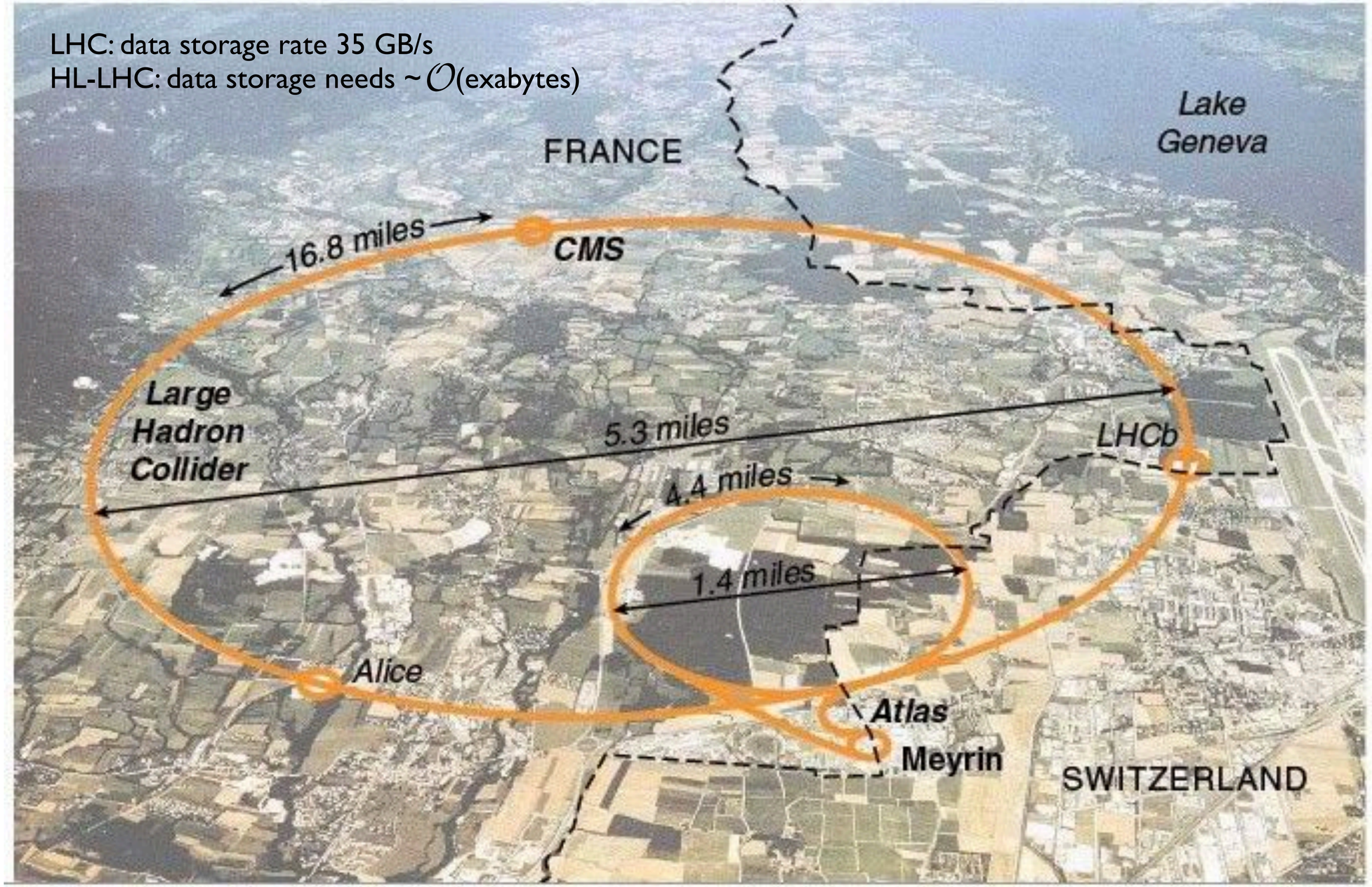
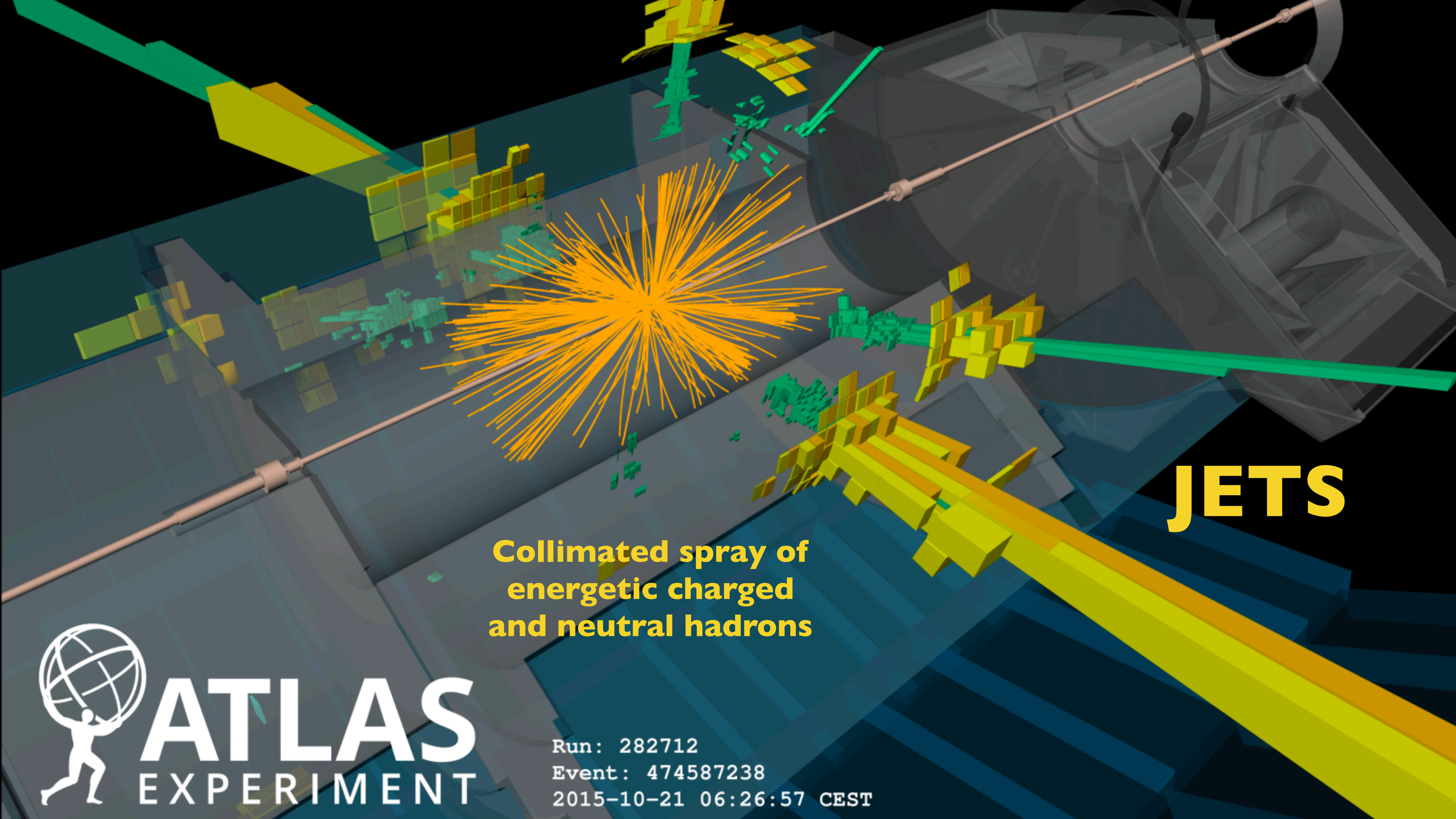
Reframing Jet Physics in Probabilistic Terms

