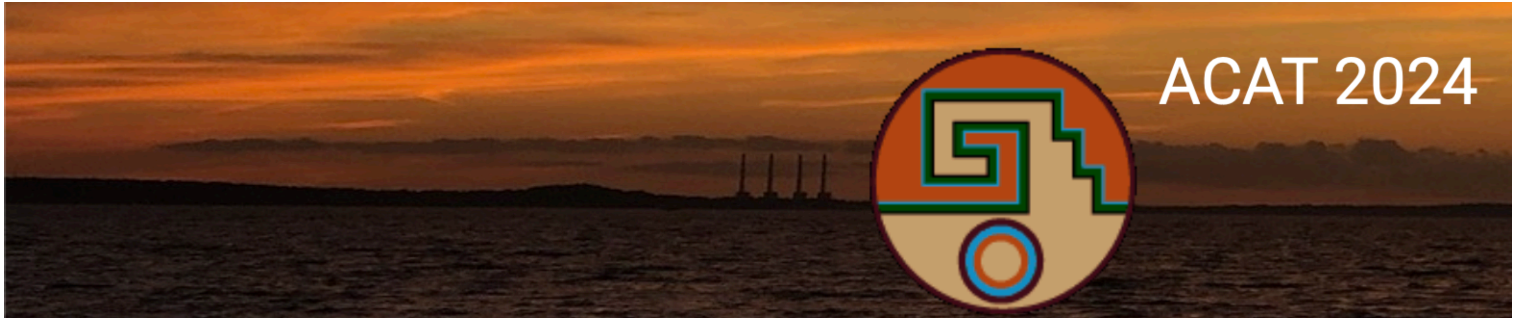
BERKELEY LAB
Bringing Science Solutions to the World

U.S. DEPARTMENT OF ENERGY
Office of Science

Reconstructing Particle Tracks in One Go with a Recursive Graph Attention Network

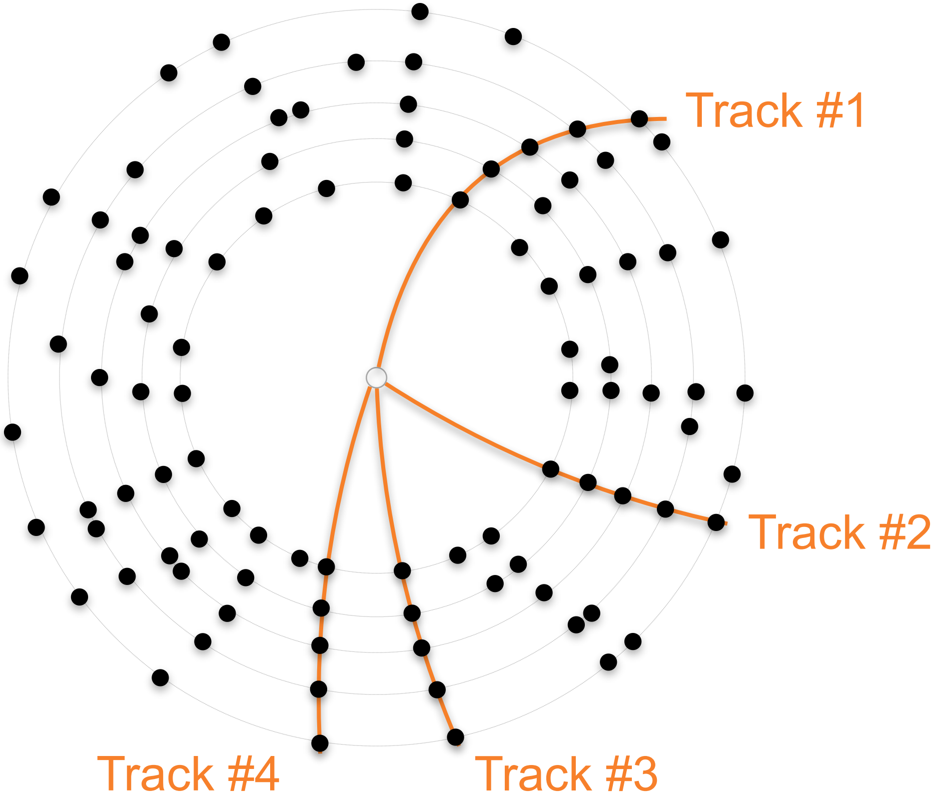
Jay Chan

Lawrence Berkeley National Laboratory



ACAT, Stony Brook NY, March 11, 2024

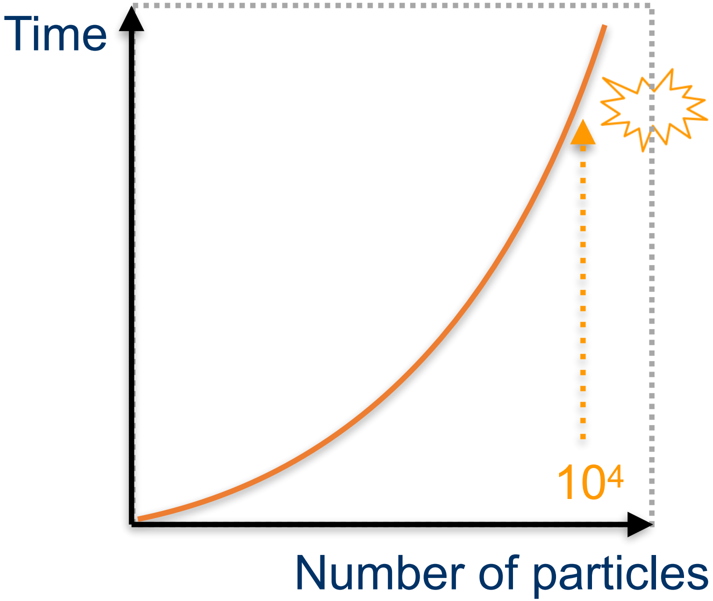
Track reconstruction is a challenge task



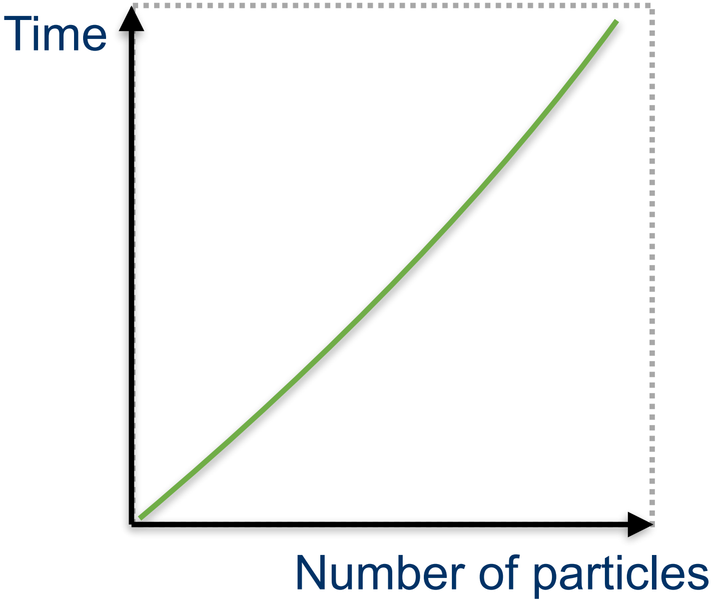
~O(10^4) particles per collision event at HL-LHC
→ ~O(10^5) hits in ATLAS ITK

A more efficient tracking algorithm is needed

Traditional algorithm:
Combinatorial Kalman Filter (CKF)

