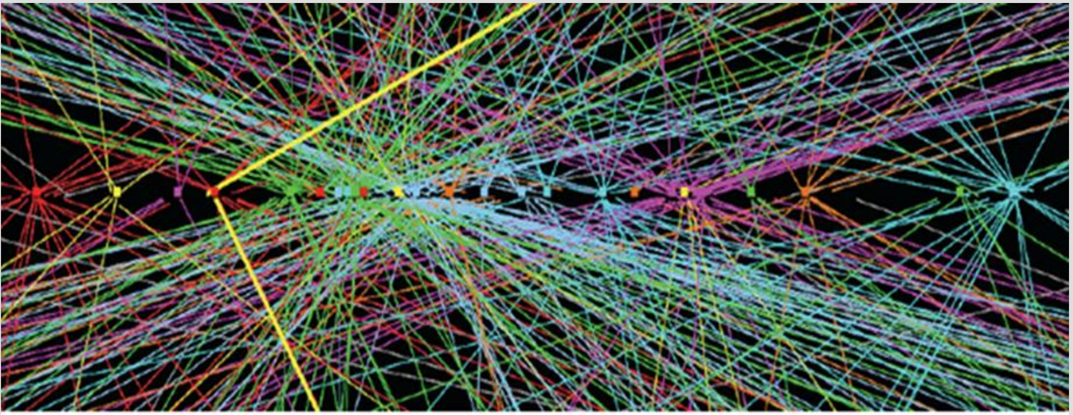
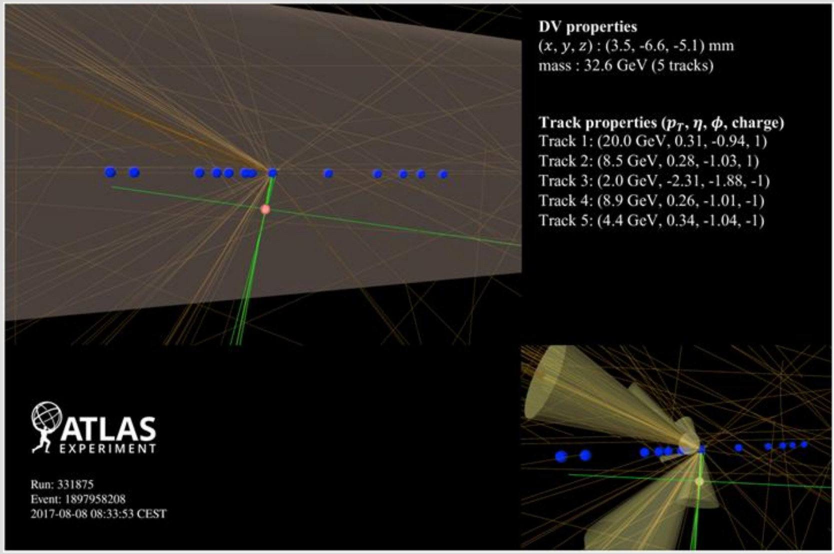
TrackOpt vertex project update

V. Kostyukhin
Siegen university

Reminder



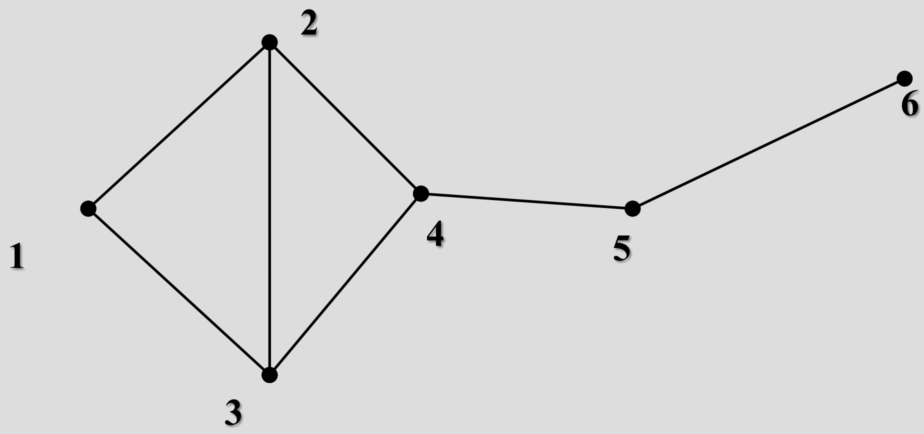


<u>Problem:</u>
Having a set of tracks find all vertices (track production points)

<u>Solution idea:</u>

1. Create a track-track compatibility graph using a priory physics knowledge, i.e. explicitly calculate a point of closest approach of the 2 tracks in 3D space and the track/vertex parameters at this point.
2. Based on this information, estimate a probability that a given 2-track vertex is real
3. Use LMC algorithm to partition this graph with weighted edges