Kyle Cranmer, **Sebastian Macaluso**
New York University

CCPP Brown Bag
March 1st, 2021

The Large Hadron Collider

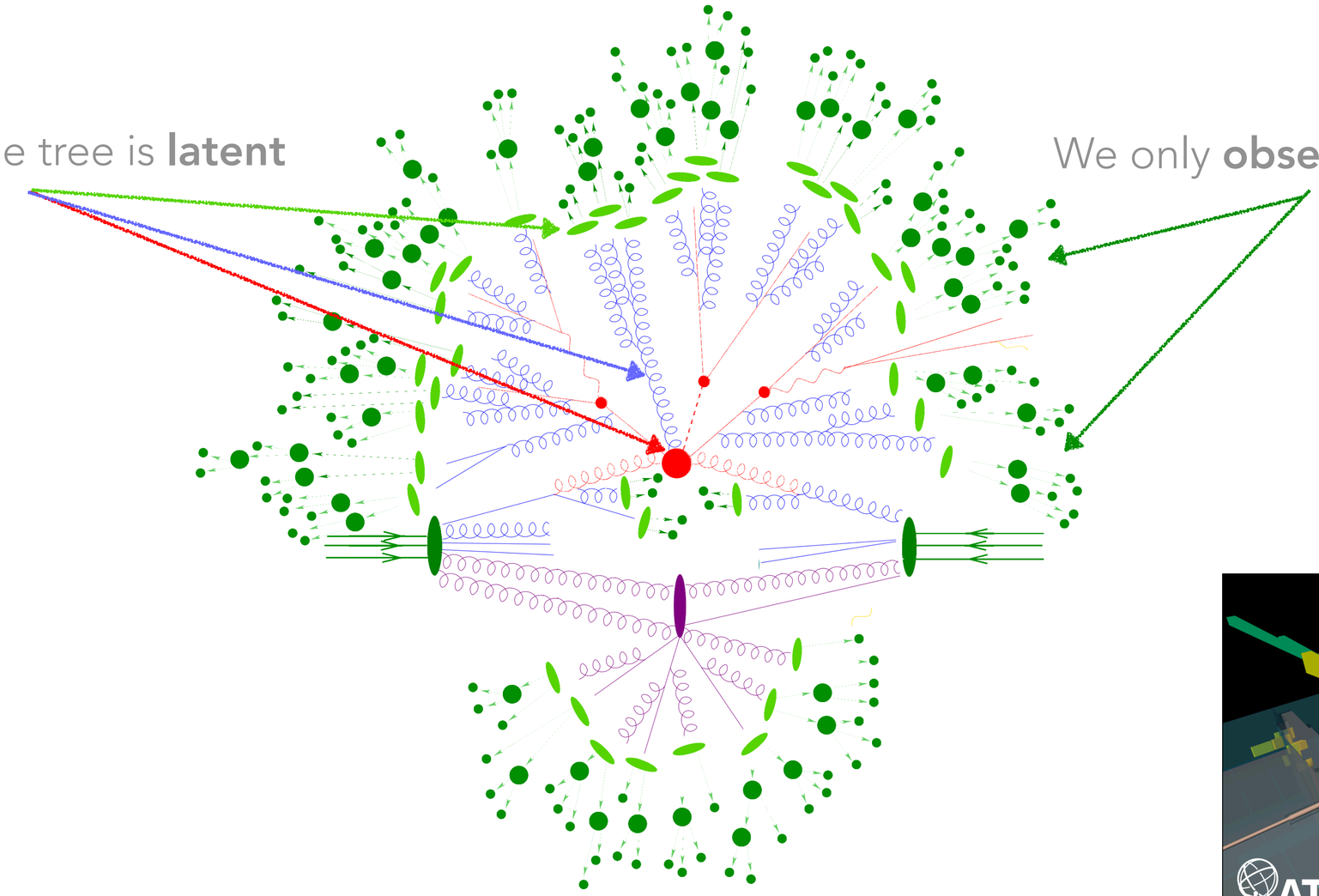


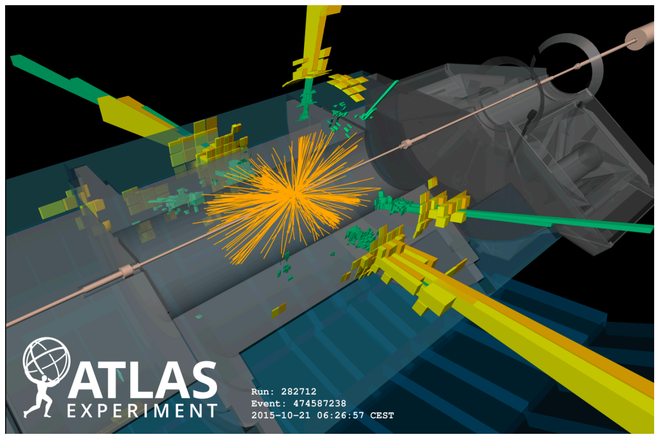

JETS
Collimated spray of
energetic charged
and neutral hadrons
ATLAS
EXPERIMENT
Run: 282712
Event: 474587238
2015-10-21 06:26:57 CEST