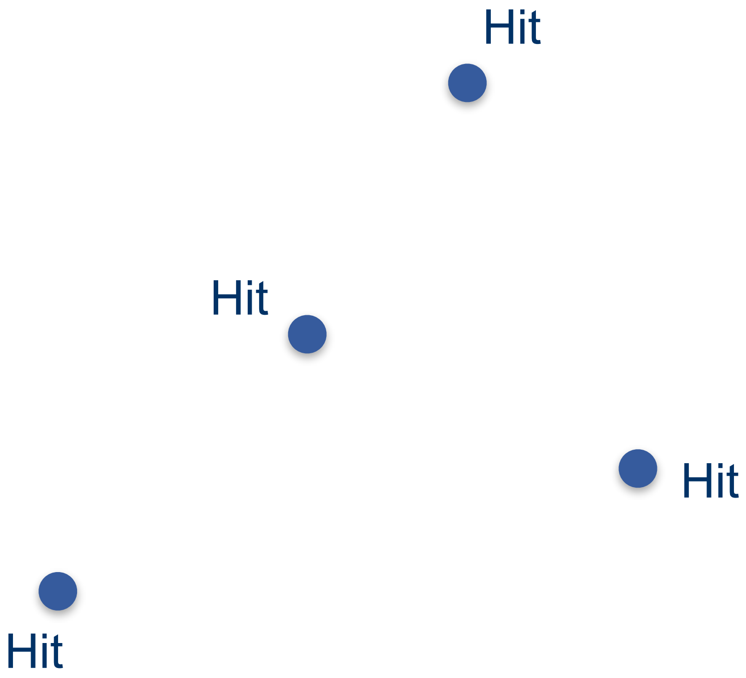


ML-based algorithms

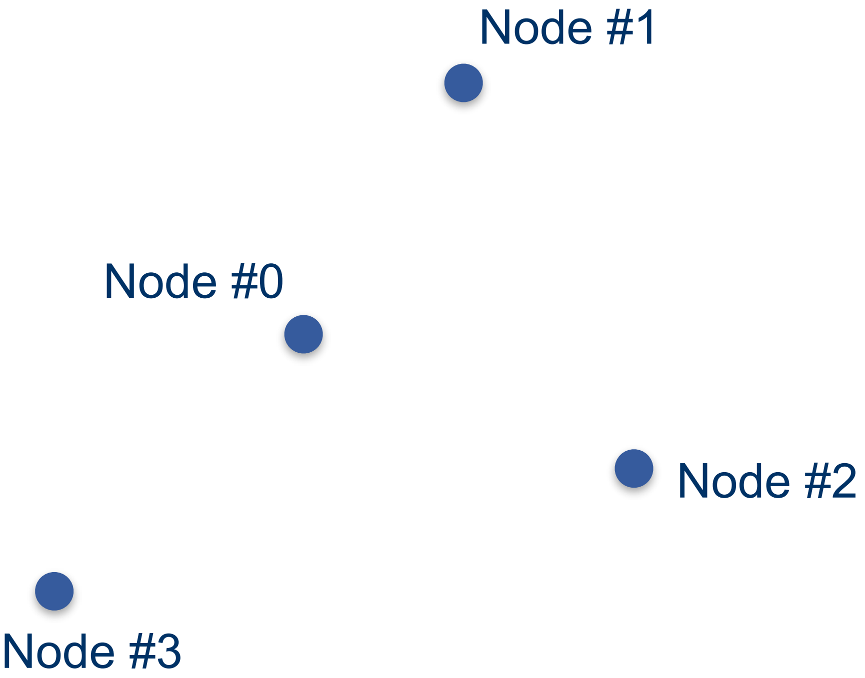


Using graph techniques for tracking is a natural choice

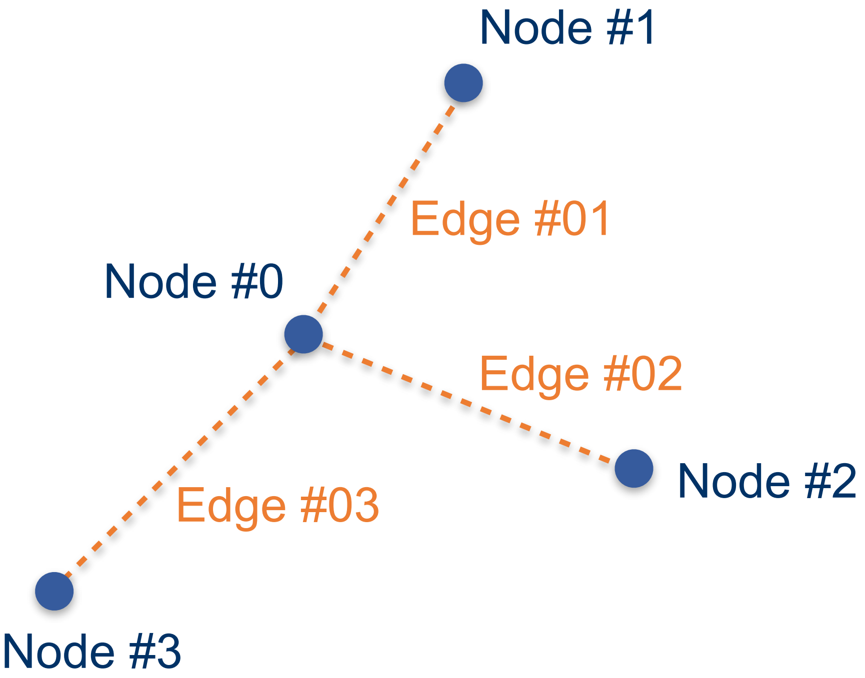
Events as graphs



Hits as nodes

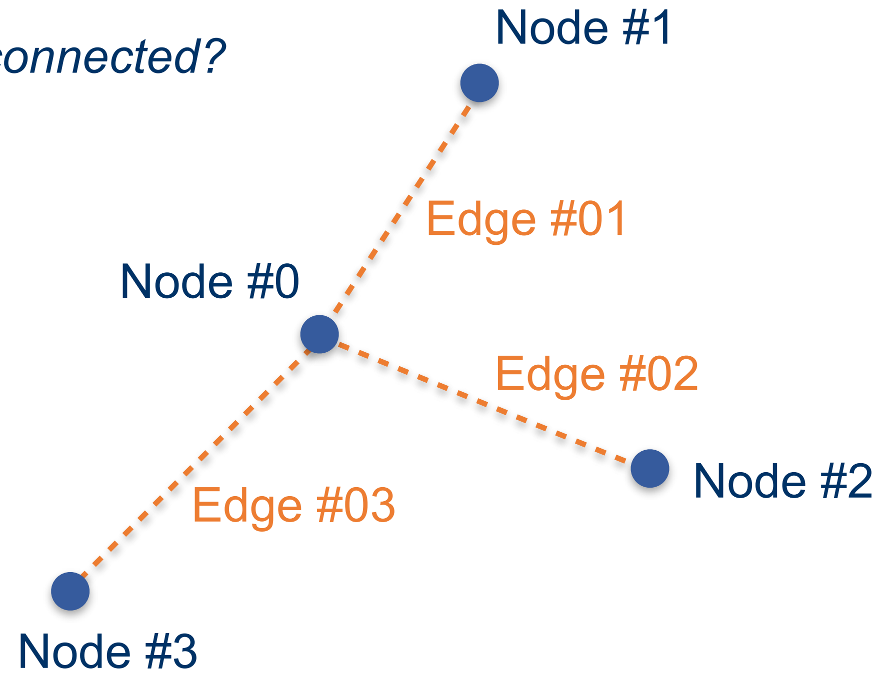


Nodes connected by edges



Nodes connected by edges

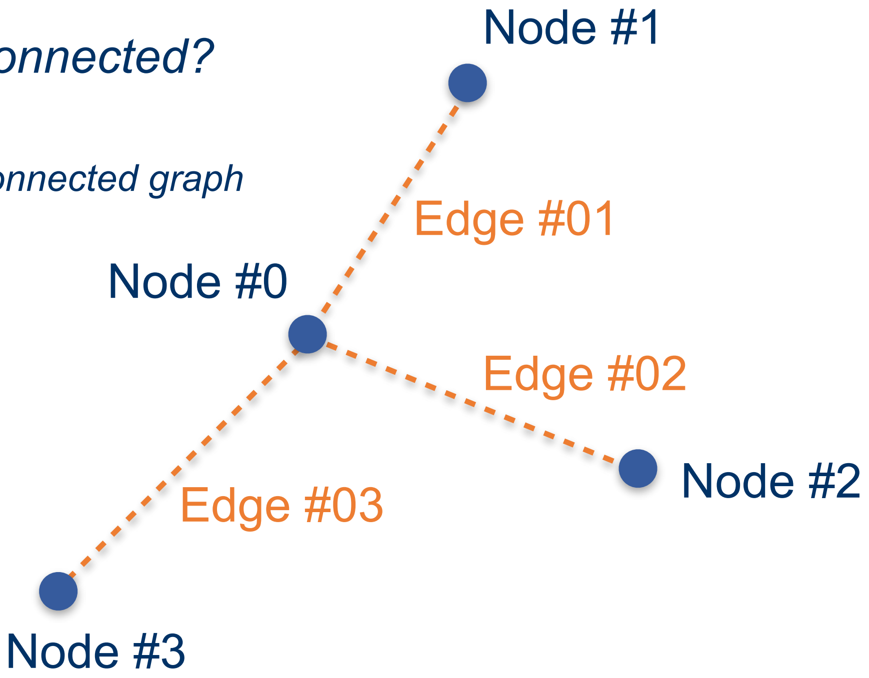
How are the nodes connected?



Nodes connected by edges

How are the nodes connected?

$O(10^5)$ hits
→ $O(10^{10})$ edges in a fully connected graph

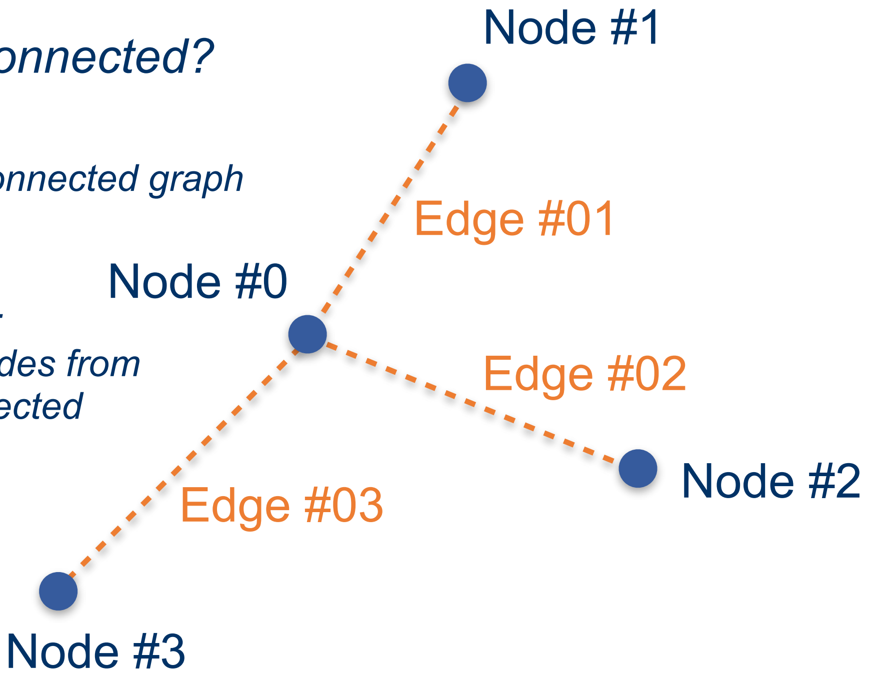


Nodes connected by edges

How are the nodes connected?

$O(10^5)$ hits
→ $O(10^{10})$ edges in a fully connected graph

Build graph in a smarter way:
*Only **relevant nodes** (i.e. nodes from the same particles) are connected*



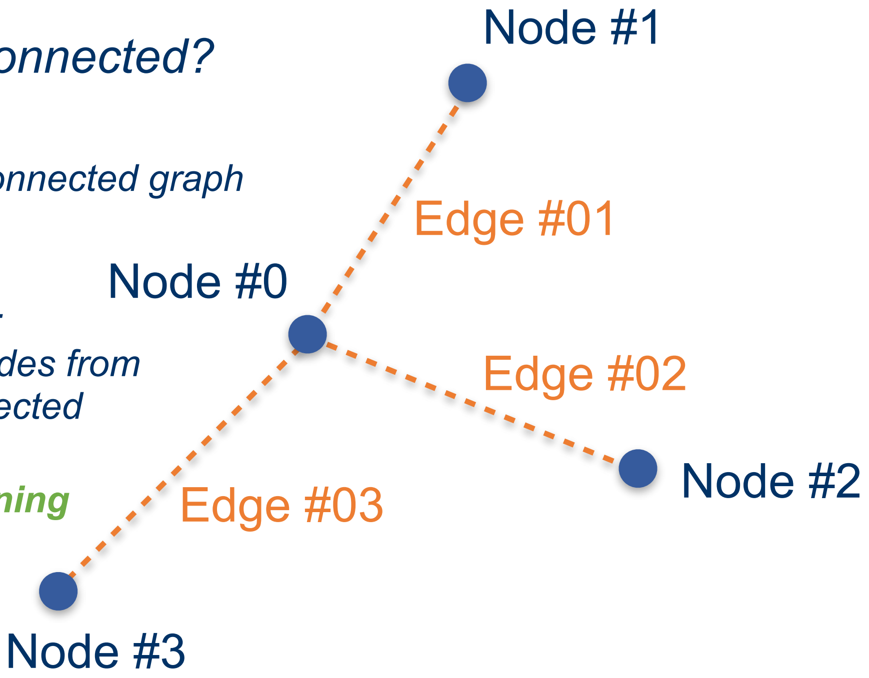
Nodes connected by edges

How are the nodes connected?

$O(10^5)$ hits
→ $O(10^{10})$ edges in a fully connected graph

Build graph in a smarter way:
*Only **relevant nodes** (i.e. nodes from the same particles) are connected*

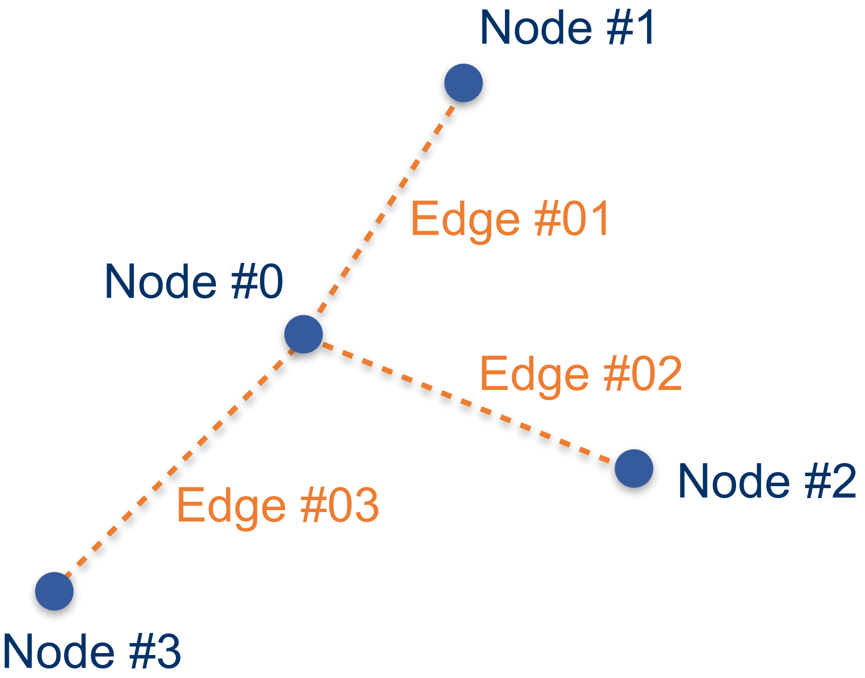
*Graph built with **metric learning** or **module map***



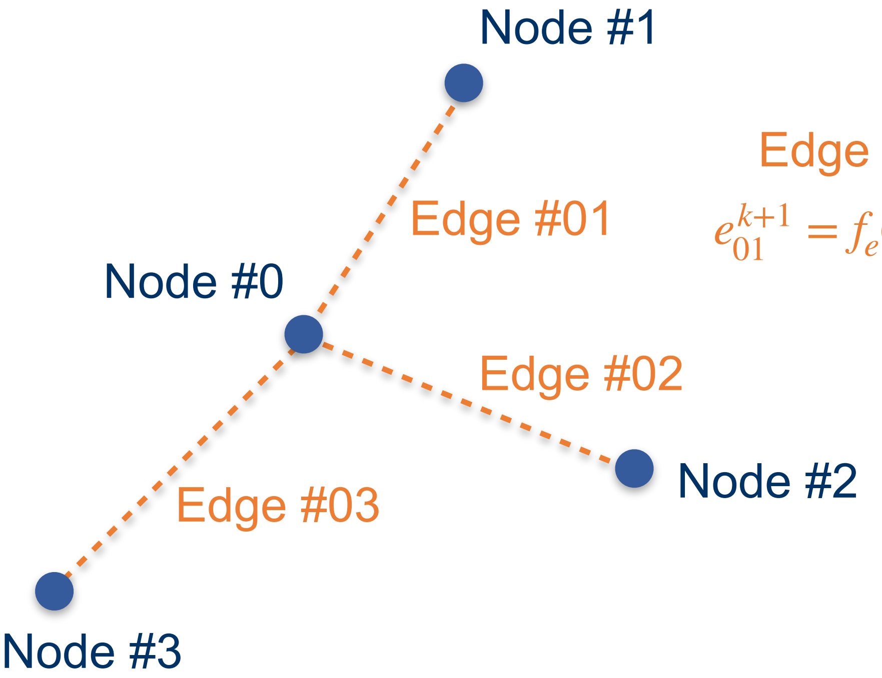
Metric learning by ExaTrkX:
[arXiv: 2103.06995]

Module map:
[arXiv: 2103.00916]

