Building GNN and Transformer Inputs

Andy Chappell

25/03/2024

FD Sim/Reco Meeting

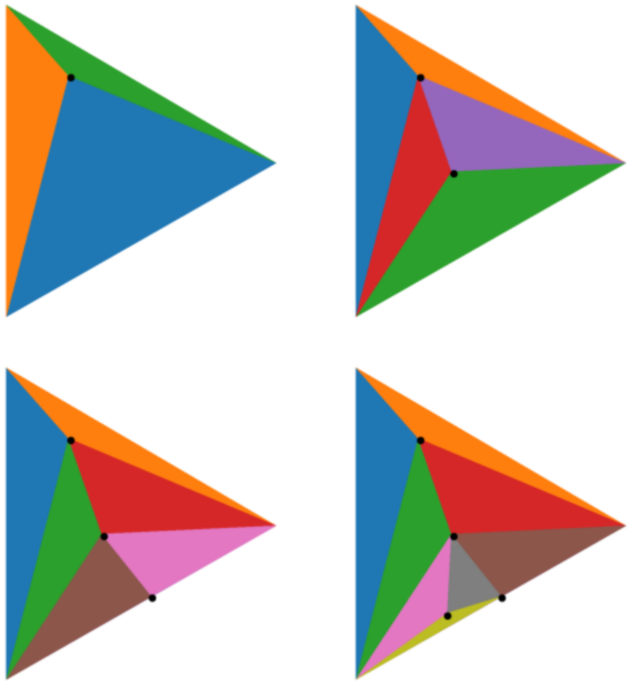
DEEP UNDERGROUND NEUTRINO EXPERIMENT

WARWICK

Delaunay triangulation

- Given plans re graph and transformer networks we will want to have the ability to generate a baseline graph relating hits
- Delaunay triangulation provides a technique for doing this
  - Triangulation such that no point in the set of points resides inside the circumcircle of any triangle formed by the method
  - Nice property is that it produces "minimal" graphs, so each point isn't connected to a thousand other points, just its local neighbourhood, and in a way that the minimum angle of each triangle is maximized
- Some interesting findings and potentially useful avenues for moving forward

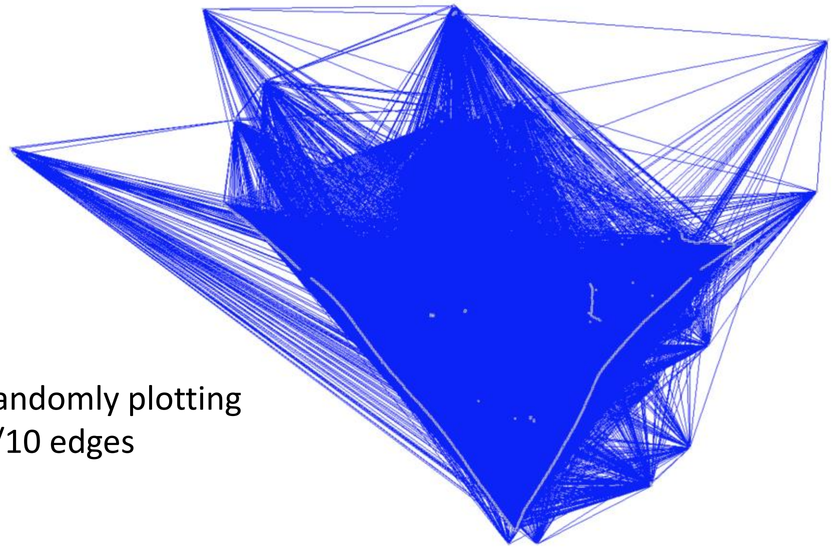
Delaunay triangulation



- The procedure starts from a minimal enclosing triangle for all hits
- For each hit (specific configuration order dependent, but always valid)
  - Check if the hit resides within circumcircle of each existing triangle
  - If it does, identify unique edges of the containing triangle then delete the containing triangle
  - Define new triangles from the new vertex and vertices of the unique edges
- When done, delete the bounding triangle vertices and associated edges

