A Distributed Greedy Heuristic for Computing Voronoi Tessellations With Applications Towards Peer-to-Peer Networks

Brendan Benshoof Andrew Rosen
Anu G. Bourgeois Robert W. Harrison
Department of Computer Science, Georgia State University

May 28, 2015

Outline

Motivation

We were creating a distributed network based off of Delaunay Triangulation and needed a fast distributed algorithm for calculating Delaunay peers. We ran into a couple of issues.

- Distributed algorithms for solving Delaunay triangulation don't really exist (every node does a global solution).
- Not any fast solutions if we start moving out of 2D Euclidean space.

This leaves approximation. However:

- Simple approximation doesn't guarantee a fully-reachable network (k-nearest neighbor, nodes in radius r).
- Other solutions [1] required prohibitively high sampling or other hidden time costs.

And nothing can handle moving nodes.

Voronoi Tessellation and Delaunay Triangulation

- Voronoi Tessellation
  - Partition space such that each point is mapped the 'voronoi generator' to which it is closest.
- Delaunay Triangulation
  - generate a triangulation, such that, for every triangle the circumcircle does not contain any other points

Voronoi Example

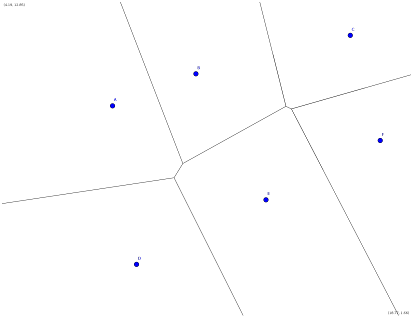

Figure: The space is partitioned, such that each point in a partition is closest to that partitions 'Voronoi generator'

Delaunay Example

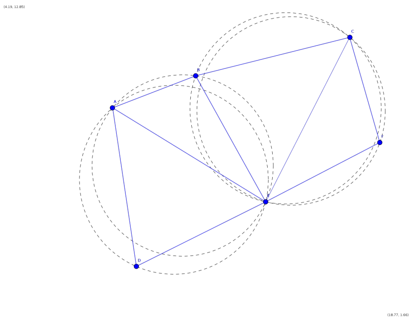

Figure: The space is partitioned, such that each point in a partition is closest to that partitions 'Voronoi generator'

Distributed Hash Tables

- Abstractly, a DHT is a mechanism for maintaining a large state in a decentralized network.
- In practice, the state is a large number of $(key, value)$ records.
- A Distributed hash table assigns those records to servers and routes request for those records to those servers.
- Current incarnations of Distributed hash tables assign servers and records locations in an arbitrary metric space.
- Servers are assigned responsibility for records that are "close" to them, and to peer with "nearby" servers.
- DHTs currently use a variety metric spaces.

Applications of DHTs

- *P2P file sharing* is by far the most prominent use of DHTs. The most well-known application is BitTorrent [4].
- *Distributed Domain Name Systems* (DNS) have been built upon DHTs [5] [7]. Distributed DNSs are much more robust that DNS to orchestrated attacks, but otherwise require more overhead.
- Distributed *machine learning* [6].
- Many *botnets* are now P2P based and built using well established DHTs [8]. This is because the decentralized nature of P2P systems means there's no single vulnerable location in the botnet.

Extant Varieties of DHT

- Ring Based DHTs
  - Chord
  - Pastry
  - Tapestry
- Tree Based DHTs
  - CAN
  - Kademlia

How are DHTs and Vonroi Tesselation/Delunay Trianguation related?

- A Server is responsible for records "closest" to it (Voronoi Tessellation).
- A Server's peers define the topology and act as the links connecting the network (Delunay Triangulation).
- All DHTs can be mapped to a Voronoi Tessilation/Delaunay Triangulation approximation in a given space.

- Ring Based DHTs
  - Chord: a unidirectional modulus ring metric space
  - Pastry, Symphony: bidirectional modulus ring metric space
- Tree Based DHTs
  - CAN: Euclidean metric space
  - Kademlia: XOR distance metric space

Why do we need a distributed Voronoi heuristic?

- The different topologies DHTs utilize present optimization trade-offs (lookup latency, number of lookup hops, network robustness, availability, processing overhead)
- The primary effort in implementing a new metric space in a DHT is implementing Voronoi Regions/Delunay Triangulation algorithm in that metric.
- DGVH allows many metrics to be tested without requiring the design and development effort of generating an exact Voronoi Regions/Delunay Triangulation algorithm.

So we created a Distributed Greedy Voronoi Heuristic, or DGVH for short.

Distributed Greedy Voronoi Heuristic

- Geometrically intuitive method of approximating the one-hop delaunay peers of a Node
- Is guaranteed to form a connected mesh (unlike k-nearest heuristic)
- Can be utilized in any continuous metric space

DGVH Intuition

- The majority of Delunay links cross the corresponding Voronoi edges.
- We can test if the midpoint between two potentially connecting nodes is on the edge of the voronoi region.
- This intuition fails if the midpoint between two nodes does not fall on their voronoi edge.

