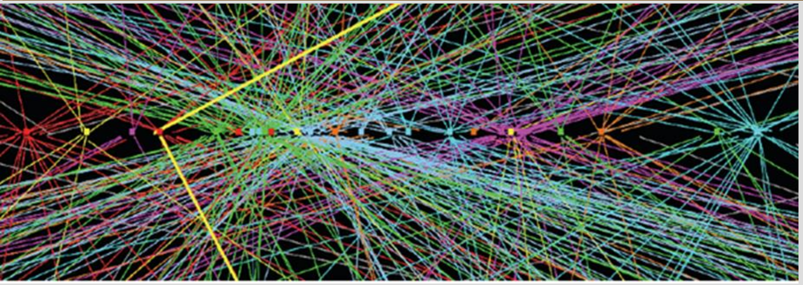
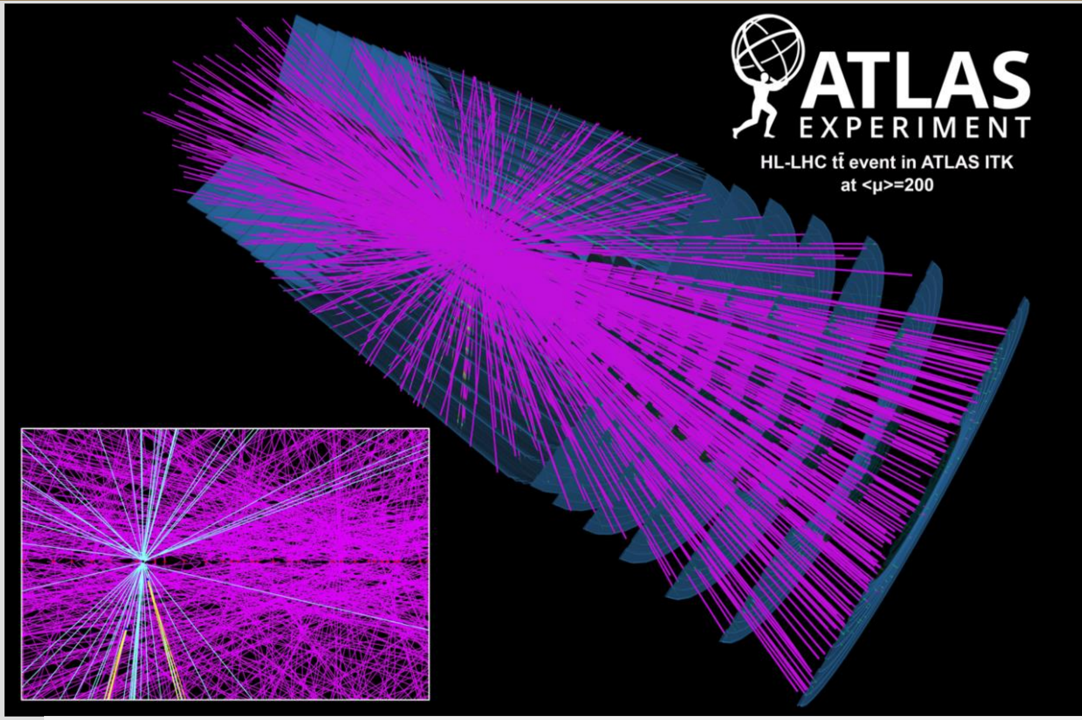
Inclusive vertex reconstruction using ML approach