Message passing through event graph structure



Message passing through event graph structure



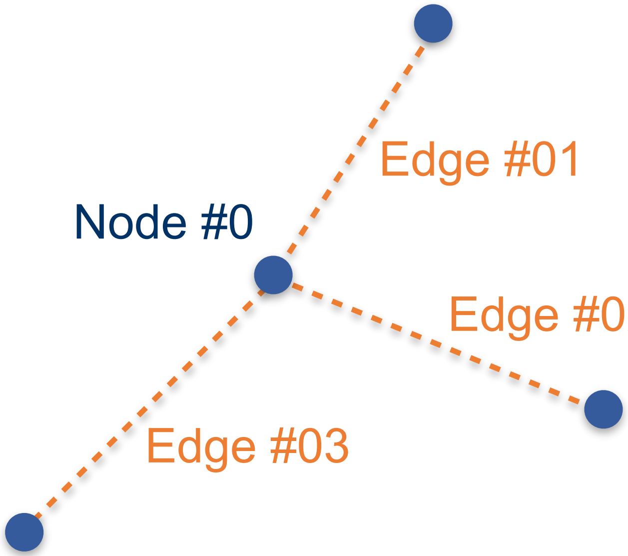
Edge update:

$$e_{01}^{k+1} = f_e(e_{o1}^{k}, n_0^{k}, n_1^{k})$$

Message passing through event graph structure

Node update:

$$n_0^{k+1} = f_n(n_o^k, \mathrm{agg}(e_{0j}^k))$$



Edge update:

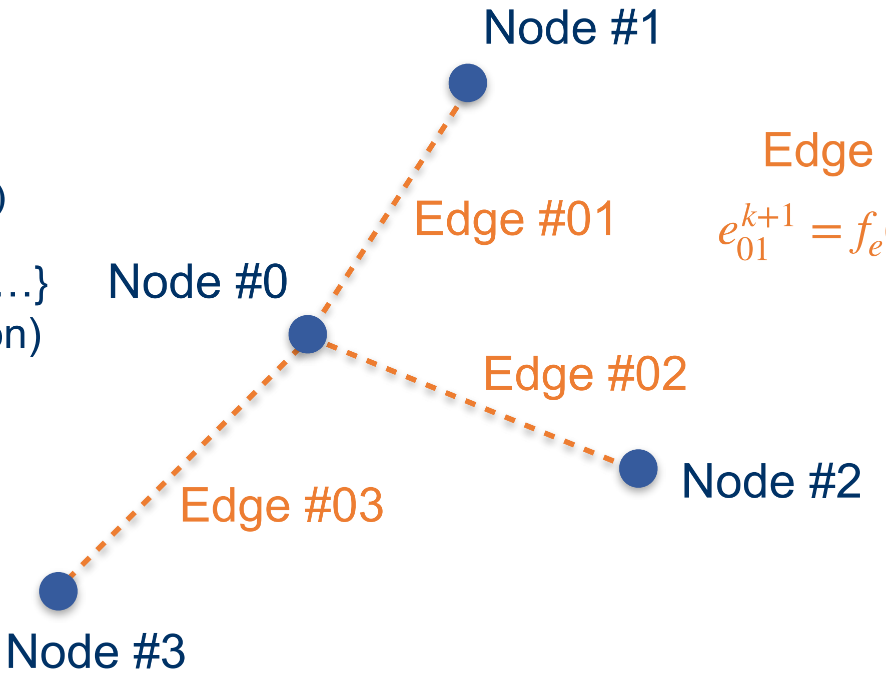
$$e_{01}^{k+1} = f_e(e_{o1}^k, n_0^k, n_1^k)$$

Message passing through event graph structure

Node update:

$$n_0^{k+1} = f_n(n_o^k, \mathrm{agg}(e_{0j}^k))$$

agg ∍ {sum, mean, max…}
^ weighted sum (attention)



Edge update:

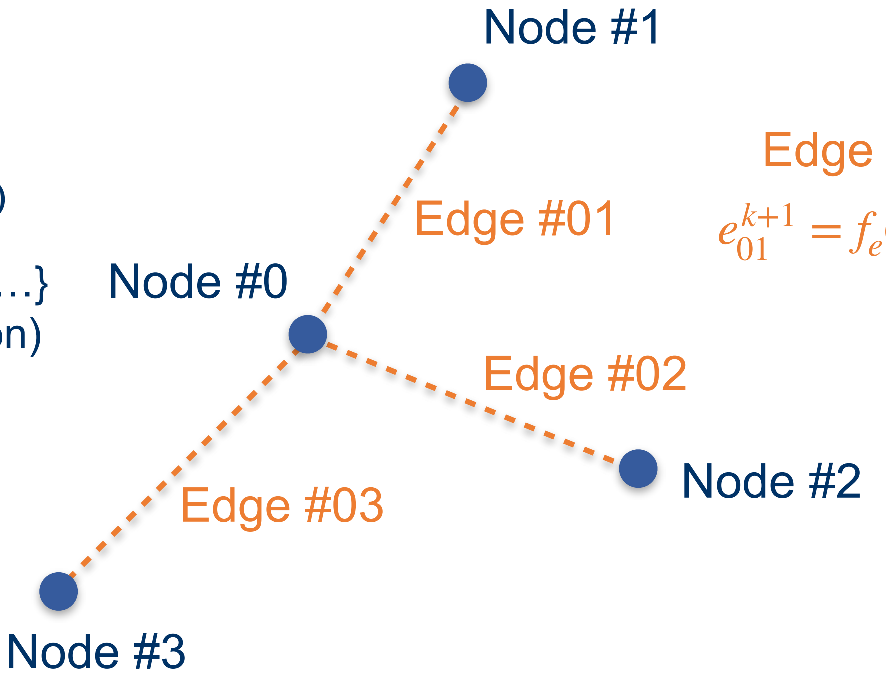
$$e_{01}^{k+1} = f_e(e_{o1}^k, n_0^k, n_1^k)$$

Message passing through event graph structure

Node update:

$$n_0^{k+1} = f_n(n_o^k, \mathrm{agg}(e_{0j}^k))$$

agg ∋ {sum, mean, max…}
^ weighted sum (attention)

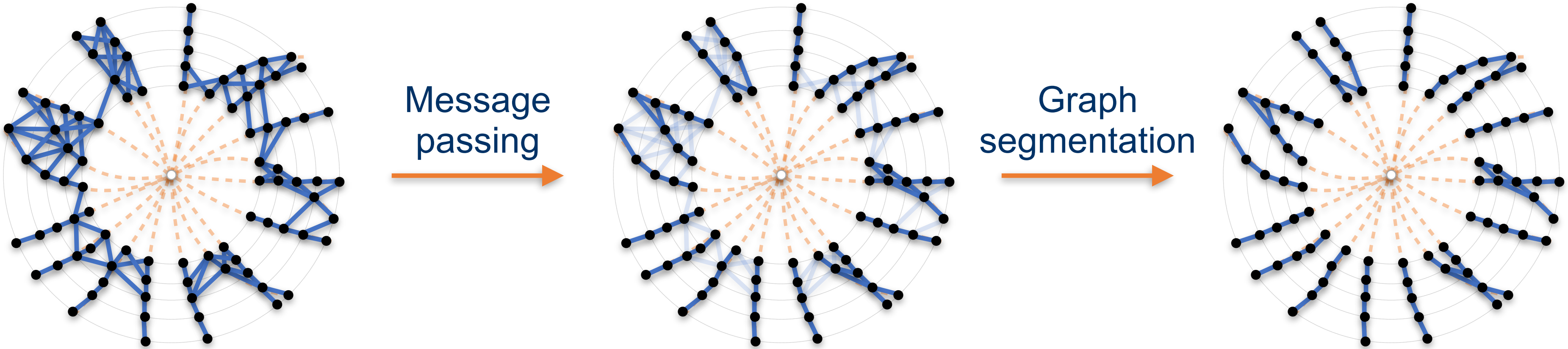


Edge update:

$$e_{01}^{k+1} = f_e(e_{o1}^k, n_0^k, n_1^k)$$

Able to learn key node / edge features from environment

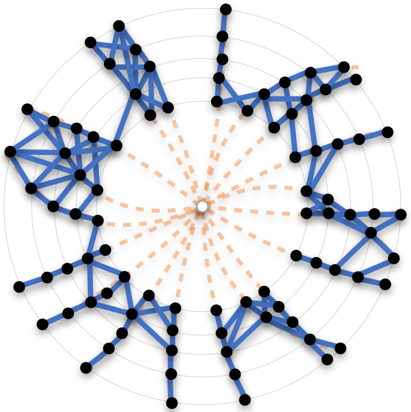
Track reconstruction as edge classification



Graph (nodes + edges) | Edge scores | Track labels

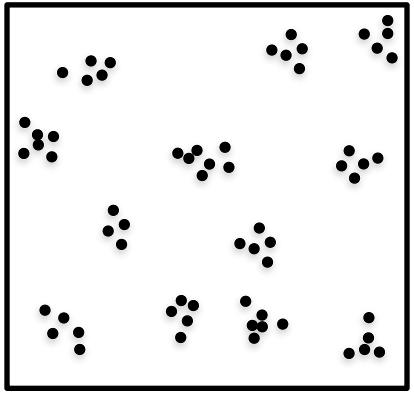
Example: ATLAS GNN4ITK pipeline (ATL-SOFT-PROC-2023-047)
See Daniel's talk Tomorrow on latest GNN4ITK results

Track reconstruction as object condensation



Graph
(nodes + edges)