Graph



Example:
6 tracks (nodes)
7 possible 2-track vertices (edges)

Node vector of features(track parameters): td0, tZ, tPhi, tEta, tQoP, tTheta, tChi2, tNDoF, tCovD0, tCovZ,tCovD0Z, tSignif

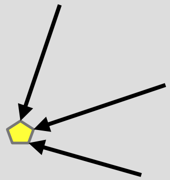
Edge vector of features(2-track vertex): vChi2, vX, vY, vZ, vcovXX, vcovXY, vcovYY, vcovXZ, vcovYZ, vcovZZ, vsumPt, vEta, vPhi, vDZ, massPiPi, vtrue, vCharge, isGamma, isKs, isLambda + truth_label

Prepared graphs saved in ROOT format (~1.5mb/ev, ~1400nodes/ev, ~26k edges/ev)

For the moment, to save CPU, for tracks closer that 3σ to the beamline no real vertex fit is done. Compatibility is calculated based on Z track position on the beamline (1D fit, like in the PV finding paper).

Edge weights with DGL

Edge weights estimation is implemented in DGL



1) Node_hidden_state = NN{Concat(Node_features, Mean(Edge_features)}

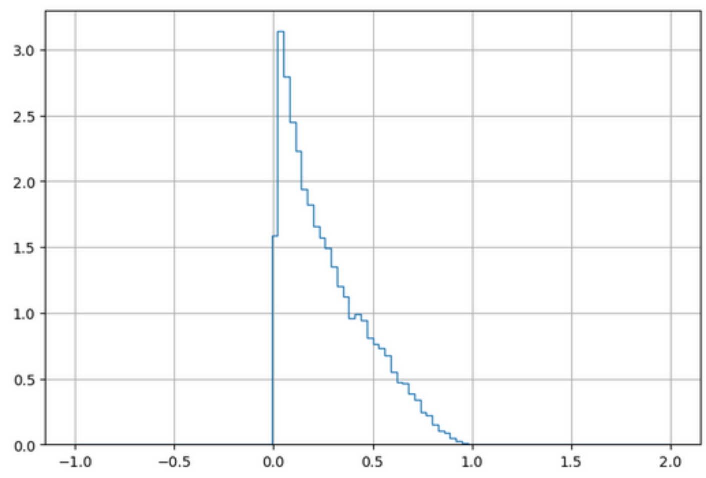
2) Edge_weight = NN{ Concat(Node_hidden_state_i, Edge_features, Node_hidden_state_j)}

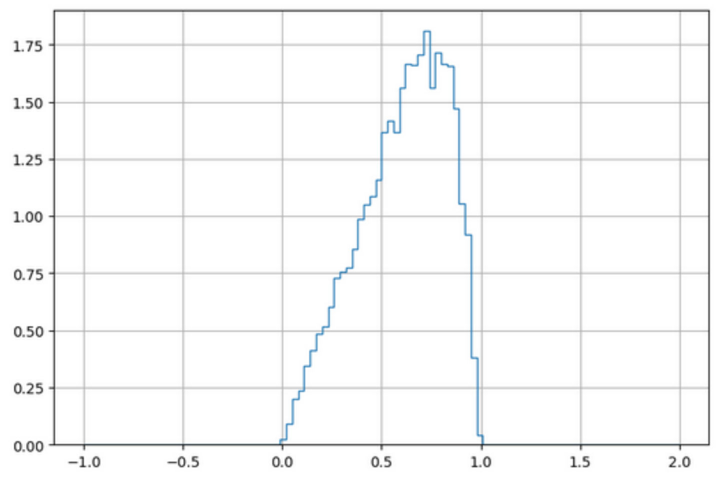
3) Node_hidden_state = NN{Concat(Node_features, Weighted_Mean(Edge_features)}
4) Upd_Edge_weight = NN{ Concat(Node_hidden_state_i, Edge_features, Node_hidden_state_j)}

Steps (3),(4) happened to be not needed, final weights are practically the same.

Loss – binary cross-entropy (classification)
Activations – Mish + Sigmoid to get probability

DGL results (very preliminary)





AUC ~ 0.84 for both primary and secondary edges(aka. 2-track vertices)

For comparison - XGBoost classification of edges based on the same edge features: AUC ~ 0.75

Next steps

1) Save training results and run LMC on weighted graphs
2) Implement metrics for comparison
3) Play with DGL
   - ✓ More graphs for training (currently batch of 50)
   - ✓ Different layers – LSTM, Transformers, etc,
   - ✓ Different reduction functions
   - ✓ …..
4) More into the future – try edge prediction approach based on node(track) features only. This will allow to estimate a performance gain due to a priory knowledge injected as edge features
   - Similarity to foundation model