M. Cristinziani, V. Kostyukhin
Siegen University

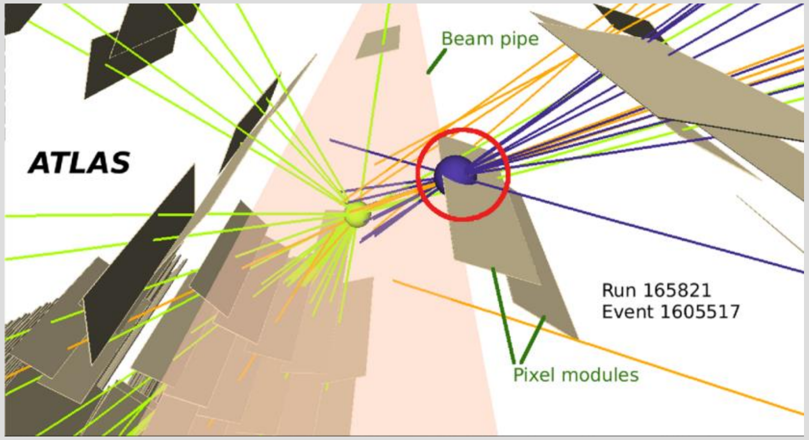
Some examples



A visual example of pile-up in the ATLAS tracker
Proc.Comp.Science v66 (2015) 540-545



A simulated tt̄bar event at average pile-up of 200 collisions per bunch crossing, with an ITk layout including the very forward extension. The bottom-left inset is a 2D r-z view of the interaction region. The vertical scale is 2.5mm and the horizontal one 12cm. All reconstructed tracks have pT>1 GeV. The tracks coming from the ttbar vertex are coloured in cyan. Two secondary vertices can be reconstructed and the tracks coming from them are highlighted in yellow.



An event from a jet-trigger data sample, where a high-mass vertex (circled) is the result of an apparently random, large-angle intersection between a track and alow-mass hadronic-interaction vertex produced in a pixel module. Tracks originating from this vertex are shown in blue, those from the primary vertex are green, and other tracks are orrange. The beampipe and pixel modules with track hits are shown.

Currently used various ad hoc identification/reconstruction algorithms for different vertex classes – PV, SV, b-tagging, etc.

Vertexing problem definition

Given a set of reconstructed tracks, one needs to find all physics vertices in it.

A physics-motivated way to solve this problem is to construct a track compatibility (adjacency) graph and to partition it into a collection of isolated, non-overlapping clusters. Each cluster represents a vertex, the parameters of which can be computed from the assigned tracks.

Challenges

- ✧ A priory unknown (big) amount of truth vertices with strongly different track multiplicity ([2,∞]);
- ✧ Order of magnitude difference in reconstruction accuracy of the tracks;
- ✧ Track-track distances comparable with the vertex-vertex distances;
- ✧ High density of the tracks and vertices and big track position reconstruction errors result in a strong overlap of the tracks from different true vertices;

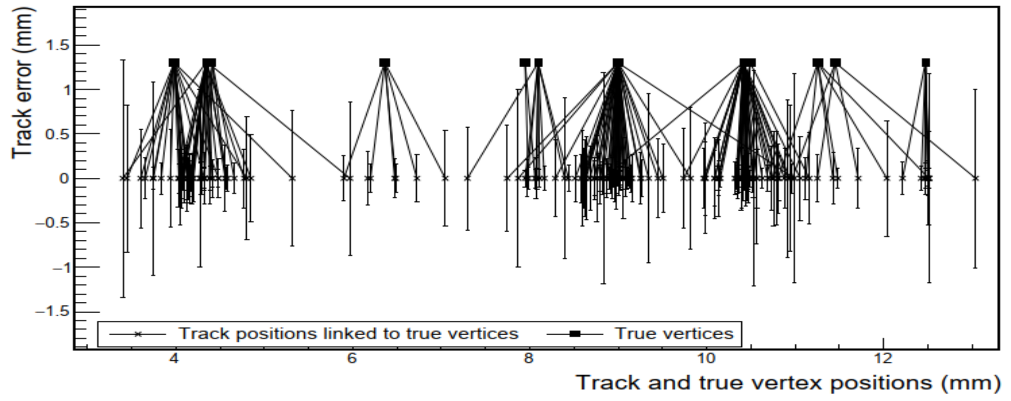


Figure 3. Example display of overlapping tracks from different vertices caused by measurement errors (zoom of a simulated DELPHES event with $\mu = 150$). The crosses at the ordinate value of 0 represent the track positions, and the vertical error bars represent the corresponding position measurement errors. Squares at ordinate values of 1.3 represent the truth vertex positions. The connecting lines show the origin vertex for every track.