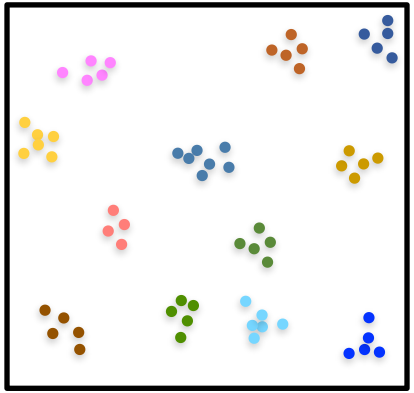
Message
passing



Node embedding

Clustering

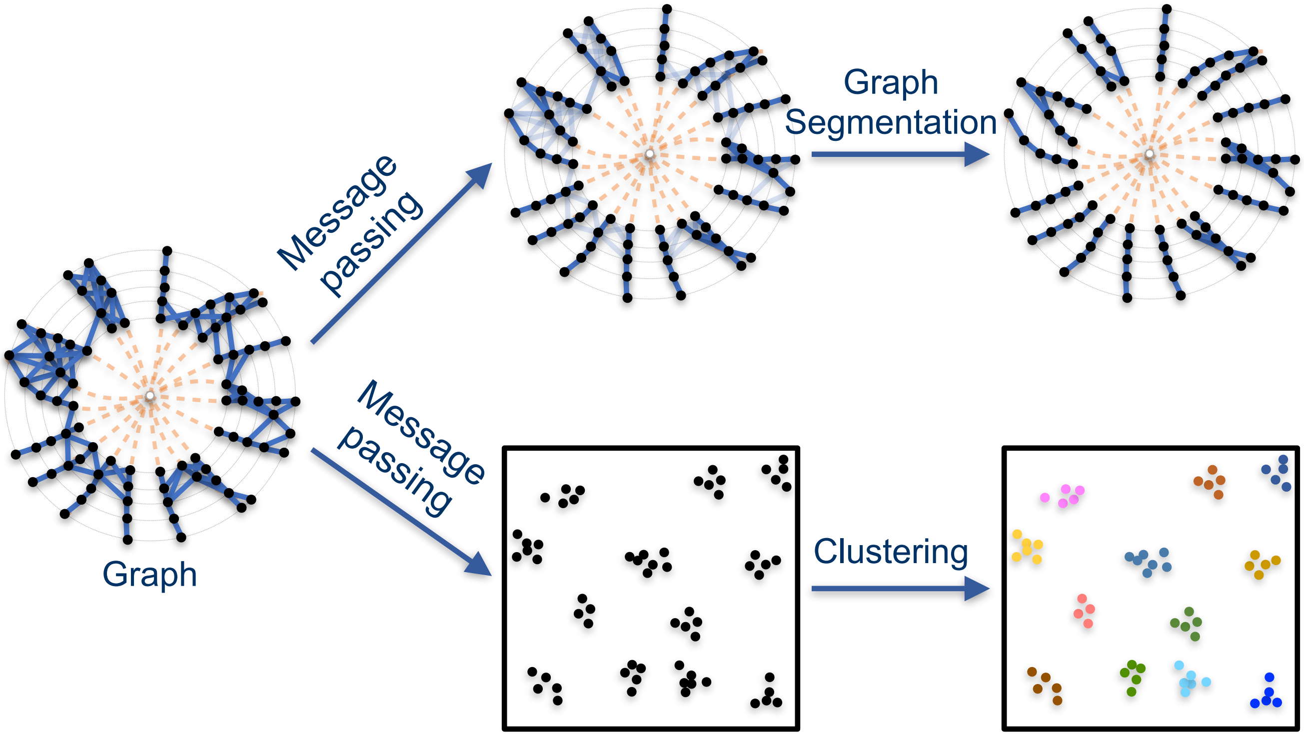
E.g. k-mean, DBSCAN,
HDBSCAN…



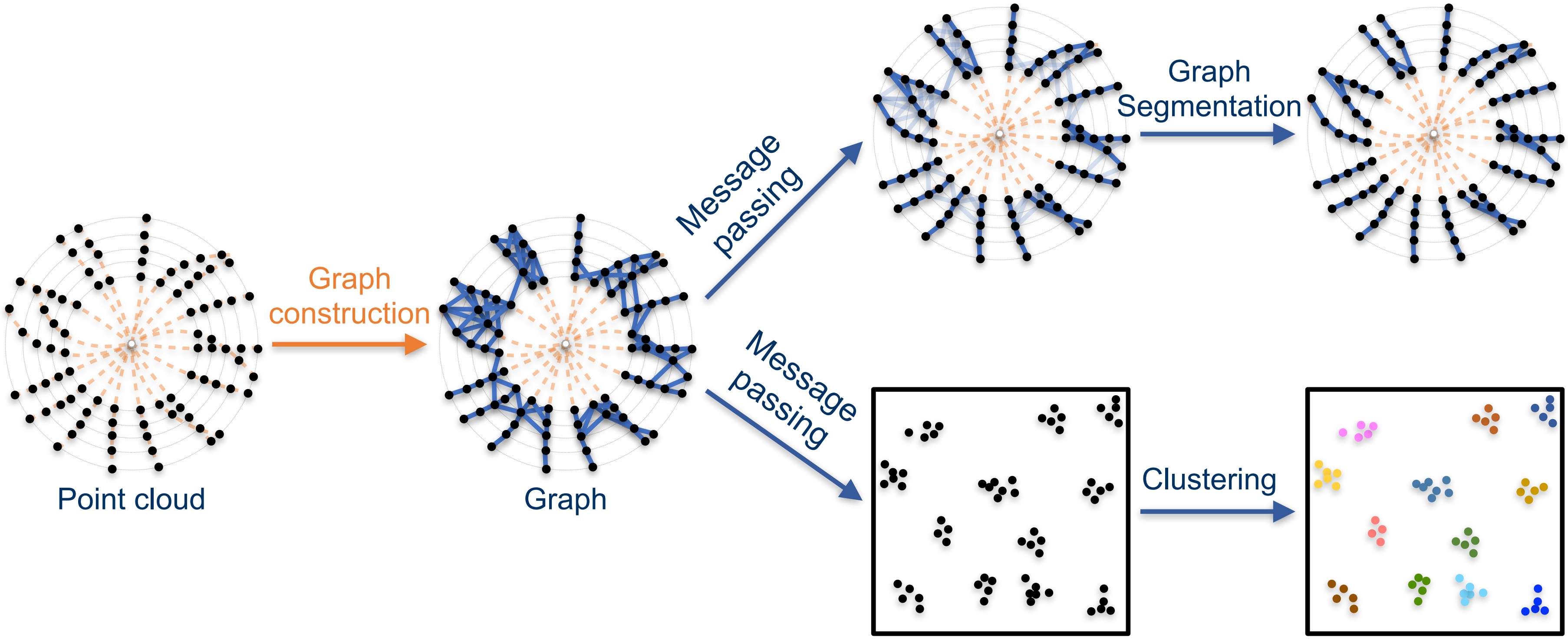
Track labels

Examples: K. Lieret et. al. (arXiv:2312.03823), D. Murnane (influencer)
See Kilian's earlier talk

Graph reconstruction required as a first step in pipeline



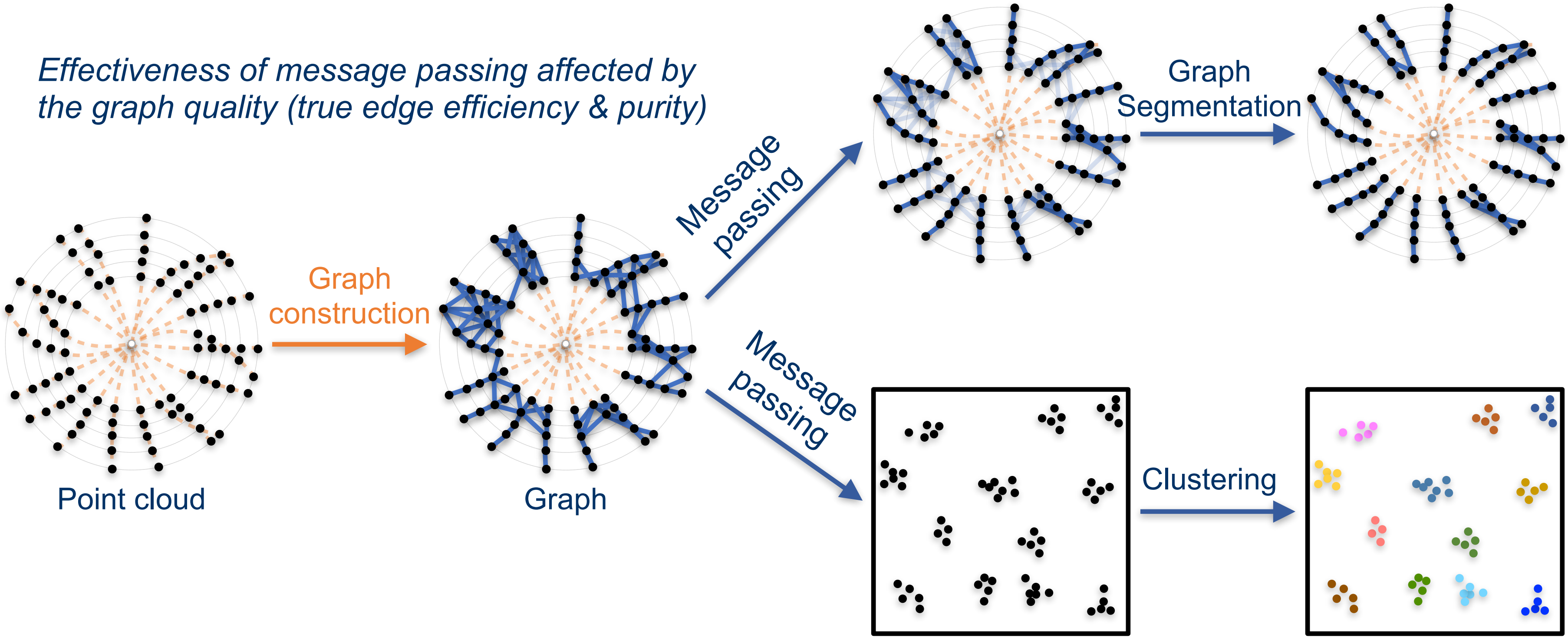
Graph reconstruction required as a first step in pipeline



BERKELEY LAB

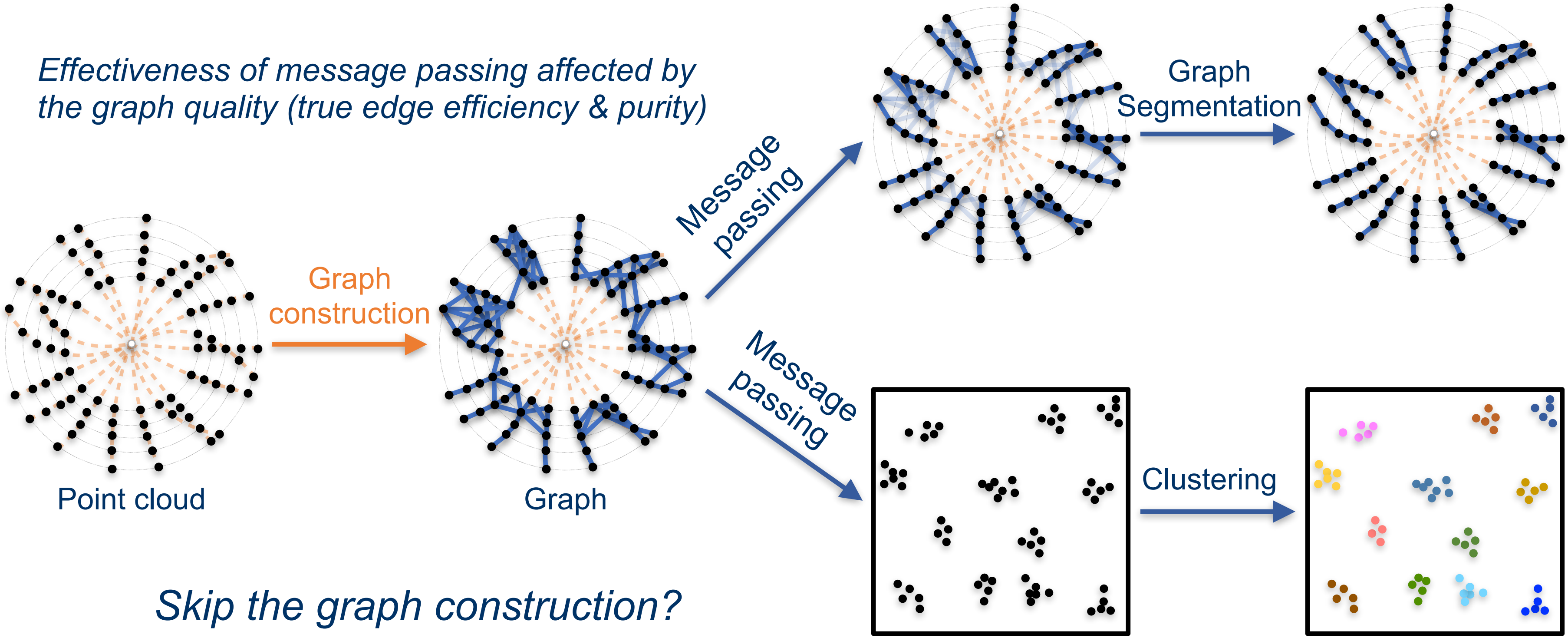
Graph reconstruction required as a first step in pipeline

Effectiveness of message passing affected by the graph quality (true edge efficiency & purity)



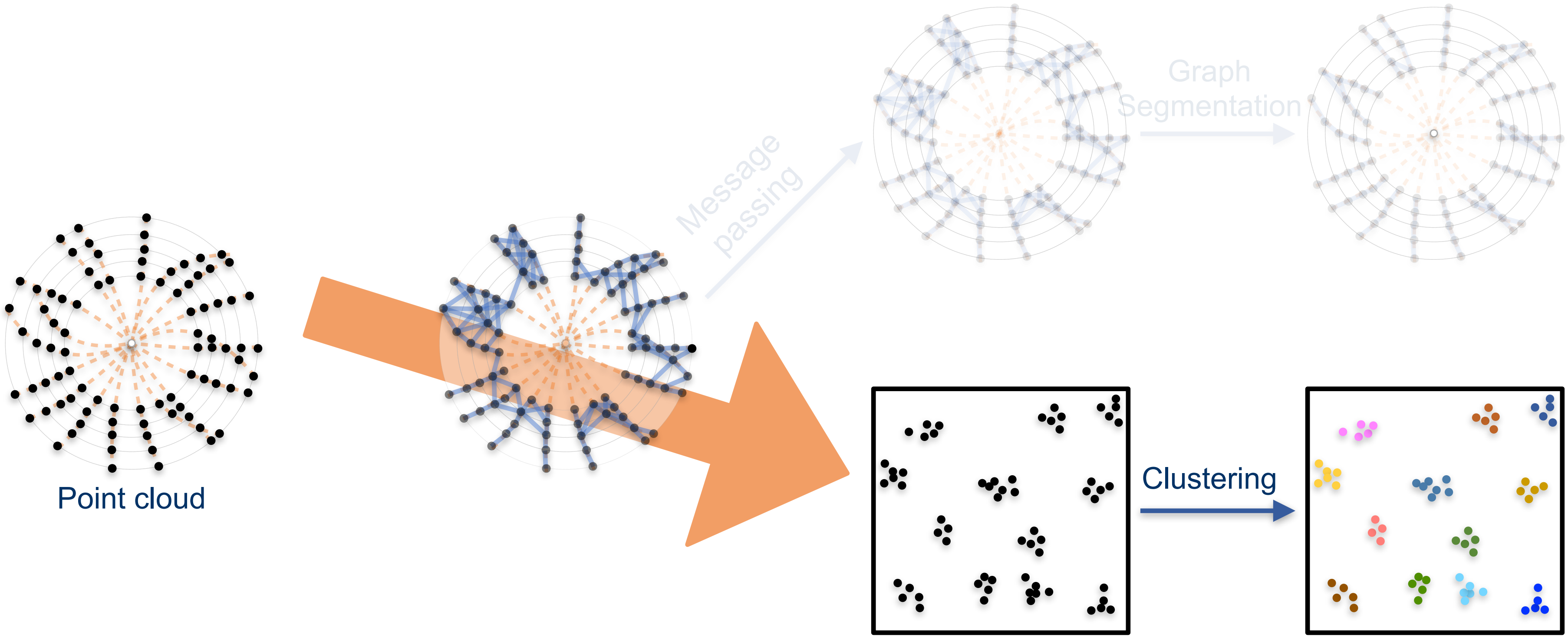
Graph reconstruction required as a first step in pipeline

Effectiveness of message passing affected by the graph quality (true edge efficiency & purity)