The problem with Delaunay triangulation

- The tessellation produced by Delaunay triangulation works well for nodes drawn from a spatially uniform distribution
- Our events don’t look like that
- Hits are typically highly correlated and this yields some unattractive features



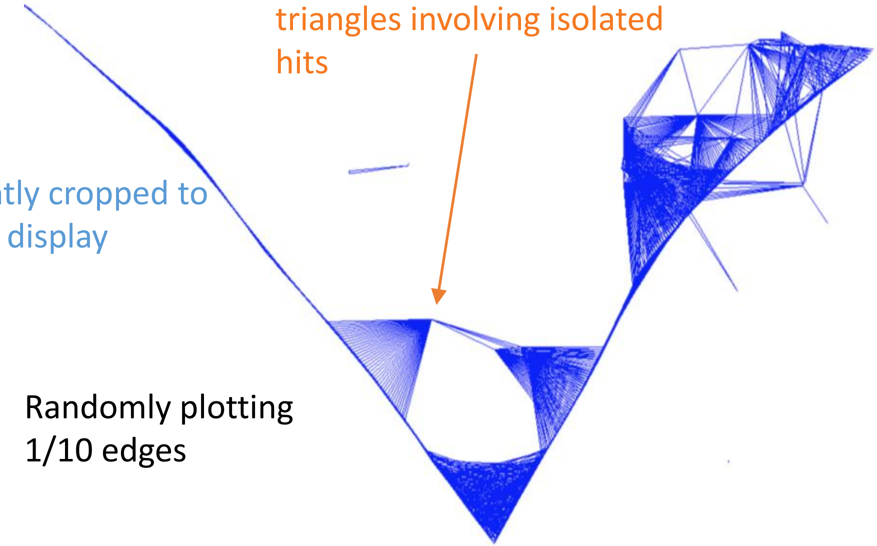
Randomly plotting
1/10 edges

The problem with Delaunay triangulation

- This can be mitigated to some degree by considering edge length ratios
- But many edges are constructed between points that aren't logically connected

Tessellation likes to create triangles involving isolated hits

Slightly cropped to help display



Randomly plotting 1/10 edges

The problem with Delaunay triangulation

- We can prune the graph by considering only the shortest two edges linked to any given node
- This is looking much more useful as a graph, but…
  - We’re a long way from our original Delaunay triangulation (and it’s not computationally trivial to do)
  - Our graph is disconnected at points – there are instances where that could be useful, but not really for a GNN

Slightly cropped to help display

An alternative

- Can we build this graph (or something like it) directly?
- Consider an adjacency matrix describing the distance between all points in an event
  - Sounds horrifying, but, with Eigen (thanks to Ryan for the suggestion), you can do this extremely quickly
  - It's a bit of a pain to figure out the implementation, but once there, it's quite short, and **much** faster than my Delaunay implementation (delaunator package may do better)

1. Package the hits up in an Nx2 matrix
2. Calculate a broadcasted difference and squared norm for each hit
3. Identify minimum coefficient to define the first edge, set coefficient to infinity
4. Repeat to get the second edge
5. Check the second edge doesn't just project beyond the first edge, throw it out if it does

(Steps 2–4: **Vectorized**)