Pattern recognition/clustering problem without an exact solution => dedicated performance metrics.

Primary vertices (1D space) is addressed in JINST 18 (2023) P07013 - joint work with M.Keuper from Computing Vision dept. (now in Mannheim)

Minimum-Cost Multicut

The *Minimum-Cost Multicut* problem definition in arXiv:1505.06973 :

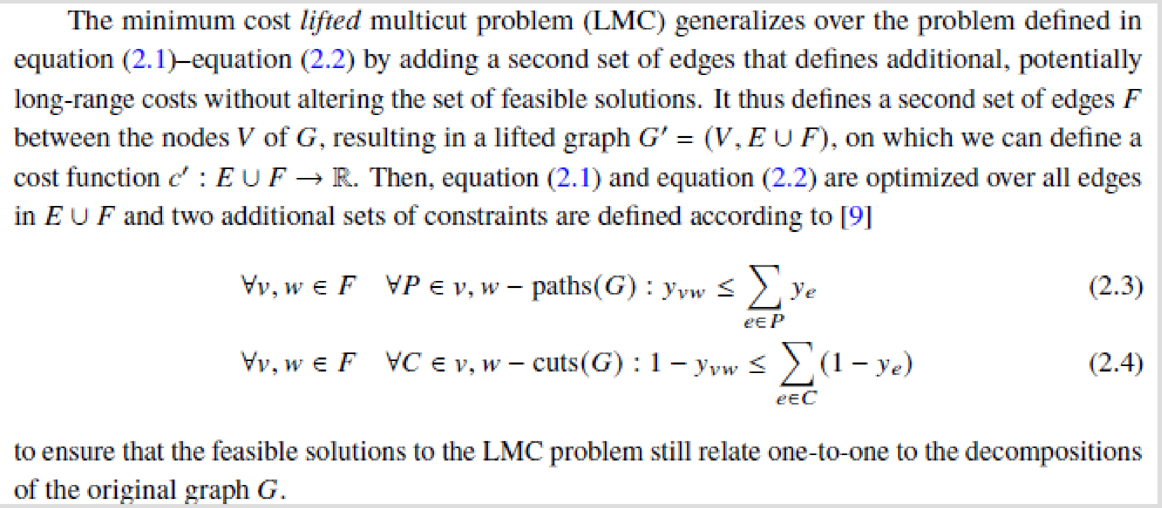
The minimum cost multicut problem is a grouping problem defined for a graph $G = (V, E)$ and a cost function $c : E \to \mathrm{R}$ which assigns to all edges $e \in E$ a real-valued cost or reward for being cut. Then, the minimum cost multicut problem is to find a binary edge labelling y according to

$$\min_{y \in \{0,1\}^E} \sum_{e \in E} c_e y_e \tag{2.1}$$

subject to

$$\forall C \in \mathrm{cycles}(G) \quad \forall e \in C : y_e \leq \sum_{e' \in C \setminus \{e\}} y_{e'} \,. \tag{2.2}$$

Trivial optimal solutions are avoided by assigning positive (attractive) costs c_e to edges between nodes $v, w \in V$ that likely belong to the same component, while negative (repulsive) costs are assigned to edges that likely belong to different components.



The minimum cost *lifted* multicut problem (LMC) generalizes over the problem defined in equation (2.1)–equation (2.2) by adding a second set of edges that defines additional, potentially long-range costs without altering the set of feasible solutions. It thus defines a second set of edges F between the nodes V of G, resulting in a lifted graph $G' = (V, E \cup F)$, on which we can define a cost function $c' : E \cup F \to \mathbb{R}$. Then, equation (2.1) and equation (2.2) are optimized over all edges in $E \cup F$ and two additional sets of constraints are defined according to [9]

$$\forall v, w \in F \quad \forall P \in v, w - \mathrm{paths}(G) : y_{vw} \leq \sum_{e \in P} y_e \tag{2.3}$$

$$\forall v, w \in F \quad \forall C \in v, w - \mathrm{cuts}(G) : 1 - y_{vw} \leq \sum_{e \in C} (1 - y_e) \tag{2.4}$$

to ensure that the feasible solutions to the LMC problem still relate one-to-one to the decompositions of the original graph G.