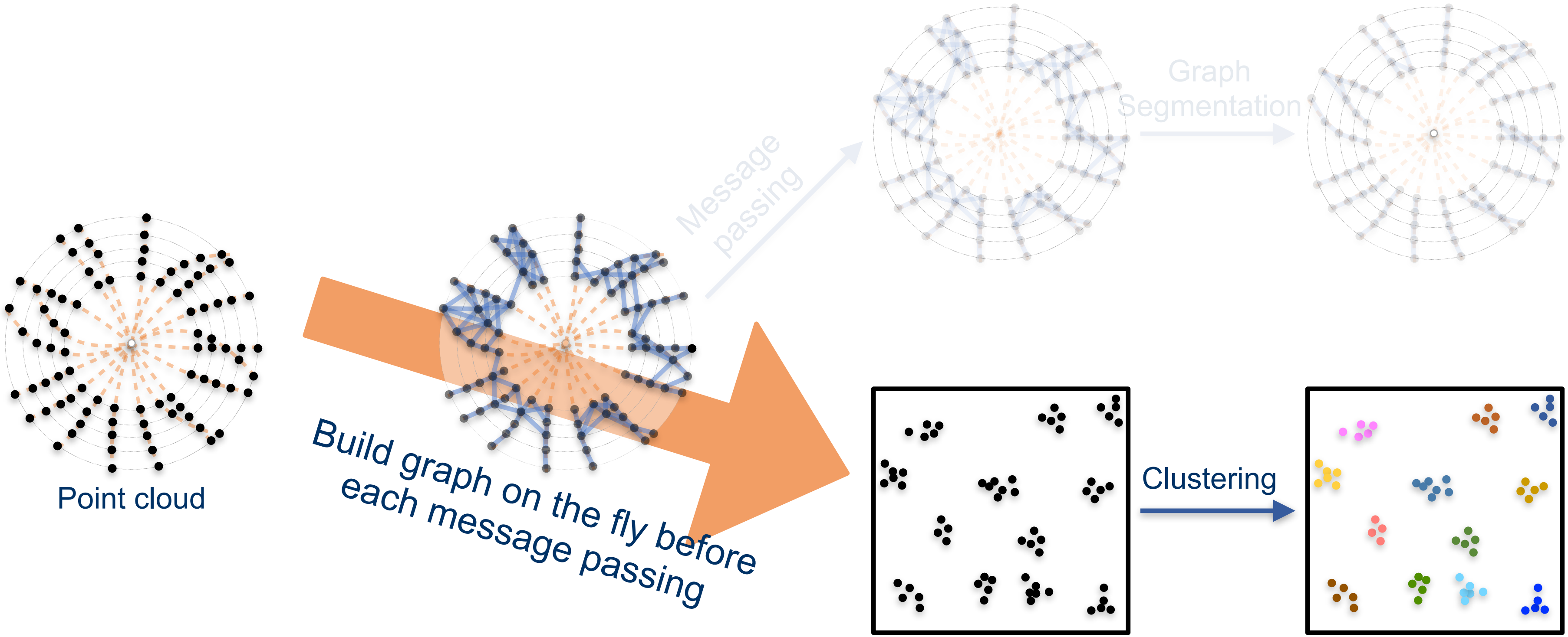


Skip the graph construction?

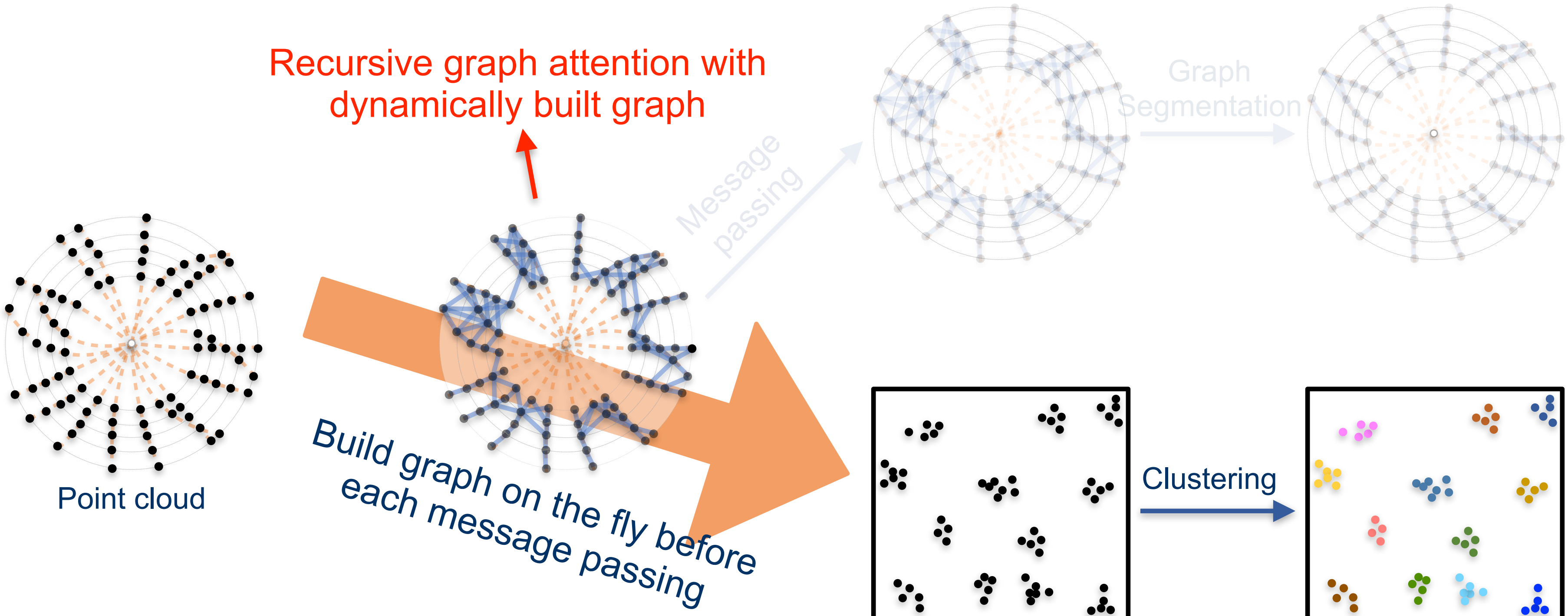
Message passing, graph construction, all at once



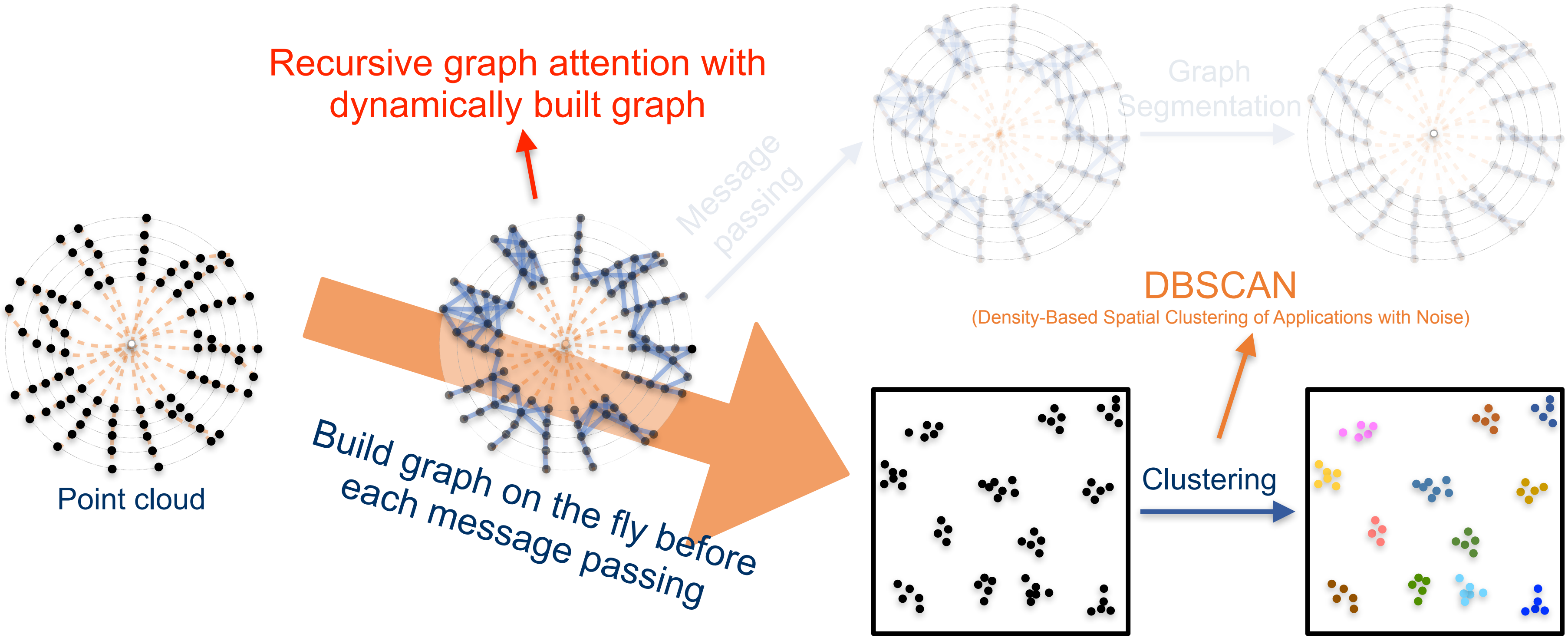
Message passing, graph construction, all at once



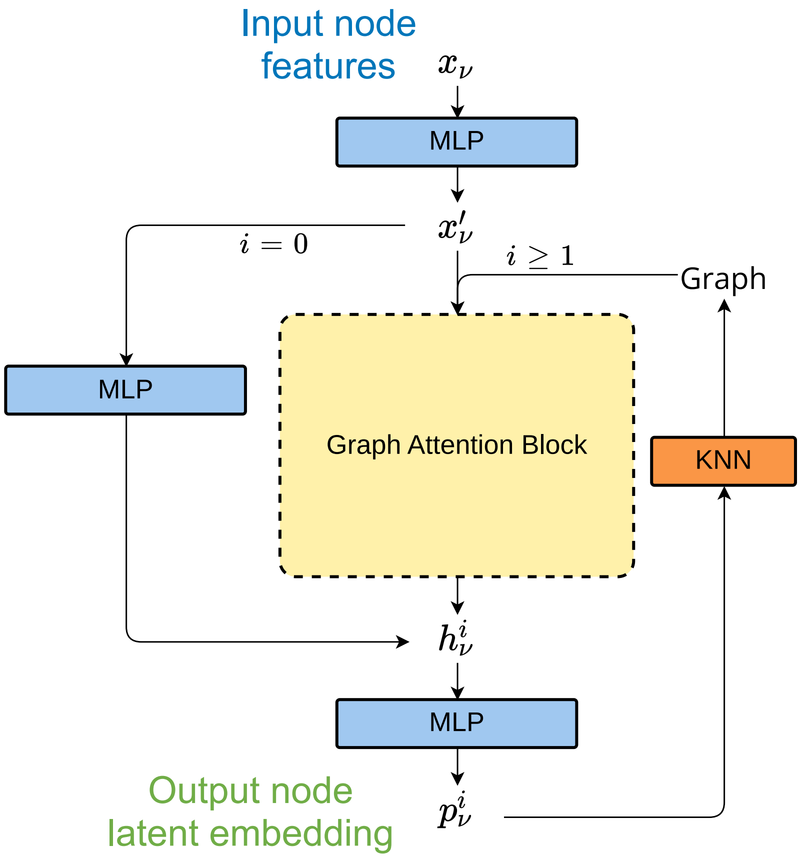
Message passing, graph construction, all at once



