n* **do**
      if *dist(i,j) > dist(i,k) + dist(k,j)* **then**
        dist(i,j) ← dist(i,k) + dist(k,j) ;
      end
    end
  end
end

Algorithm 1: Floyd-Warshall Algorithm

Let n be the number of vertices in the graph. The above algorithm runs in time $O(n^3)$. Another way to compute the all pair shortest path is by invoking the Dijikstra's algorithm for each vertex. Dijikstra's algorithm finds the shortest path from a given vertex v to all other vertices in the graph. Hence, invoking it n times by choosing a different starting vertex v each time, will result in calculating the all pair shortest path. A single invocation of the Dijikstra's algorithm takes $O(n + m \log m)$. Hence, n iterations of this algorithm takes a time of $O(nm + n^2 \log m)$. In the worst case, m can be $O(n^2)$ and hence the worst case running time using this approach is also $O(n^3)$.

3 Related Measures

In this section, we will look at some of the related measures and lower bounds on their running time.

3.1 Radius

The radius R of a graph G(V,E) is a single value which denotes the following value

$$\min_{v} \max_{u} \mathtt{dist}(u, v)$$

Given a vertex v, let α denote the maximum value of the shortest distance to any vertex. Radius of the graph is the smallest possible value for α over all vertices in the graph.

The vertex v which minimizes the value α is called the center of the graph.

$$\text{Center of graph} = \arg\min_{v} \max_{u} \mathtt{dist}(u, v)$$

3.2 Diameter

Another closely related measure is the diameter of the graph. It is the maximum possible distance between any two vertices in the graph. Mathematically,

$$\text{Diameter} = \max_{u,v} \mathtt{dist}(u, v)$$

3.3 Median of the graph

Median of the graph is the minimum value over all the vertices v in the graph, such that the sum of distances to all vertices u in the graph from v is minimized. Formally,

$$\text{Median} = \min_{v} \sum_{u} \mathtt{dist}(v, u)$$

3.4 Betweeness Centrality

We now define another measure called the betweeness centrality. This measure is very closely related to the number of s-t shortest paths, passing through a given vertex v.

For a given vertex v, the betweeness centrality BC(v) is defined as

$$BC(v) = \sum_{s,t \in V \setminus \{v\}; s \neq t} BC_{s,t}(v)$$

where, $BC_{s,t}(v)$ is the fraction of shortest paths between s,t, that uses the vertex v.

Notice that when all the shortest paths are unique, i.e. there exists exactly one shortest path between a pair of vertices s,t, then $BC(v)$ denotes the number of s,t shortest paths that passes through v.

3.5 Positive Betweeness Centrality

Positive betweeness centrality is an indicator function for a vertex v. If there exits a s,t shortest path through v, then the value is 1, else it is 0. Formally,

$$PBC(V) = \begin{cases} 1 & \text{if } BC(v) > 0 \\ 0 & \text{otherwise} \end{cases}$$

3.6 Negative Triangle

Given an undirected graph G with integer edge weights in $\{-M, \ldots, M\}$, the problem is to check if there exists a triangle in the graph, such that the sum of weights on this triangle is negative.

Scribe: Karthik Abinav S
Lecture 4 **Date:** 09/11/2014

4 Sub-cubic Equivalence

The measures Radius, Diameter, Median are all more generally referred to as "Centrality Measures". These measures appear in a variety of applications such as social network, transportation and allocation problems, biological networks, etc. Hence, calculating these measures efficiently has a lot of real life implications.

It is clear that, if we have a sub-cubic algorithm for the APSP problem, we can use that algorithm to calculate the above centrality measures in sub-cubic time. It only takes $O(n^2)$ time, once we have calculated the dist(u,v) for all pairs $u, v \in V$. The question is can we calculate these centrality measures without calculating the APSP. In other words, is there a truly sub-cubic algorithm for calculating radius, diameter and median ? The answer to this was given by Abboud et al [2] in which they claim that there exists a truly sub-cubic algorithm for computing the centrality measures if and only if there exists a truly sub-cubic time algorithm for the APSP problem.

The next natural question is to ask are there algorithms for the APSP problem which does better than the $O(n^3)$ running time of Floyd-Warshall. Fredman[4] gave a $O(\frac{n^3 \sqrt[3]{\log\log n}}{\sqrt[3]{\log n}})$, which is a slight improvement over the Floyd-Warshall running time, in the year 1976. Note that this running time is still not truly sub-cubic, since $\frac{\sqrt[3]{\log n}}{\sqrt[3]{\log\log n}} = o(n^\epsilon)$ for any given constant $\epsilon > 0$. After several years of research and a series of algorithms, Ryan Williams[3] gave an algorithm which ran in time $O(\frac{n^3}{2^{\Omega(\sqrt{\log n})}})$ in the year 2014. This running time, though is better than all previous known algorithms, still doesn't achieve truly sub-cubic time. It still remains an open question as to wheter there exists an algorithm for the APSP problem running in time $O(n^{3-\epsilon})$ for a given constan $\epsilon > 0$.