- Repeat for all hits
- You now have a collection of edges

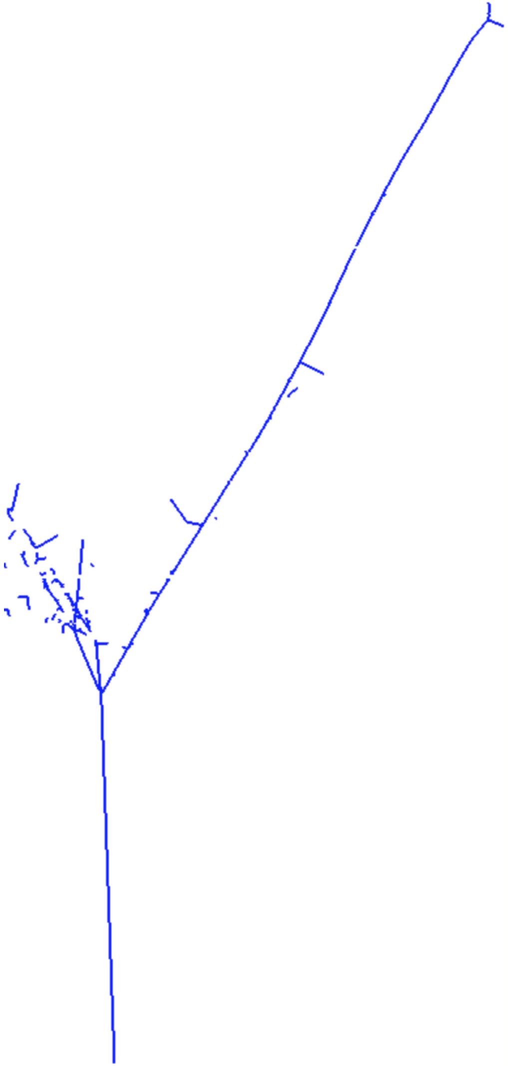
An alternative

- The resultant graph is similar to the pruned Delaunay graph, with far less code
- We still have disconnected regions
  - This can be addressed with additional post-processing
  - We can construct Nx2 matrices of each disconnected region and identify the closest approach to connect them
  - This can be vectorized, just like the initial steps

Slightly cropped to help display

An event with an EM shower

- Disconnections more evident here, highlighting the need for the extra connection step
- Each hit only attempts to connect to two other hits at most, this might be too extreme for GNNs
  - Hits can have more than two connections depending on how many other hits try to connect to them, but in general edge multiplicity is low
  - We could explore upping the edge count in the provisional connection stage – *might* help us interconnect shower hits from the outset
  - There will be practical limits to increasing the allowed edge count
  - Needs some care to avoid generating extra colinear edges



Connecting the disconnected regions

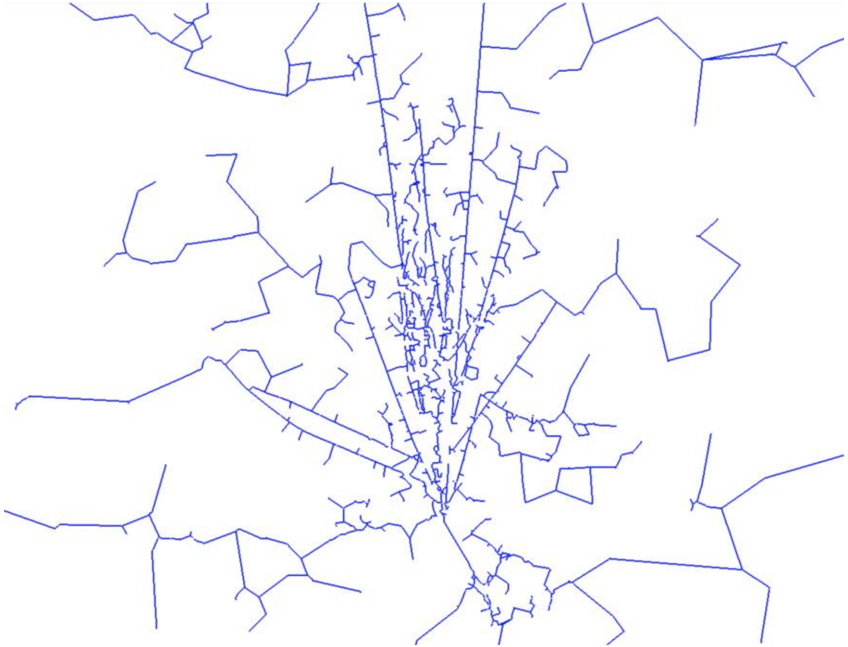
- Recall our example event (note, full W event display here)
- Identify each connected region and then look for closest approach to other regions
- Connect closest hits until no regions remain disconnected
  - With vectorization, this step continues to run quickly