Message passing, graph construction, all at once

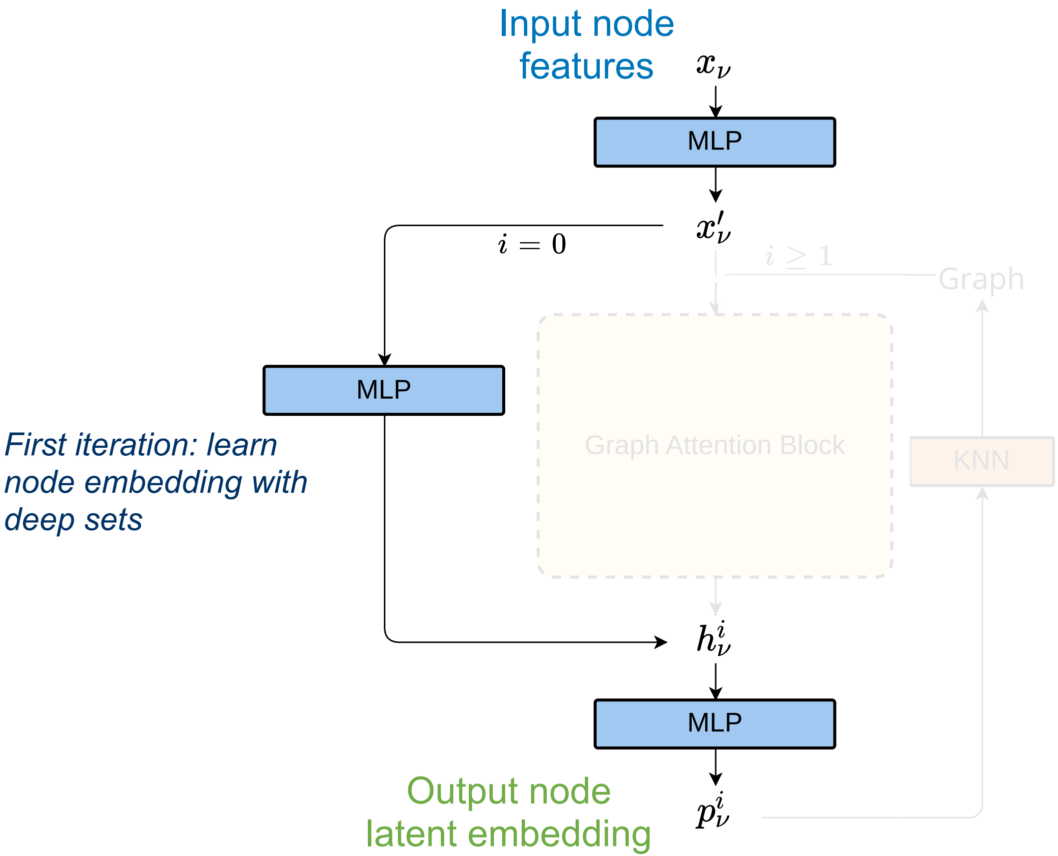


Recursive Graph Attention Network

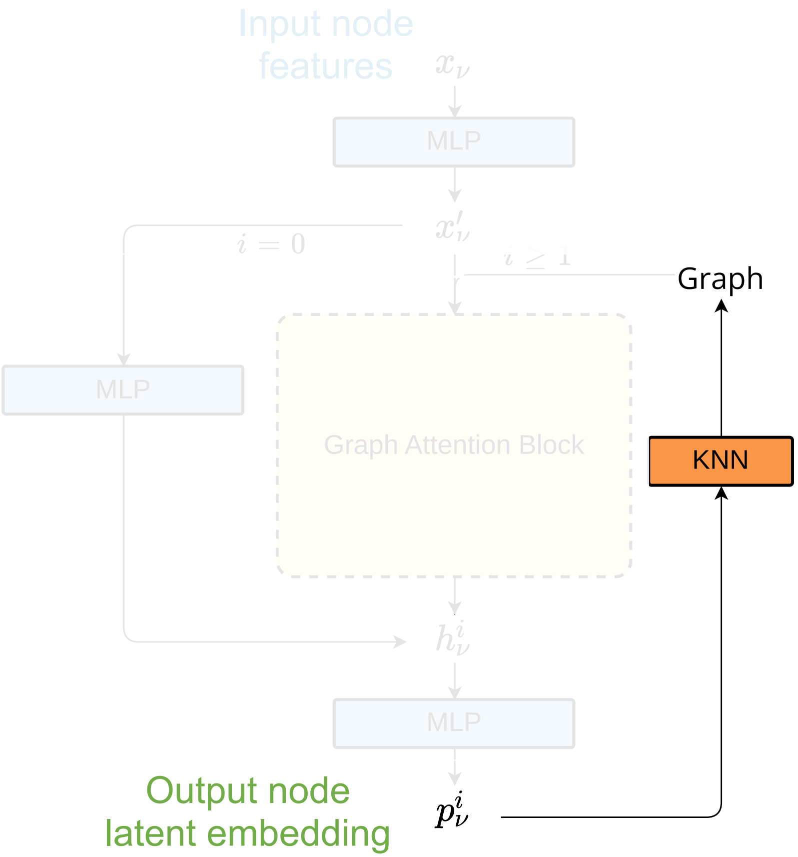


BERKELEY LAB

Recursive Graph Attention Network



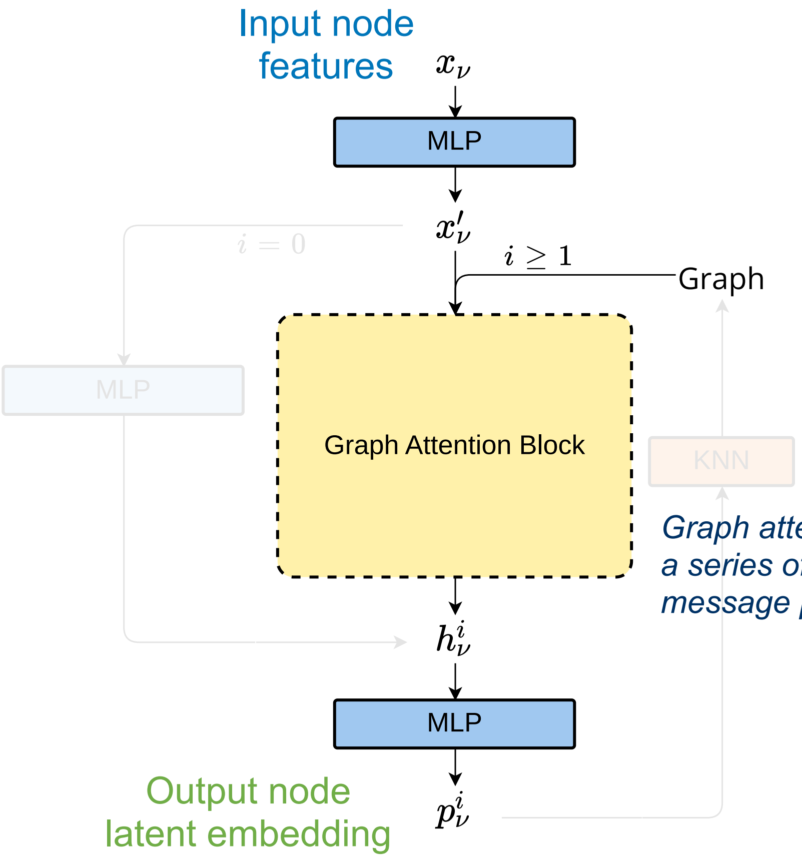
Recursive Graph Attention Network



Evolutional KNN graph

K-nearest-neighbor (KNN): Connect each node to its k closest nodes in the embedding space

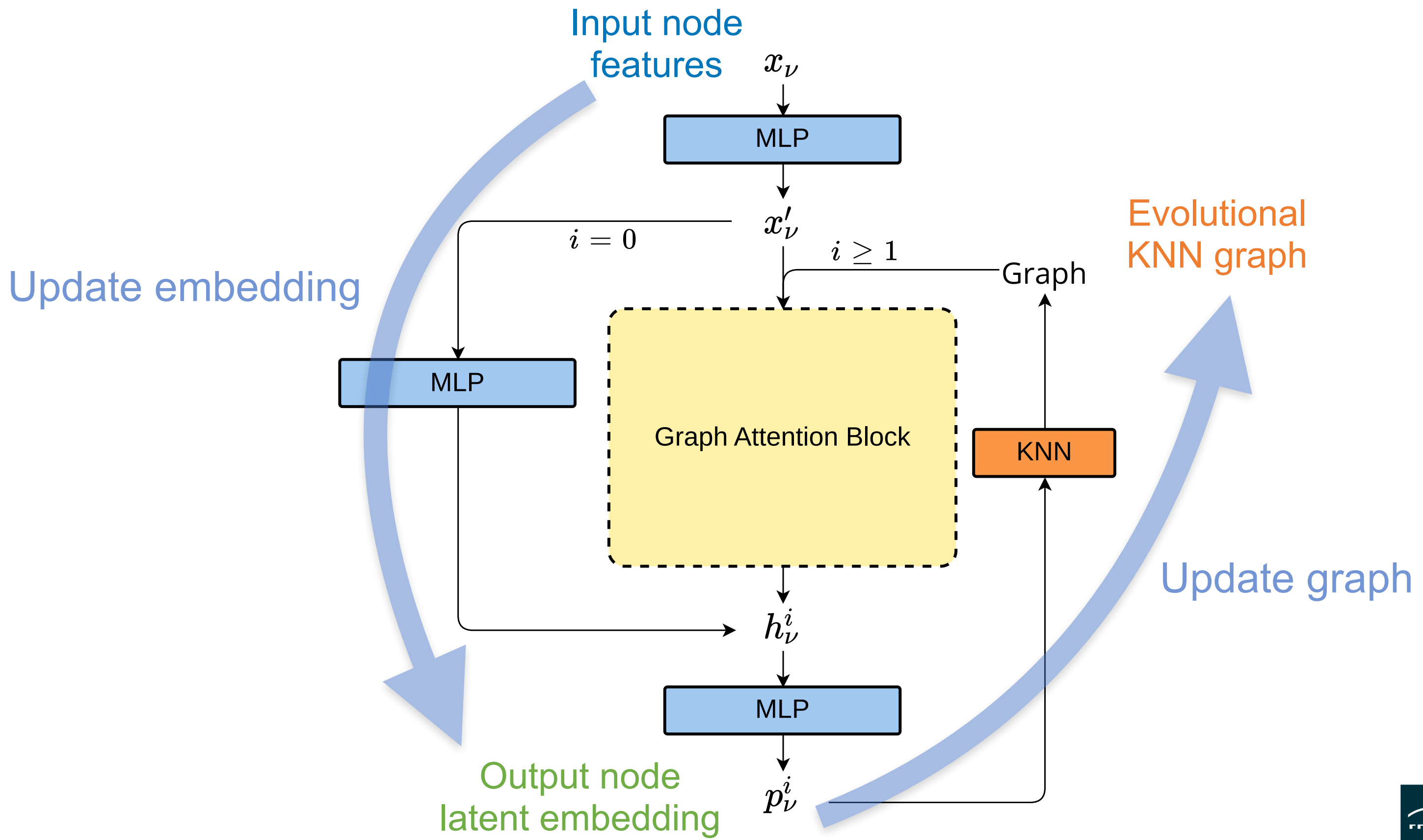
Recursive Graph Attention Network



Graph attention block consists of a series of graph-attention-style message passings

BERKELEY LAB

Recursive Graph Attention Network



Training loss function

for each pair of nodes (edge):

$$L = \underbrace{y\, d^2}_{\text{Attractive loss}} + \underbrace{(1-y)\, max^2(0, m-d)}_{\text{Repulsive loss}}$$

Attractive loss for positive pair y = 1
(hits come from the same particle)

Repulsive loss for negative pair y = 0
(hits come from different particles)

d = Euclidean distance between two hits

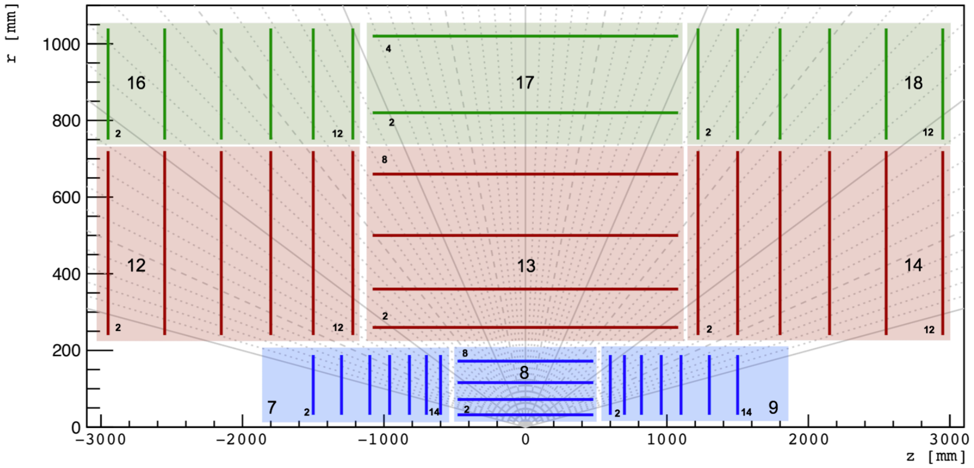
$$L_{\text{tot}} = \sum_{e_{\text{signal}}} L(e_{\text{signal}}) + \sum_{e_{\text{random}}} L(e_{\text{random}}) + \sum_{e_{\text{KNN}}} L(e_{\text{KNN}})$$

Signal edges (hits from same particles)

Random edges (randomly select 2 hits)

KNN edges for "hard negative mining"

Test case with TrackML dataset

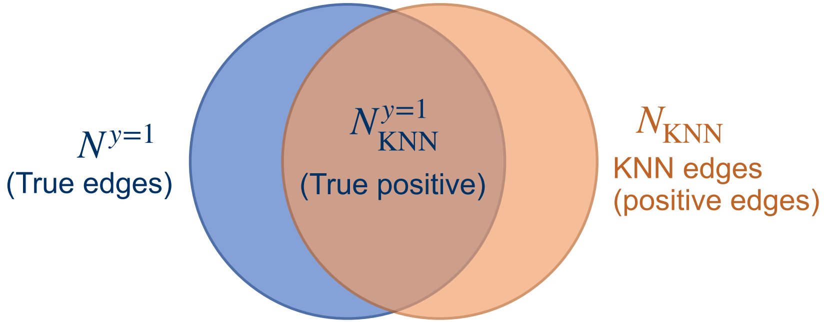


- Formulated in the [Kaggle TrackML challenge]() (HL-LHC like detector)
- Each event ~O(10^4) particles; ~O(10^5) hits
- For proof of concept, apply a cut on p_T = 1 GeV for all particles
  - → ~O(10^4) hits

BERKELEY LAB

Edge-wise Performance in KNN Graphs

Performance evaluated on *KNN graphs* to decouple effect from clustering



$$\mathrm{Eff}_{\mathrm{KNN}} = \frac{N_{\mathrm{KNN}}^{y=1}}{N^{y=1}}$$

$$\mathrm{Pur}_{\mathrm{KNN}} = \frac{N_{\mathrm{KNN}}^{y=1}}{N_{\mathrm{KNN}}} = \frac{N_{\mathrm{KNN}}^{y=1}}{N_{\mathrm{hits}} \cdot k}$$