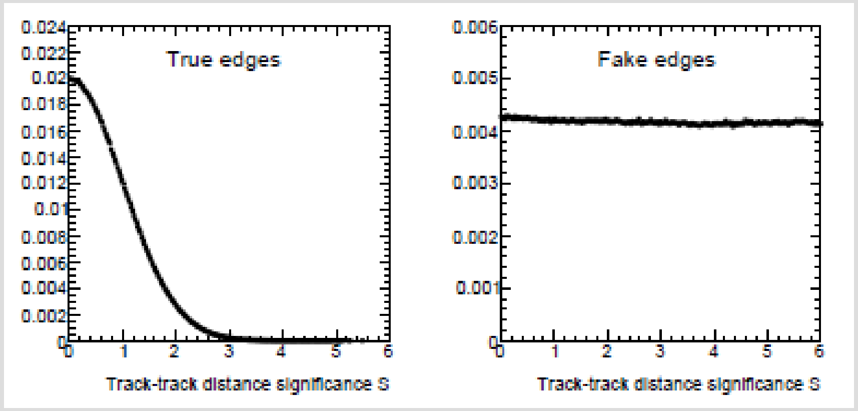
For the vertex-finding problem, this formulation allows encoding Euclidean distance constraints in the structure of graph G (e.g. point observations that are spatially distant cannot originate from the same vertex), while the cost function can be naturally defined in the distance significance space to account for measurement errors. The lifted multicut approach encodes both metrics in the same graph.

Edge score options (1D space)

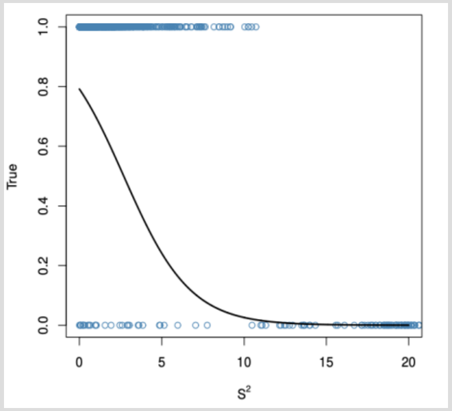
The edge weights are to be negative for edges that should be cut and positive for those connecting nodes that should be joined.

1. Probability distribution ratio $w = \log(\frac{p_{true}}{p_{false}})$ of the minimal track-track distance significance $S = \sqrt{\frac{(x_i - x_j)^2}{\sigma_i^2 + \sigma_j^2}}$;
2. Logistic regression $p = \frac{1}{1+e^{-z}}$ where $z = \beta_0 + \beta_1 \cdot S$. Then $w = \log(\frac{1}{1-p})$ weight has necessary features.
3. BDT edge classification (7 variables) score in [-1,1] range