Hadron-hadron collision - Monte Carlo event generator

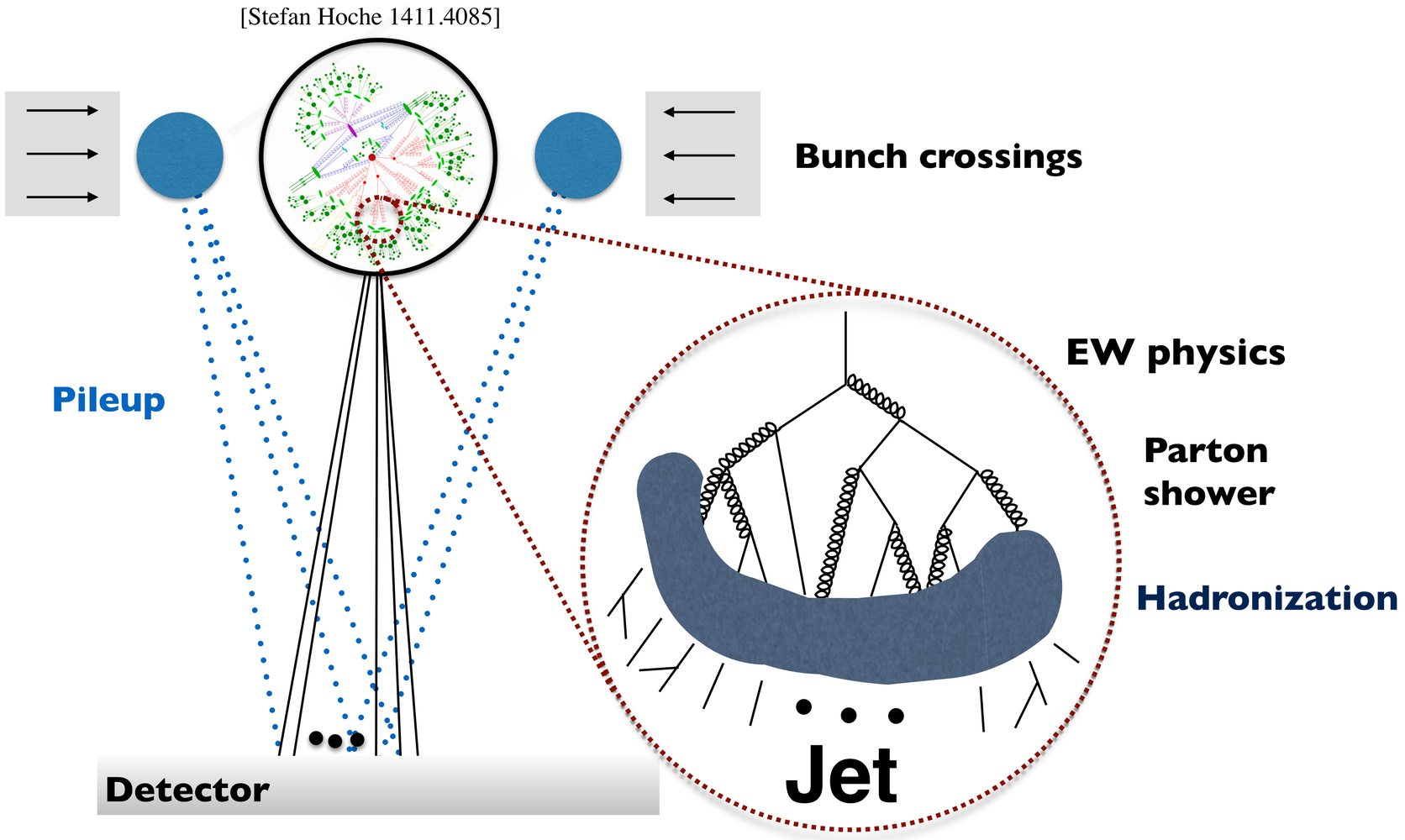
Evolution of the tree is **latent**

We only **observe** the **leaves**

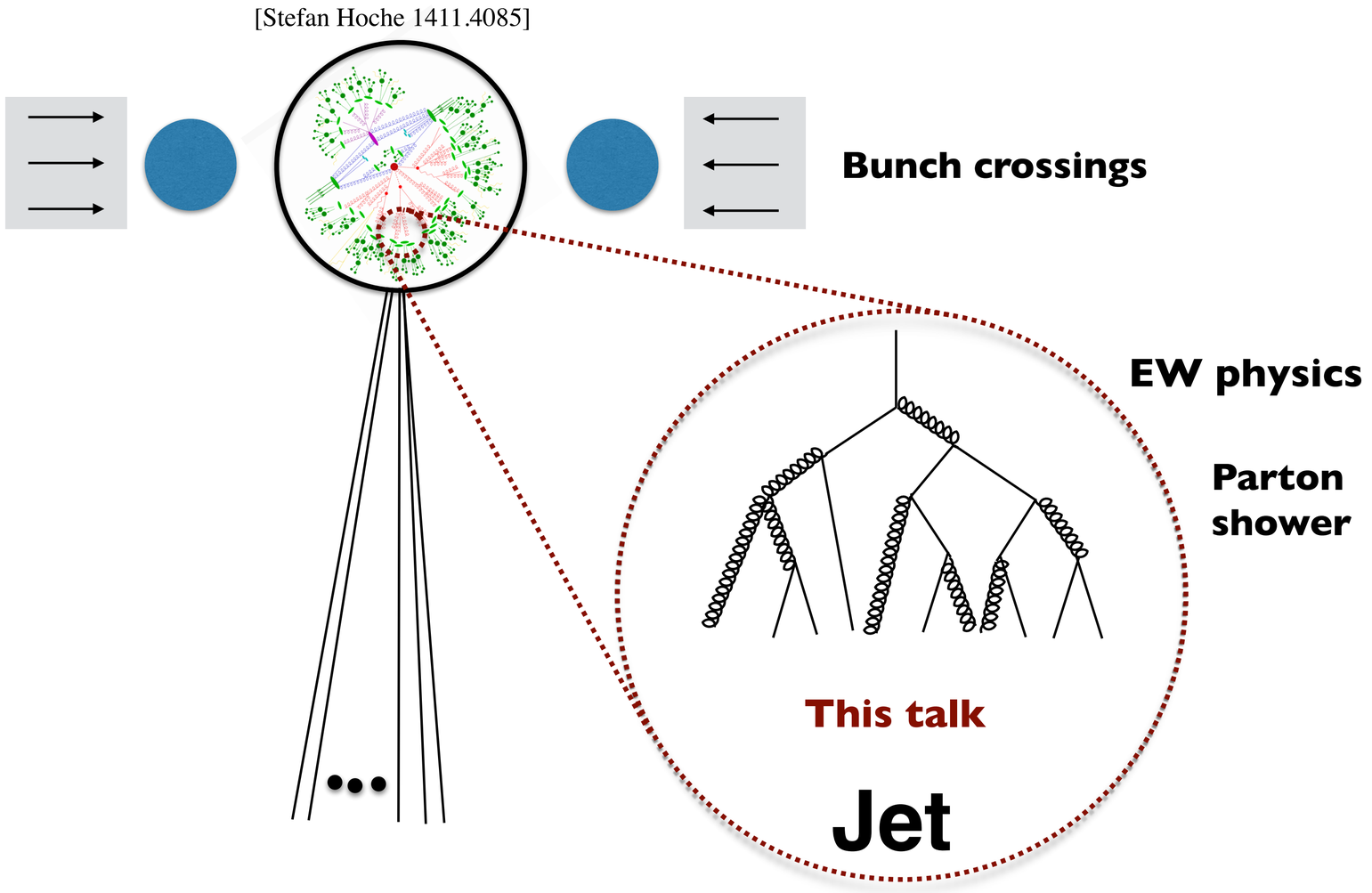




Schematic Representation