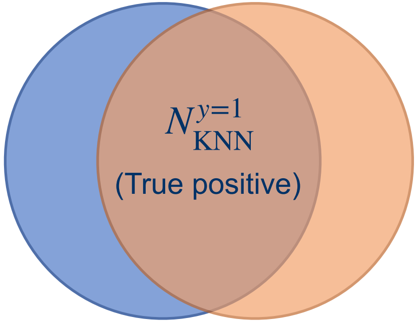
$$N_{\mathrm{KNN}}^{y=1} \leq \boxed{\sum_{\mathrm{node}^i} min(k,\, n_{\mathrm{hits}}^i - 1)}$$

Determines the **upper bounds** of $\mathrm{Eff}_{\mathrm{KNN}}$ and $\mathrm{Pur}_{\mathrm{KNN}}$

Edge-wise Performance in KNN Graphs

Performance evaluated on *KNN graphs* to decouple effect from clustering

$N^{y=1}$
(True edges)



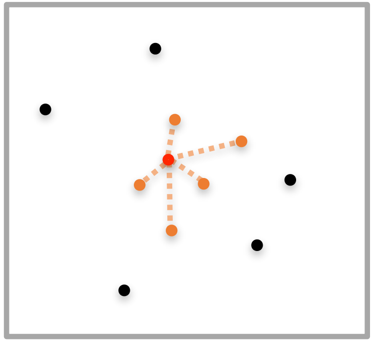
N_{KNN}
KNN edges
(positive edges)

$$\mathrm{Eff}_{\mathrm{KNN}} = \frac{N^{y=1}_{\mathrm{KNN}}}{N^{y=1}}$$

$$\mathrm{Pur}_{\mathrm{KNN}} = \frac{N^{y=1}_{\mathrm{KNN}}}{N_{\mathrm{KNN}}} = \frac{N^{y=1}_{\mathrm{KNN}}}{N_{\mathrm{hits}} \cdot k}$$

$$N^{y=1}_{\mathrm{KNN}} \leq \boxed{\sum_{\mathrm{node}^i} min(k,\, n^i_{\mathrm{hits}} - 1)}$$

Determines the **upper bounds** of $\mathrm{Eff}_{\mathrm{KNN}}$ and $\mathrm{Pur}_{\mathrm{KNN}}$



E.g.

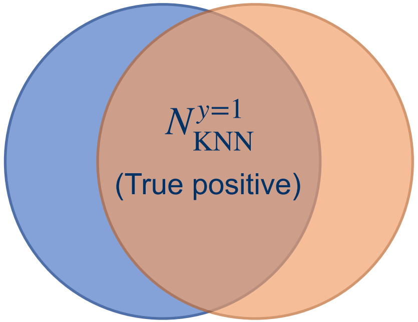
nhits = 6
$N^{y=1}$ = 5

BERKELEY LAB

Edge-wise Performance in KNN Graphs

Performance evaluated on *KNN graphs* to decouple effect from clustering

$N^{y=1}$
(True edges)



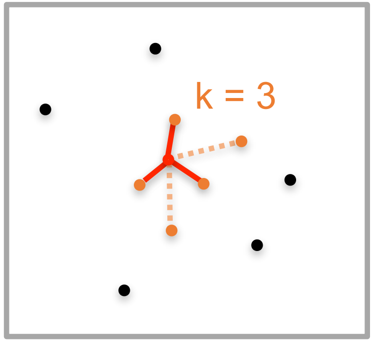
N_{KNN}
KNN edges
(positive edges)

$$\mathrm{Eff}_{\mathrm{KNN}} = \frac{N^{y=1}_{\mathrm{KNN}}}{N^{y=1}}$$

$$\mathrm{Pur}_{\mathrm{KNN}} = \frac{N^{y=1}_{\mathrm{KNN}}}{N_{\mathrm{KNN}}} = \frac{N^{y=1}_{\mathrm{KNN}}}{N_{\mathrm{hits}} \cdot k}$$

$$N^{y=1}_{\mathrm{KNN}} \leq \boxed{\sum_{\mathrm{node}^i} min(k,\, n^i_{\mathrm{hits}} - 1)}$$

Determines the **upper bounds** of $\mathrm{Eff}_{\mathrm{KNN}}$ and $\mathrm{Pur}_{\mathrm{KNN}}$



E.g.

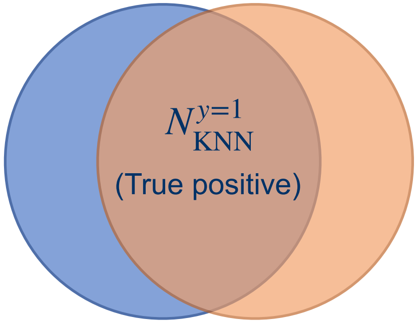
nhits = 6
$N^{y=1}$ = 5

Eff = 3/5 = 0.6
Pur = 3/3 = 1.0

Edge-wise Performance in KNN Graphs

Performance evaluated on *KNN graphs* to decouple effect from clustering

$N^{y=1}$
(True edges)



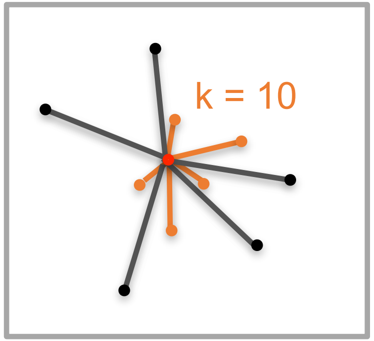
N_{KNN}
KNN edges
(positive edges)

$$\text{Eff}_{\text{KNN}} = \frac{N^{y=1}_{\text{KNN}}}{N^{y=1}}$$

$$\text{Pur}_{\text{KNN}} = \frac{N^{y=1}_{\text{KNN}}}{N_{\text{KNN}}} = \frac{N^{y=1}_{\text{KNN}}}{N_{\text{hits}} \cdot k}$$

$$N^{y=1}_{\text{KNN}} \leq \boxed{\sum_{\text{node}^i} min(k,\, n^i_{\text{hits}} - 1)}$$

Determines the **upper bounds** of Eff_{KNN} and Pur_{KNN}

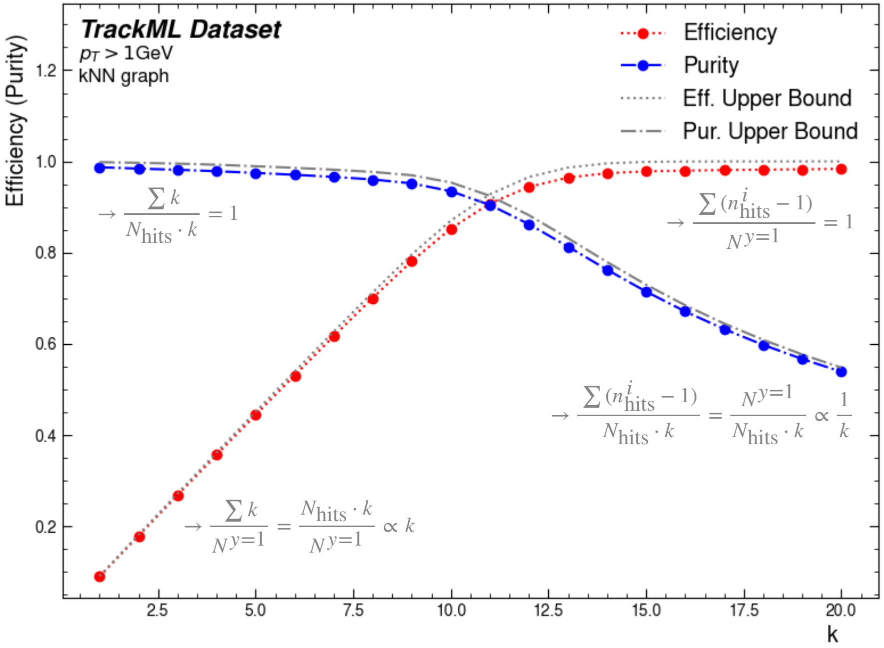


E.g.

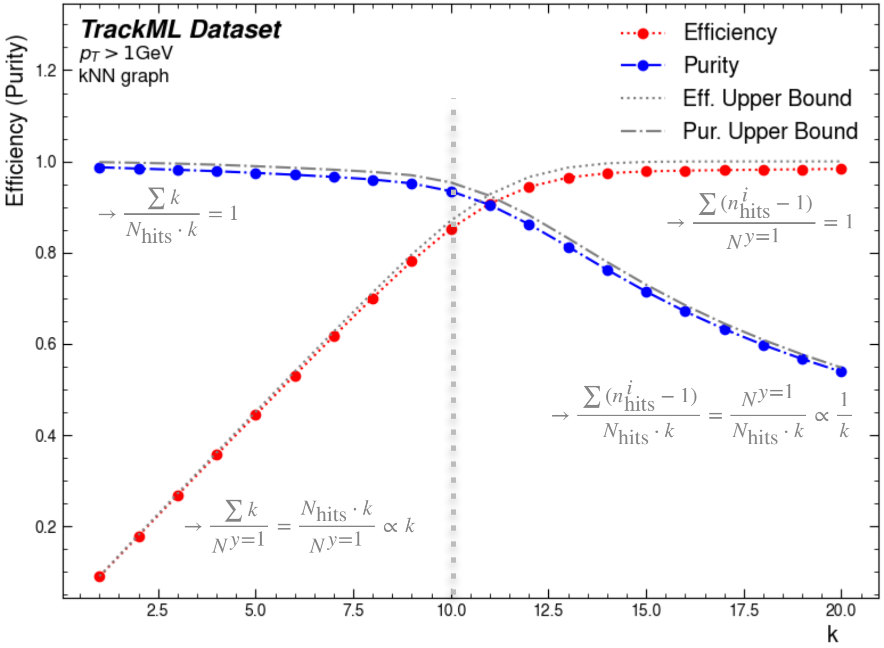
nhits = 6
$N^{y=1} = 5$

Eff = 5/5 = 1.0
Pur = 5/10 = 0.5

Edge-wise Performance in KNN Graphs (vs k)

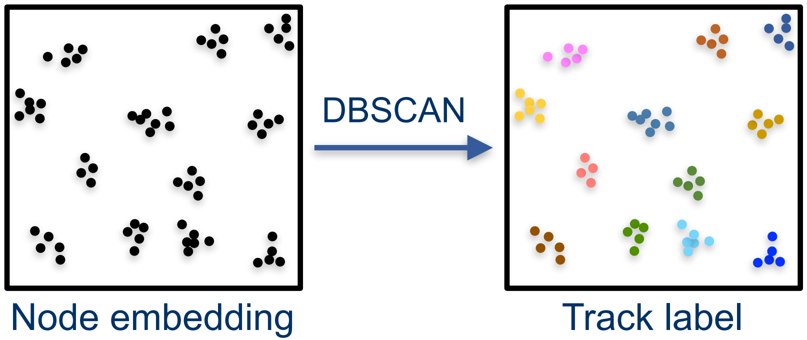


Edge-wise Performance in KNN Graphs (vs k)



At k=10:
Eff = 85% (upper bound = 87%)
Pur = 93% (upper bound = 95%)

DBSCAN and track performance

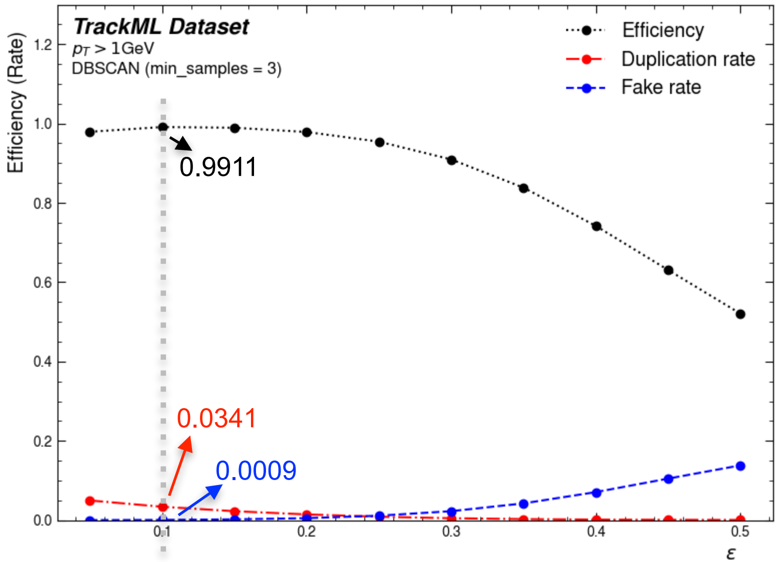


A matched track = (>50% hits in this track candidate come from same particle)