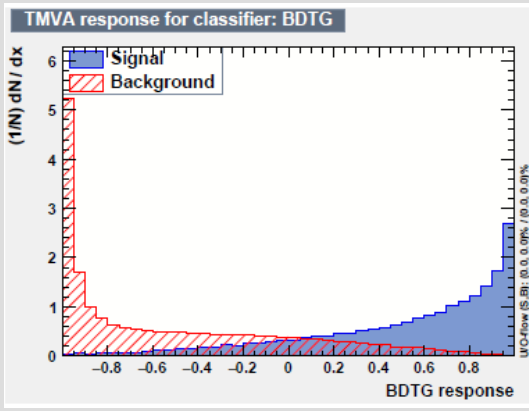
Track-track significance S



Logistic regression



BDT classification

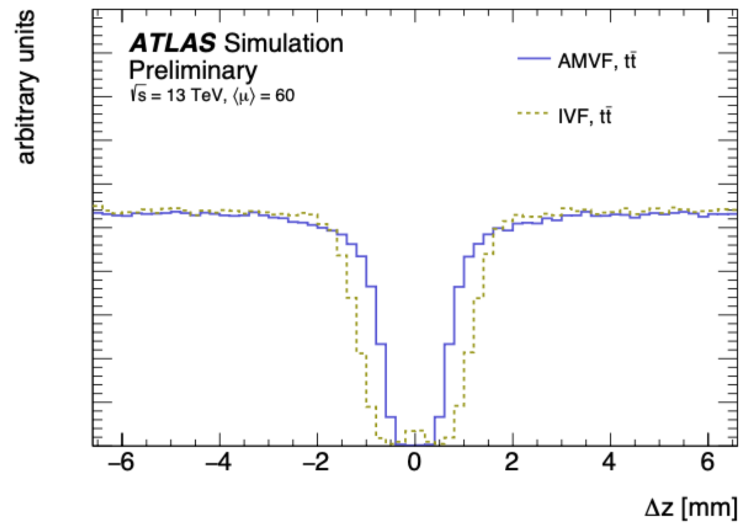


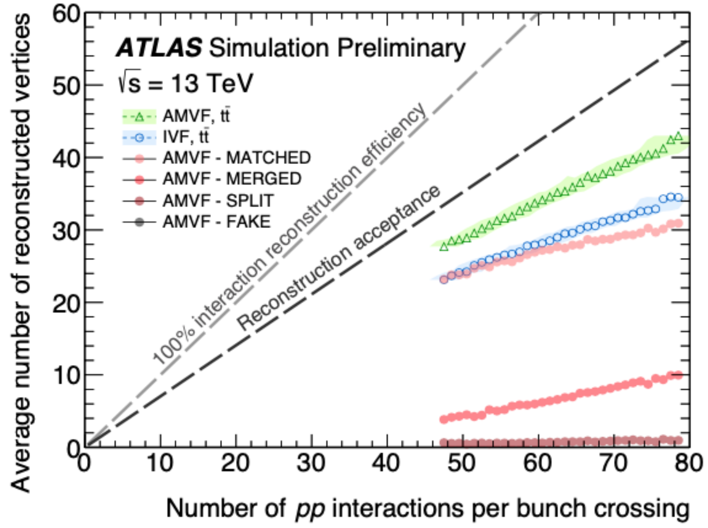
μ=250 pileup reconstruction results

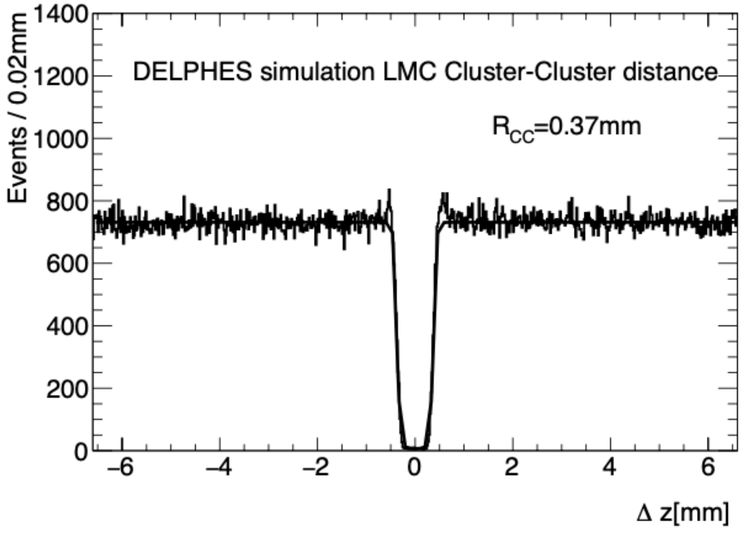
Edge weight		VI	VI weighted	Silhouette	Silhouette weighted	Unique	Merged	Fake	N_{trk}^{wrong}	CPU
PDF ratio	base	1.782	0.990	0.477	0.526	68.7	53.2	6.4	42%	3.0s
	cnst	1.638	0.887	0.531	0.569	71.0	52.7	5.3	21%	1.7s
Regression	base	1.753	0.961	0.467	0.517	77.1	51.2	11.	38%	3.2s
	cnst	1.672	0.895	0.505	0.547	77.8	51.1	9.9	21%	1.7s
BDT	base	1.691	0.941	0.307	0.040	72.8	52.4	15.	12%	3.0s
	cnst	1.651	0.882	0.330	0.055	74.5	52.0	14.	9%	1.2s