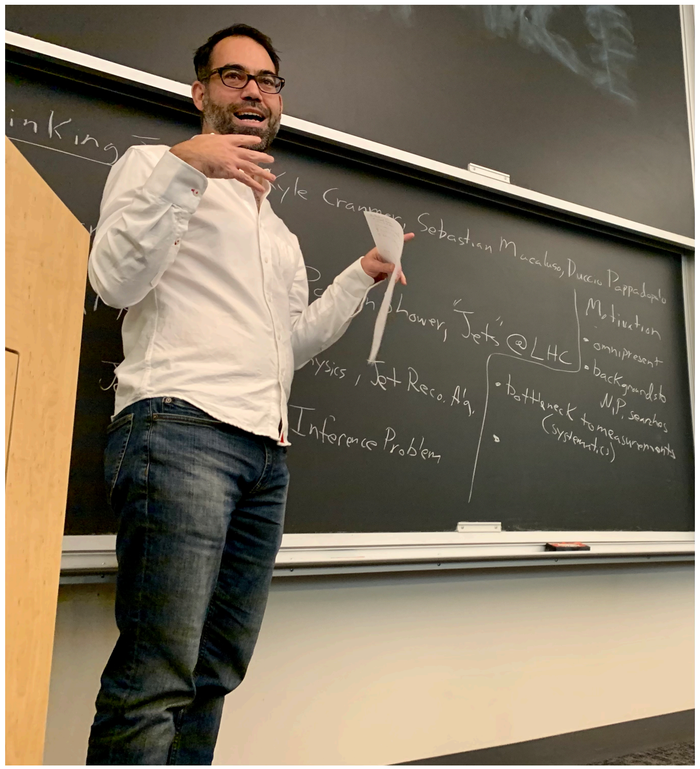


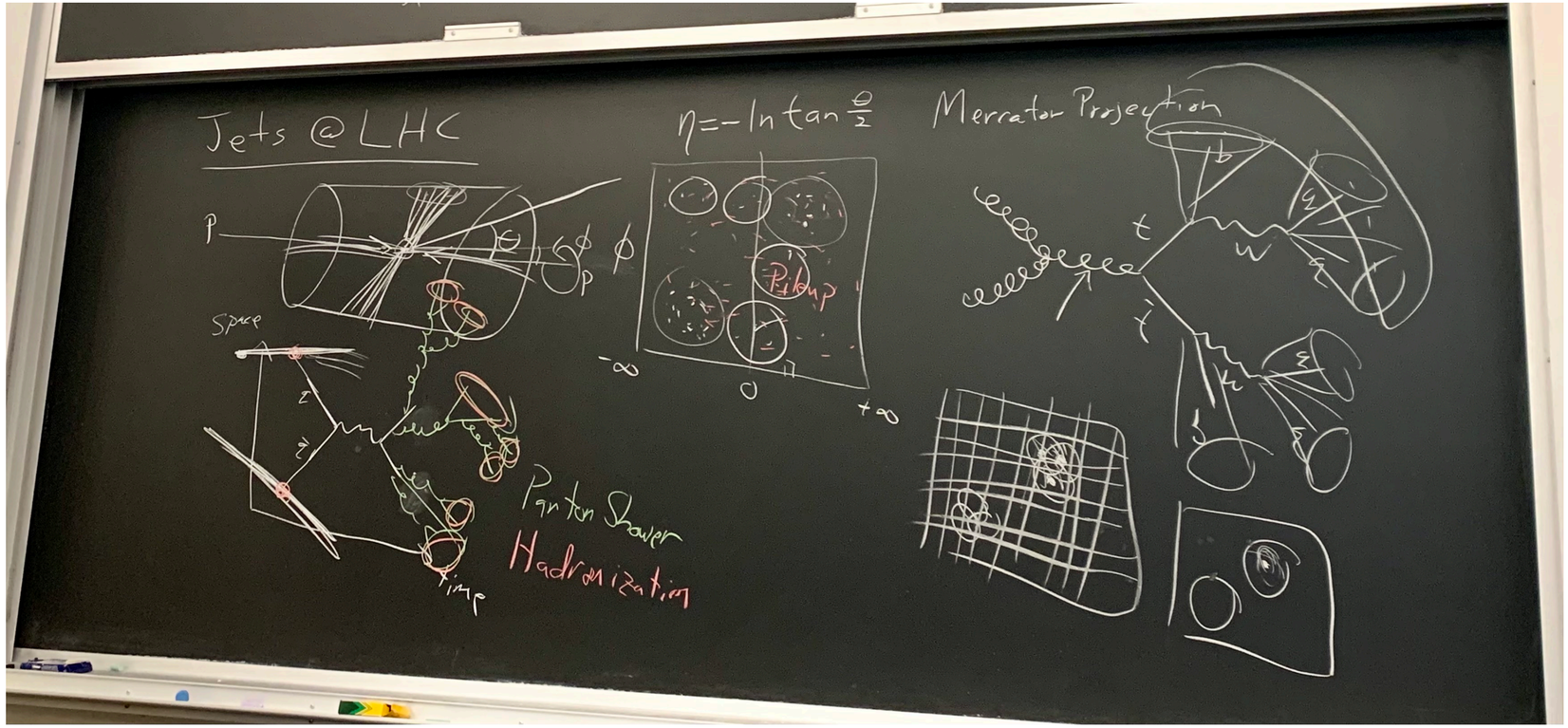
Schematic Representation



Follow up from Brown Bag talk by Kyle Cranmer on Nov. 25, 2019

Rethinking Jets



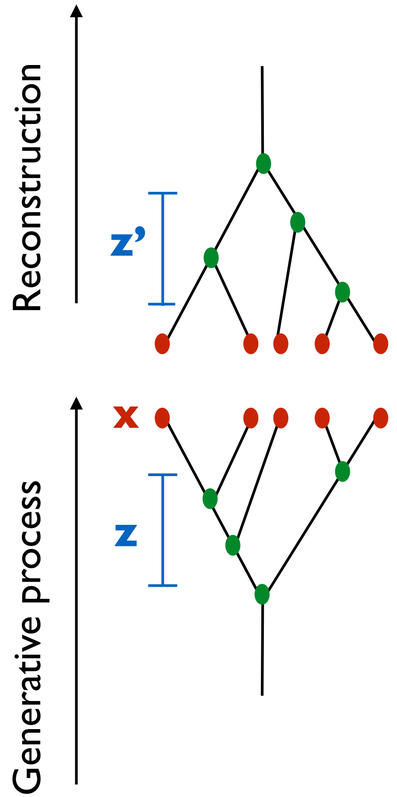