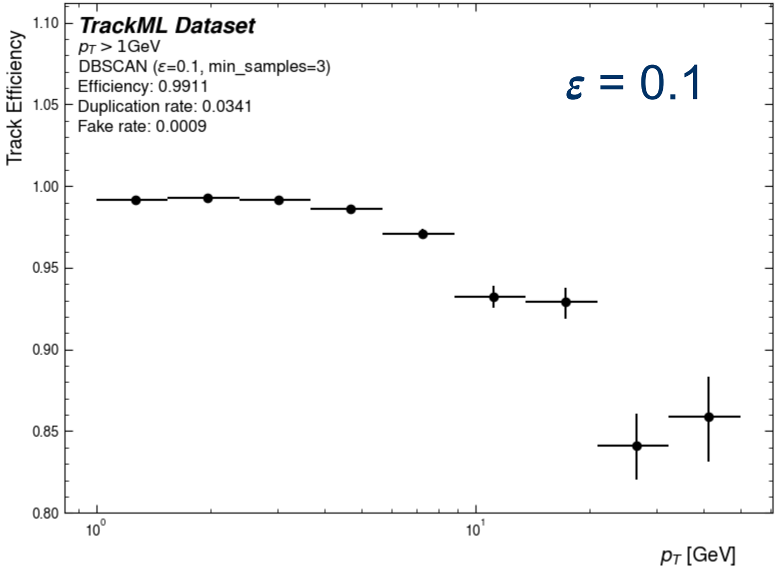
$$\mathrm{Eff}_{\mathrm{track}} = \frac{N^{\mathrm{reco}}_{\mathrm{particles}}}{N_{\mathrm{particles}}} \qquad r_{\mathrm{fake}} = \frac{N_{\mathrm{tracks}} - N^{\mathrm{matched}}_{\mathrm{tracks}}}{N^{\mathrm{reco}}_{\mathrm{particles}}} \qquad r_{\mathrm{duplicate}} = \frac{N^{\mathrm{matched}}_{\mathrm{tracks}} - N^{\mathrm{reco}}_{\mathrm{particles}}}{N^{\mathrm{reco}}_{\mathrm{particles}}}$$

DBSCAN track performance

Track performance vs ε (DBSCAN)

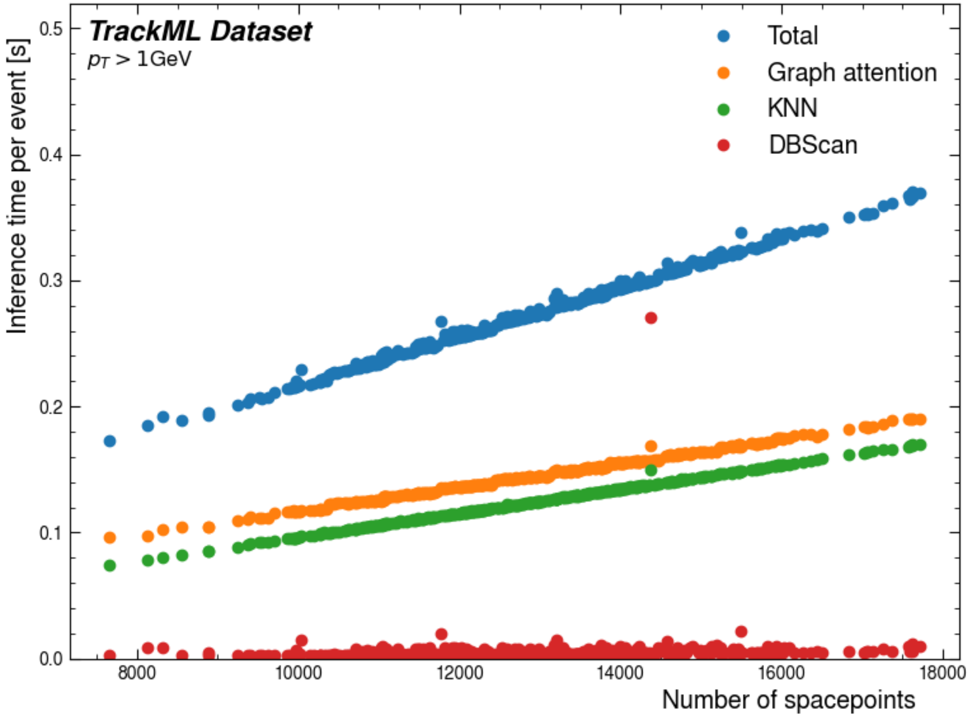


Track efficiency vs p_T



BERKELEY LAB

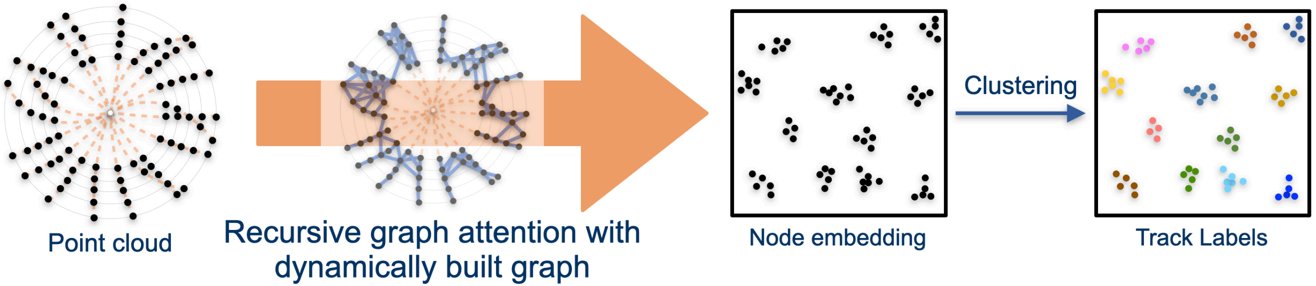
Computing performance



Computational cost mainly coming from graph attention and KNN

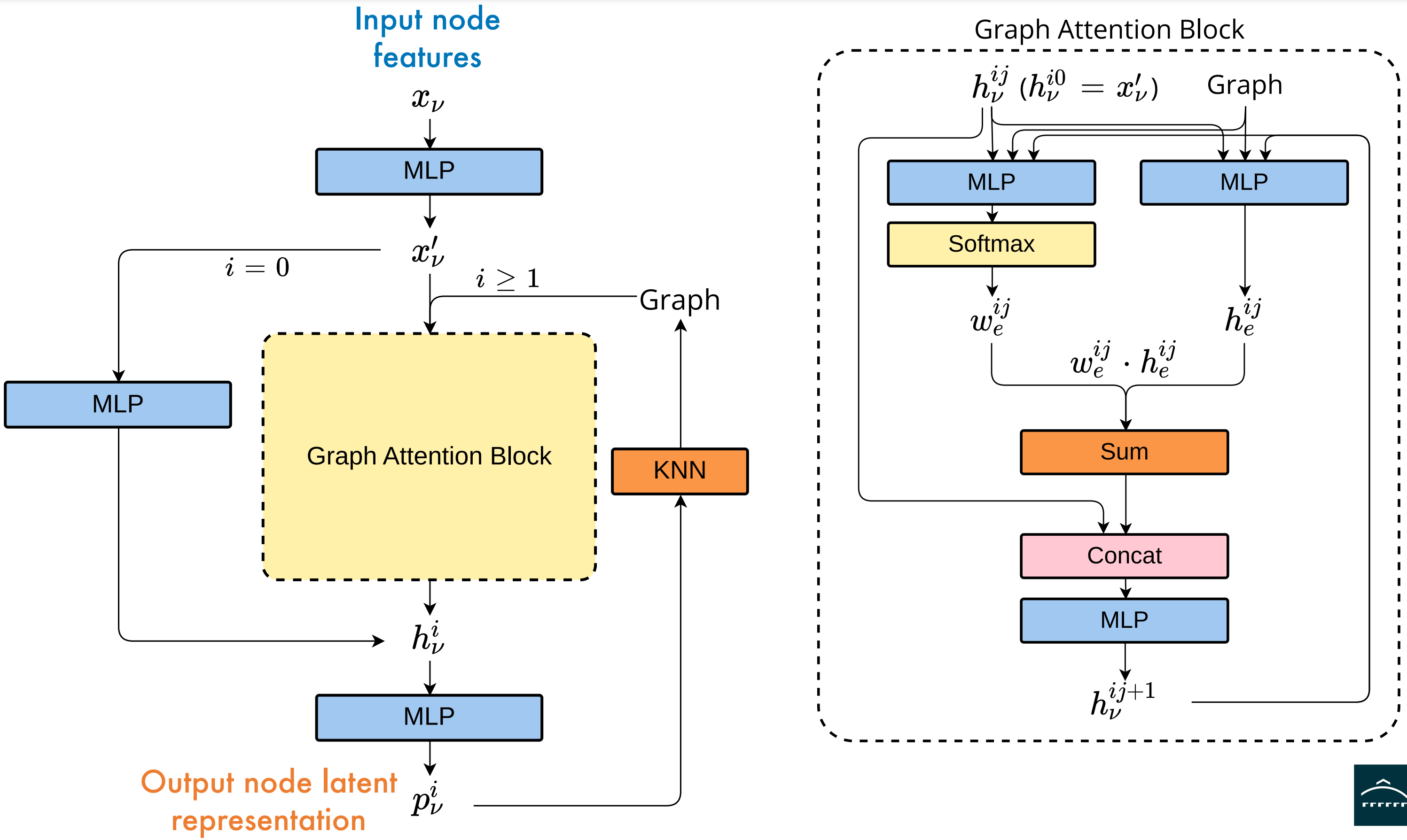
Summary

- Propose a one-shot object-condensation tracking algorithm with recursive graph attention
  - Does not require graph construction as the first step (take point cloud as input for message passing)
  - Achieve excellent track performance in the TrackML test case
- Future work aims to improve computational cost: main contribution from KNN and graph attention

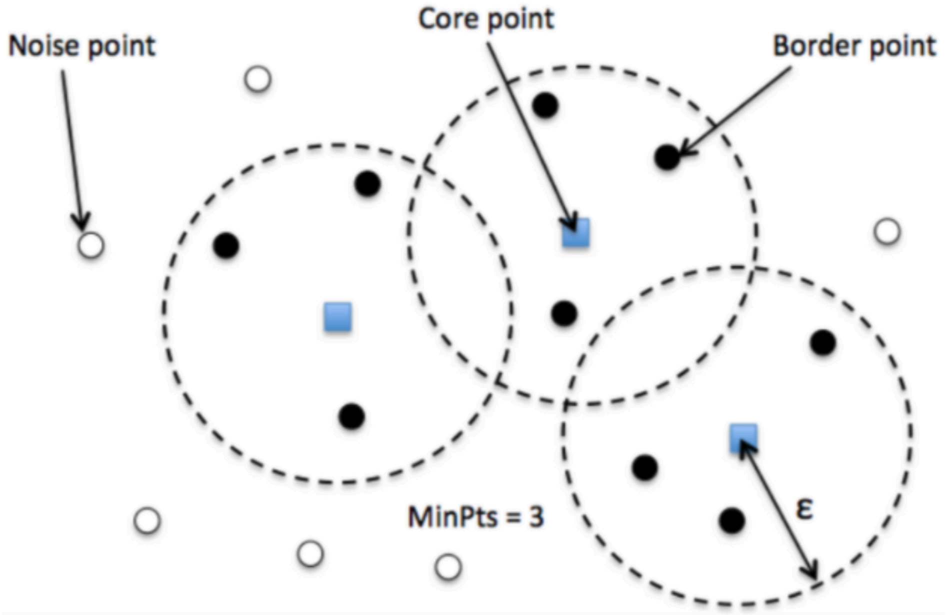


Backups

Recursive Graph Attention Network



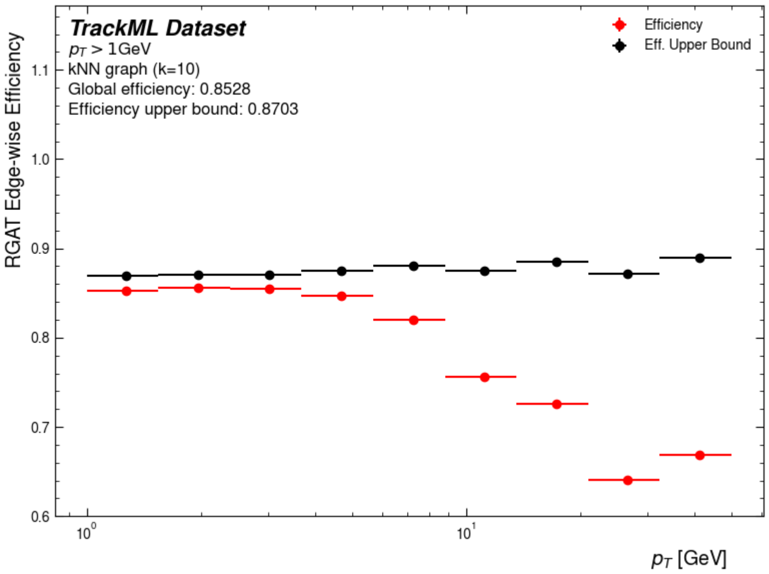
Density-Based Spatial Clustering of Applications with Noise



Idea: a cluster in data space is a contiguous region of high point density, separated from other such clusters by contiguous regions of low point density

Edge-wise Performance in KNN Graphs (vs p_T)

Eff_{KNN}



Pur_{KNN}