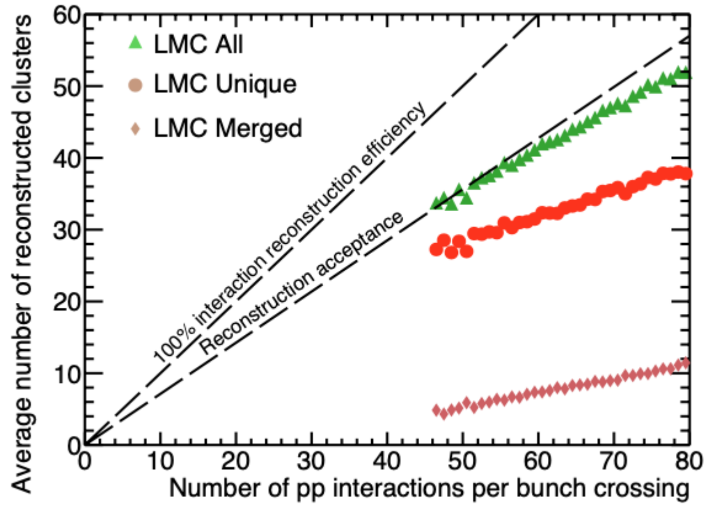
1D performance: Comparison

From JINST 18 (2023) P07013









Next steps

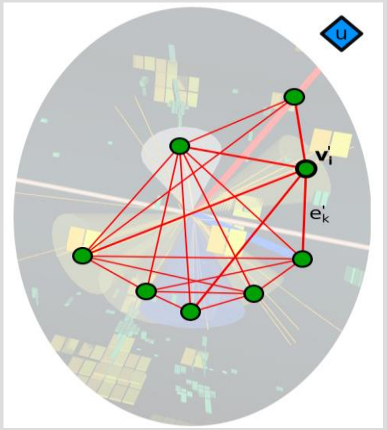
Switch to 3D space <-> simultaneous reconstruction of primary + secondary vertices (V^0s, B/C hadrons, hadronic interactions, conversions, etc.) with a possibility to look for exotic LLPs(no a priory information).
Challenging benchmark – FCChh environment (~1000 PVs, ~10^4 tracks).
Main problem – construction of an efficient adjacency graph, 1D-type weights are insufficient.

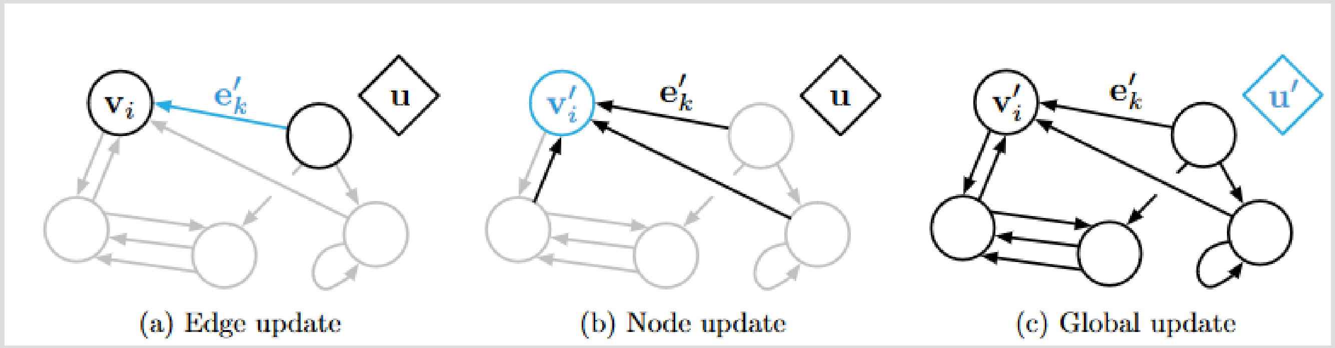
What is planned:

1) 4D (+time) tracking and vertexing. Needs implementation in the track adjacency(compatibility) edge weights assignment.
2) Massive a priory information usage – expected vertex positions (beamline, material layers, invariant masses, jets, detector hits, etc.). Can't be implemented using pairwise node relations, multi-node relations are needed. E.g. edge==2-track_vertex, how compatible/distant are 2 edges(vertices) of the same node(track)?