APSP conjecture: In fact, it is conjectured that the APSP problem is truly cubic, i.e. there doesn't exist an algorithm which runs in time $O(n^{3-\epsilon})$ for a given constant $\epsilon > 0$, such that it computes the exact values of the all pair shortest paths.

4.1 Sub-cubic reductions

A problem A is said to have a sub-cubic reduction to a problem B, if we have an algorithm that is truly sub-cubic for B then using this algorithm as a subroutine we can come up with an algorithm that is truly-subcubic for problem A.

Sub-cubic Equivalence: Problem A and problem B are said to be sub-cubic equivalent, if there is a sub-cubic reduction from problem A to problem B and vice-versa.

Theorem 1 (AGW '15) *Finding Radius, Median and Betweeness Centrality are subcubically equivalent. Hence, devising a truly-subcubic algorithm for one of them, gives a truly-subcubic algorithm for all the problems.*

We will now look at a diagramatic view of the various reductions and explicitly mark the ones which are non-trivial and the ones that are folklore.

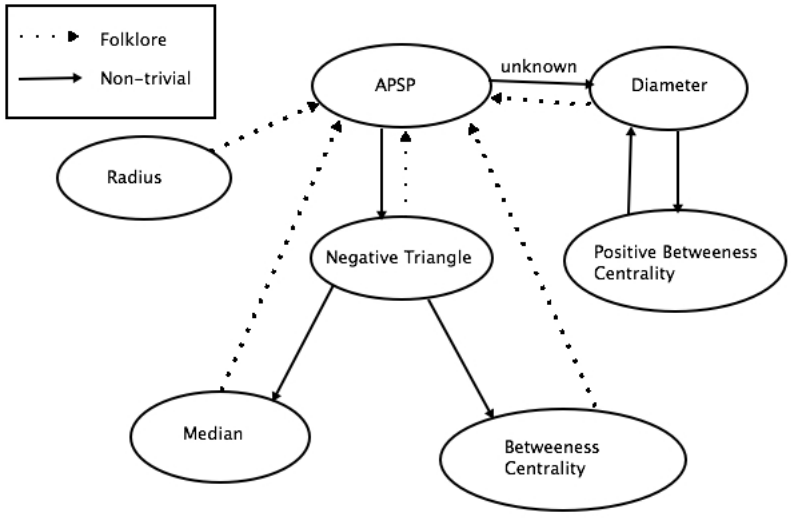

Figure 1: Sub-Cubic Equivalence between various problems

4.2 Reduction from Diameter to Positive Betweeness Centrality

Theorem 2 *Given a T(n,m)-time algorithm for the positive betweeness centrality, we can obtain a* $\tilde{O}(T(n,m))$[1]*- time algorithm for the diameter problem.*

Proof: Without loss of generality, let us assume that all distances and hence, the diameter are even (Multiply all distance by 2 initially, and divide the finally obtained diameter by 2). Now, given a graph G(V,E) as input to the diameter problem, obtain a graph G'(V', E') as follows. V' = V ∪ b i.e. add a new vertex labelled b. E' = E ∪ S, where $S = \{(b, u, \frac{D}{2}) : u \in V\}$. The largest possible

[1] $\tilde{O}$ hides the polylog multiplicative factor

value of D such that the positive centrality of the vertex b is 1, is the required diameter of the original graph G. Hence, performing a binary search on the value of D, constructing a graph G' for every such D and using the algorithm for computing positive betweeness centrality as a sub-routine, will give us the required value of D. ■

Consider the following example of a graph, whose diameter is to be determined.

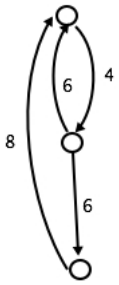

Figure 2: Original graph G

Now, applying the transformation for a given value of D, we obtain G' as shown in the diagram

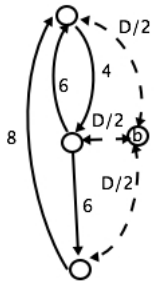

Figure 3: Transformed Graph G'

Now doing a binary search on the values of D, to find the largest value of D such that positive centrality measure of this new vertex b is 1, we get D as 10.

4.3 Reduction from Positive Centrality Measure to Diameter

Theorem 3 *Given a T(n,m)-time algorithm for the diameter problem, we can obtain a* $\tilde{O}(T(n,m))$*-time algorithm for the Positive Betweeness Centrality prob-*

lem.