Jet Clustering

- Task: reconstruct the unobserved latent tree z from observed x.
- In more general terms we want to find the "correct" hierarchical clustering given a set of leaves (jet constituents) x.

Hierarchical Clustering
Recursive partitioning of a data set into successively smaller clusters.

- We could think of hierarchical clustering as inverting the generative process.



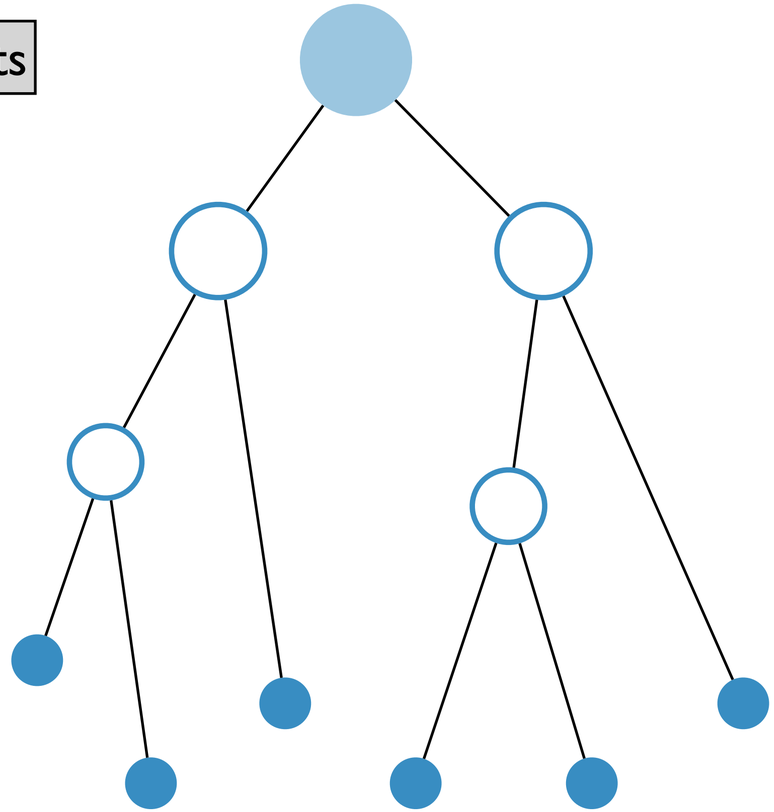
Hierarchical Clustering and Large-Scale Structure Formation