There is no well-defined efficient recipe for how to compile the 4D + various prior information to the track compatibility weights (probabilities) for clustering;

1) Try Graph Neural Net with hidden states, which can be trained to provide improved edge weights for the precise LMC clustering;
2) Differentiable vertex fitting in GNN to improve the accuracy of clustering (arXiv:2310.12804);
3) Optimal prior information sharing between GNN and LMC(constraints).





(a) Edge update (b) Node update (c) Global update

Full GNN solution?

arXiv:2204.01366 “Learning to solve Minimum Cost Multicuts efficiently using Edge-Weighted Graph Convolutional Neural Networks”

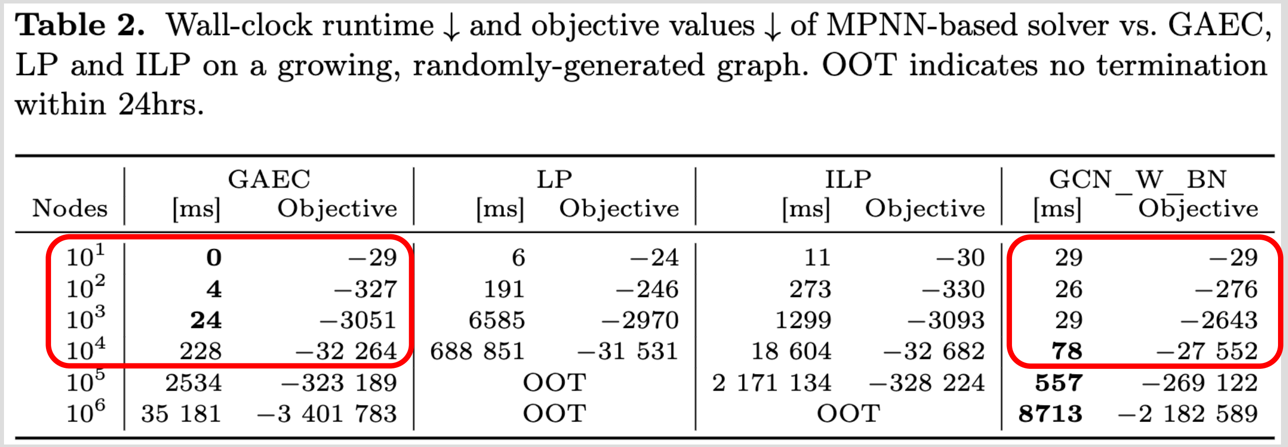


Table 2. Wall-clock runtime ↓ and objective values ↓ of MPNN-based solver vs. GAEC, LP and ILP on a growing, randomly-generated graph. OOT indicates no termination within 24hrs.

	GAEC		LP		ILP		GCN_W_BN	
Nodes	[ms]	Objective	[ms]	Objective	[ms]	Objective	[ms]	Objective
10^1	**0**	−29	6	−24	11	−30	29	−29
10^2	**4**	−327	191	−246	273	−330	26	−276
10^3	**24**	−3051	6585	−2970	1299	−3093	29	−2643
10^4	228	−32 264	688 851	−31 531	18 604	−32 682	**78**	−27 552
10^5	2534	−323 189	OOT		2 171 134	−328 224	**557**	−269 122
10^6	35 181	−3 401 783	OOT		OOT		**8713**	−2 182 589

LMP (GAEC here) is as fast as Convolutional Neural Net (GCN_W_BN here) up to ~10^4 nodes(tracks) but much more precise (lower objective).

LMC doesn’t require any prior information and training – perfect for search for unknown (LLP, etc.)

More advanced ML setup, e.g. foundation model?