DGVH Algorithim

```
Given node n and its list of candidates.
peers ← empty set that will contain n's one-hop peers
Sort candidates in ascending order by each node's distance to n
Remove the first member of candidates and add it to peers
for all c in candidates do
    m is the midpoint between n and c
    if Any node in peers is closer to m than n then
        Reject c as a peer
    else
        Remove c from candidates
        Add c to peers
    end if
end for
```

DGVH Example

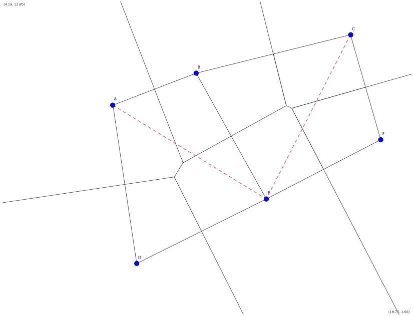

Figure: 'DGVH connects all nodes where the midpoint falls on the border of the voronoi regions'

Realistic Candidate set Size

- In application, a single node will not need to calculate the triangulation for every peer in the network, rather it will only calculate it's own peers.
- A smart "join" process will prevent the Candidate Set from reaching $O(n)$
- Expected size of the candidate set is $O(degree^2)$ which compromises current peers and 2-hop peers
- The average degree of many metric spaces is $O(1)$, then therefore DGVH will practically run in $O(1)$ time in those metric spaces.

Error Mitigation

- As DGVH is a heuristic, it will likely have errors when compared to a global Delaunay triangulation.
- This error rate is difficult to find analytically and will vary between metric spaces and distributions of locations
- A simple method of mitigating this error is to keep both one and two hop peers.

Experiment 1

In our first experiment, we wanted to test the accuracy of our heuristic.

- We generated a random graph and created the Delaunay triangulation using a global solution and DGVH.
- We found DGVH had approximately one error per node.

Our second set of experiments demonstrate that this error rate is sufficiently low for distributed applications.

Results

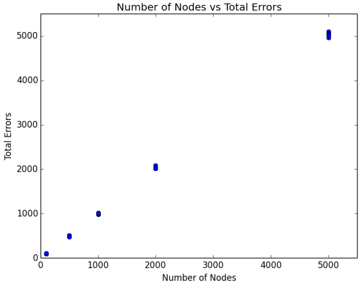

Figure: As the size of the graph increases, we see approximately 1 error per node. We can also see that the error rate and number of nodes has a linear relationship.

Experiment 2

Our second experiment were designed to test how well nodes could form a routing topology using DGVH.

- For each trial, we create a random graph.
- During the first two cycles, nodes are given 10 random connections.
- In each cycle (including the first two):
  - Each node gossips and runs DGVH.
  - 2000 random lookups from random nodes to random locations performed.
- The rate of successful lookups approaches 1.0 as time progresses.

Results

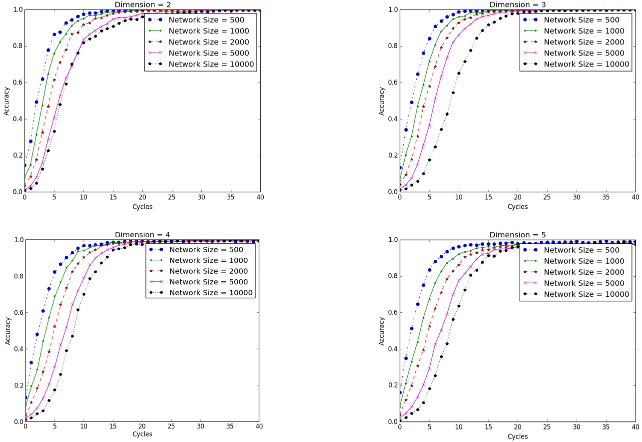

Figure: These figures show, starting from a randomized network, DGVH forms a stable and consistent network topology. The Y axis is the percentage of successful lookups out of 2000 queries and the X axis is the number of gossips cycles.

Other Applications

One type of distributed system that can use Voronoi tessellations are *wireless ad-hoc networks*.

- Solves the coverage-boundary problem [2].
- Can be used for sleep scheduling [3]

Conclusions

- DGVH provides a simple approximation Voronoi Triangulation / Delaunay Triangulation, of similar complexity to picking k-nearest node or nodes in distance k.
- Unlike other simple approximation methods, DGVH guarantees a fully connected graph.
- In practice, DGVH supplemented with K-nearest peers approximation provides an accurate, fast and distributable computation, with a fully connected graph as result.

Olivier Beaumont, A-M Kermarrec, Loris Marchal, and Etienne Rivière.
Voronet: A scalable object network based on voronoi tessellations.
In *Parallel and Distributed Processing Symposium, 2007. IPDPS 2007. IEEE International*, pages 1–10. IEEE, 2007.

Bogdan Carbunar, Ananth Grama, and Jan Vitek.
Distributed and dynamic voronoi overlays for coverage detection and distributed hash tables in ad-hoc networks.
In *Parallel and Distributed Systems, 2004. ICPADS 2004. Proceedings. Tenth International Conference on*, pages 549–556. IEEE, 2004.

Xinyu Chen, Michael R Lyu, and Ping Guo.
Voronoi-based sleeping configuration in wireless sensor networks with location error.
In *Networking, Sensing and Control, 2008. ICNSC 2008. IEEE International Conference on*, pages 1459–1464. IEEE, 2008.

Bram Cohen.
The bittorrent protocol specification, 2008.

Russ Cox, Athicha Muthitacharoen, and Robert T Morris.
Serving dns using a peer-to-peer lookup service.
In *Peer-to-Peer Systems*, pages 155–165. Springer, 2002.

- Mu Li, Li Zhou, Zichao Yang, Aaron Li, Fei Xia, David G Andersen, and Alexander Smola.
  Parameter server for distributed machine learning.
- Vasileios Pappas, Daniel Massey, Andreas Terzis, and Lixia Zhang.
  A comparative study of the dns design with dht-based alternatives.
  In *INFOCOM*, volume 6, pages 1–13, 2006.
- Sherif Saad, Issa Traore, Ali Ghorbani, Bassam Sayed, David Zhao, Wei Lu, John Felix, and Payman Hakimian.
  Detecting p2p botnets through network behavior analysis and machine learning.
  In *Privacy, Security and Trust (PST), 2011 Ninth Annual International Conference on*, pages 174–180. IEEE, 2011.