Proof: Given a procedure to calculate diameter of a graph, we want to use it in sub-cubic time, to get an algorithm to obtain PCM(b) for a given vertex b. To do this, we will modify the input graph G(V,E), to obtain a graph G'(V',E') as follows:

- V' = V $\cup$ { v_a for every $v \in V$ } $\cup$ { v_b for every $v \in V$ }
- E' = E $\cup$ E_1 $\cup$ E_2 $\cup$ E_3 $\cup$ E_4, where E_1, E_2, E_3, E_4 are constructed as follows. E_1 is an edge from every v_a to the corresponding v with weight D' - dist(v,b).
  E_2 is an edge from every v to the corresponding v_b with weight D' - dist(v,b).
  E_3 is an edge from every v to the correponding v_a with weight 0.
  E_4 is an edge from every v_b to the corresponding v with 0.

  Here, D' = 2 $*$ diameter(G). Diagramatically, it looks as follows.

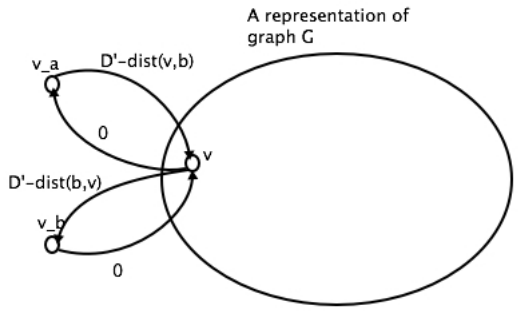

Figure 4: Pictorial representation of construction of graph G'

Note, we need to invoke the Dijikstra's algorithm once to calculate the distance between vertex b and all vertices v in graph G. This runs in sub-cubic time.

Now, select all pairs of vertices (s,t) in the original graph. We can see that for any pair (s,t) in G, the dist(s_a, t_b) in the graph G' $\leq$ 2D'. This is because, D^* = dist(s_a,t_b) = D' - dist(s,b) + D' - dist(b,t) + dist(s,t) = 2D' -(dist(s,b) + dist(b,t) - dist(s,t)) $\leq$ 2D'. The last inequality is due to triangle inequality.

The equality in the triangle inequality holds iff the path whose sum of weights is equal to dist(s,t) passes through the vertex b. In other words, D^* = 2D' iff PBM(b) =1. This completes the reduction.

Observe that the diameter in the new graph is at most five times the original graph G. ■

4.4 Reduction from Negative Triangle to Radius

Theorem 4 *Given a T(n,m)-time algorithm for the radius problem, we can obtain a* $\tilde{O}(T(n, m))$*-time algorithm for the Negative Triangle problem.*

Proof: Given an instance of the negative triangle problem, i.e. a graph G(V,E) with weights in the set $\{-M, \ldots, M\}$ we will construct a graph H(V',E') which is in the form of four layers I,J,K,L. The vertex set V' has four times the number of vertices in V and a special vertex called b. For a vertex $v \in V$, we have four vertices v_I, v_J, v_K, v_L, each present in the corresponding layer. Now, add the following edges in the graph(Let Q = 3M):

- An edge from b to each of v_I in the layer I whose weight is 2Q + M.
- If there exists an edge (u,v) in the original graph, add an edge between u_I and v_J in the new graph H, with weight Q+w(u,v). Similarly, add an edge between u_J and v_K and an edge between u_K and v_L, all with weights Q + w(u,v).
- For every edge (u,v) and $u \neq v$ in the original graph, add an edge between u_I and v_L with weight 2Q in the new graph.

It is important to note that in the above construction there is no edge between any two vertices in the same layer. With the above construction of H, we will make the following claim 1. Also, note that H has $O(n)$ nodes and the maximum edge weight is $O(M)$. The following diagram gives a pictorial representation of the construction of graph H.

Claim 1 *The radius of the graph H is strictly less than 9M if and only if G contains a negative triangle.*

Proof: We will make four observations which will essentially lead to the claim.

- **Observation 1**: If radius R < 9M, then the center of the graph is contained in the layer I. This is easy to see, since for vertices in the other layers J,K,L , the distance to vertex b is at least 3Q(=9M).