Initial graphs

Fully connected graph

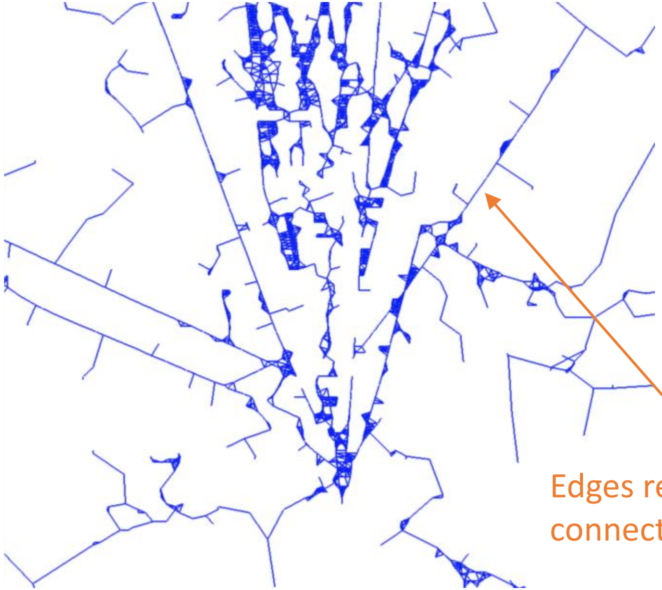
Connecting the disconnected regions

- Fully connected dense shower region
- This is where favouring low edge multiplicity might be going too far
- GNNs pass messages to more distant nodes by following edges
  - The depth of the message passing may need to be quite high to connect some spatially local nodes
- Increasing the baseline per node edge count may allow a useful trade-off
  - Maintain spatially local edges between nodes
  - Allow neighbouring nodes to be reached in few steps

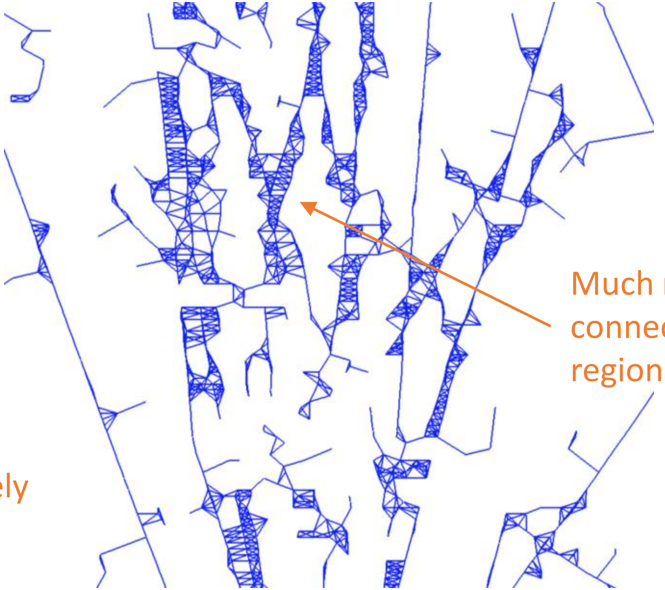


Increasing edge density in shower regions

- Step through the adjacency matrix considering N closest hits
  - Limit the number of hits to consider
  - Veto co-linear edges emerging from the source hit
  - Veto “long” edges
- Same event as previous slide
  - Allowing up to 5 edges emerging from source
  - Minimum opening angle ~5^{o}
  - Maximum secondary edge length $\sqrt{3}$ cm



Edges remain sparsely connected



Much more densely connected shower region

Summary

- Restricted, locally connected graphs are likely to be useful in various deep learning contexts (and perhaps non-DL too)
- Delaunay triangulation produces minimal tessellations, but also many edges of questionable value due to the nature of our hit distributions (many non-local connections)
  - By the time you prune this, you have to ask if it was worth generating the tessellation in the first place
- Alternative approach focuses on minimal, local connections
  - Very fast with Eigen vectorization
  - Low total edge count relative to the number of hits
  - Option to have disconnected, or fully connected graphs
- Tunable edge per node to help GNN message passing
  - Retains sparse edges
  - Allows dense connections within shower regions
  - Overall edge count remains relatively low