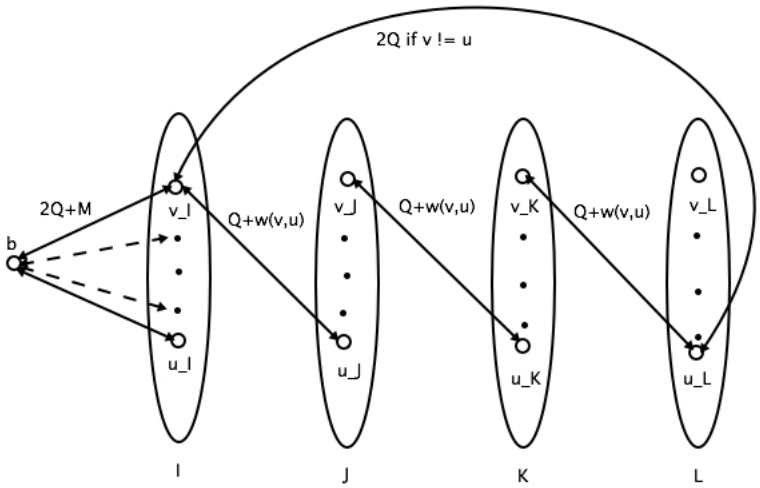

Figure 5: Pictorial representation of construction of graph H

- **Observation 2**: Any vertex v_I in layer I is at a distance of at most 2Q+2M(=8M) from every other vertex in the graph, except for the corresponding vertex v_L in layer L.
- **Observation 3**: If vertex v in graph G was present in a negative triangle, then dist(v_I,v_L) in graph H $<$ 3Q(= 9M). (The 3Q appears for the distance between layer I and layer J + layer J and layer K + layer K and layer L. The strict inequality is because the triangle was a negative triangle in original graph and hence the sum of weights is less than 0)
- **Observation 4**: Finally, if a vertex v is not in a negative triangle in the original graph, then dist(v_I,v_L) $\geq$ min(3Q, 4Q-2M)(=9M).

The claim follows easily from the above four observations. ■

Given this claim, we need to construct the graph H and check if the radius in this new graph is strictly less than 9M. This completes the reduction. ■

As shown by Williams and Williams[7], the problems APSP and Negative Triangle are equivalent under sub-cubic reductions. It is also a folklore that

there exists a sub-cubic reduction from radius to the APSP problem. Hence, the above theorem implies that, there exists a sub-cubic reduction from the Negative triangle problem to the APSP problem.

5 Approximate versions of the problems

In this section, we will breifly touch upon some of the approximation versions of this problem and see the best known running time for those problems.

Theorem 5 (Zwick '98) *We can compute a* $(1+\epsilon)$*-approximation to the APSP problem running in time* $\tilde{O}(\frac{n^\omega}{\epsilon})$*, where* ω *is the exponent in the running time of the fastest known matrix multiplication problem(i.e.* $\omega < 2.3728639$*)*

Before we describe the next theorem, we will define a well known hypothesis known as the Strong Exponential Time Hypothesis (SETH).

Definition 2 (SETH) *The hypothesis claims that, unless* $\mathbb{P} = \mathbb{NP}$*, the general version of the satisfiability problem (SAT) cannot be solved in time* $O(2^{\delta n})$ *time for a given constant* $\delta < 1$*.*

The next theorem gives a running time bound on computing the approximate value of the diameter and the betweeness centrality measure.

Theorem 6 (RV '13) *If there exists an algorithm, which computes a 1.5-approximation of the diameter and the betweeness centrality measure in time* $O(m^{2-\epsilon})$ *for a given constant* $\epsilon > 0$*, then SETH would be false.*

6 Open Problems

Some of the open problems which are closely related to these topics are as follows:

- Is the diameter problem equivalent to the APSP problem under sub-cubic reductions ?
- Is there a theorem for approximating the radius and median problem, similar to the one given by RV'13 ?
- Lastly, the big open problem is does there exist a truly sub-cubic time algorithm to the APSP problem?

References

[1] Abboud, Amir, Vassilevska Williams, Virginia. *Popular conjectures imply strong lower bounds for dynamic problems.* 16 September 2014. FOCS 2014

[2] Abboud, Amir, Grandoni, Fabrizio, Vassilevska Williams, Virginia. *Subcubic Equivalences Between Graph Centrality Problems, APSP and Diameter.* SODA 2015

[3] Williams, Ryan. *Faster all-pairs shortest paths via circuit complexity.* STOC 2014.

[4] Fredman, Michael. *New Bounds on the Complexity of the Shortest Path Problem.* SIAM J. Comput., 5(1), 8389. (7 pages). 1976

[5] Zwick, Uri. *All Pairs Shortest Paths in weighted directed graphs - exact and almost exact algorithms.* Proc. of 39th FOCS (1998), 310-319.

[6] Roditty, Liam, Vassilevska Williams, Virginia. *Fast approximation algorithms for the diameter and radius of sparse graphs.* STOC 2013: 515-524

[7] Williams, Ryan, Vassilevska Williams, Virginia. *Subcubic Equivalences between Path, Matrix and Triangle Problems.* FOCS 2010: 645-654