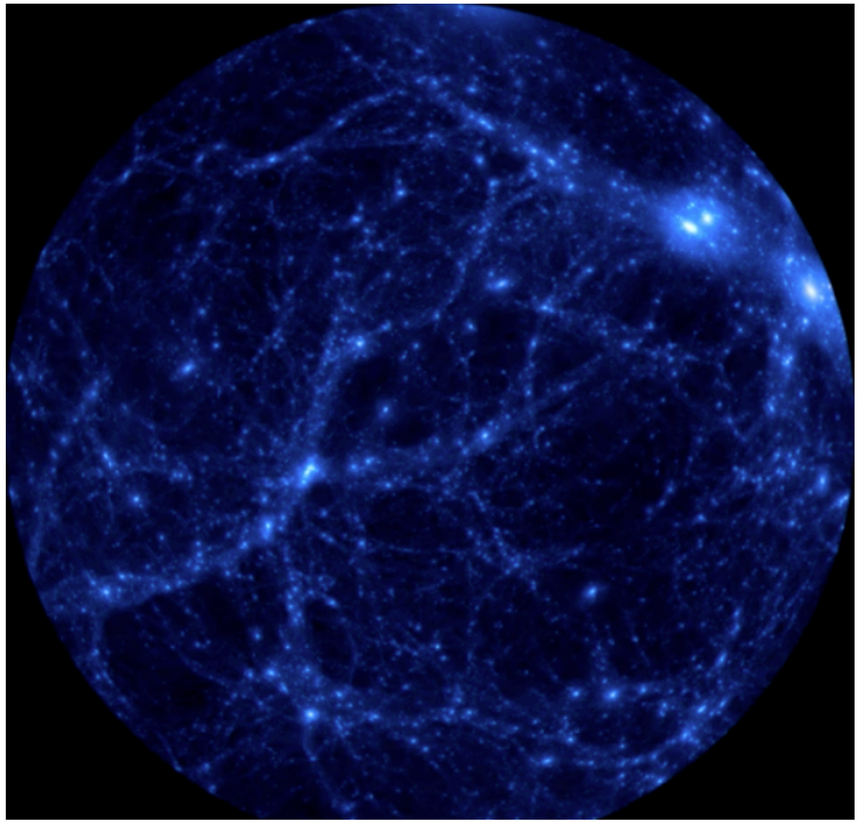
Walls and filaments

Superclusters

Galaxy clusters

Galaxies





CLUES (Constrained Local UniversE Simulations)

Generalized kt clustering algorithms

- Standard techniques use bottom-up (agglomerative) clustering.
- "Heuristic" domain-specific clustering algorithms that do not use the generative model.
- Idea: sequentially cluster jet constituents.
- Characterized by permutation invariance: add 4-momenta of each pair that is clustered.
- Merge closest pair based on a distance measure:

$$d_{ij}^{(\alpha)} = \min(p_{ti}^{2\alpha}, p_{tj}^{2\alpha}) \frac{\Delta R_{ij}^2}{R^2}$$

$$\Delta R_{ij}^2 = (y_i - y_j)^2 + (\phi_i - \phi_j)^2$$

$\alpha = \{1, 0, -1\}$ specifies the the kt, Cambridge/ Aachen and anti-kt algorithms respectively.

Jet Clustering

- Finding the "correct" clustering would be beneficial for downstream tasks in data analyses, e.g. classification, regression.

- Deep learning algorithms:
  - Very successful to solve downstream tasks using low-level features (jet constituents 4-vectors) without inferring the latent structure.
  - Some include physics knowledge directly, e.g. TreeNiN.

- Monte Carlo parton shower generators encode our understanding of the physics process that produces a jet.

- During simulation, successive splittings of the initial state particle generates a set of leaves, following a Markov chain.

- Many trees could give rise to the same set of leaves.

We need to think of jets in probabilistic terms.

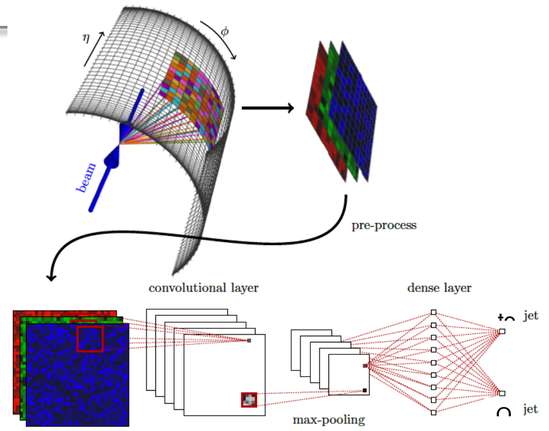
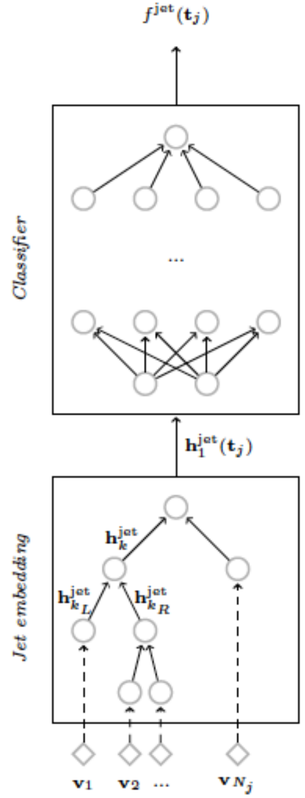


Fig. from Komiske, Metodiev & Schwartz '16

Tree Network in Network
[Macaluso, Cranmer '19]

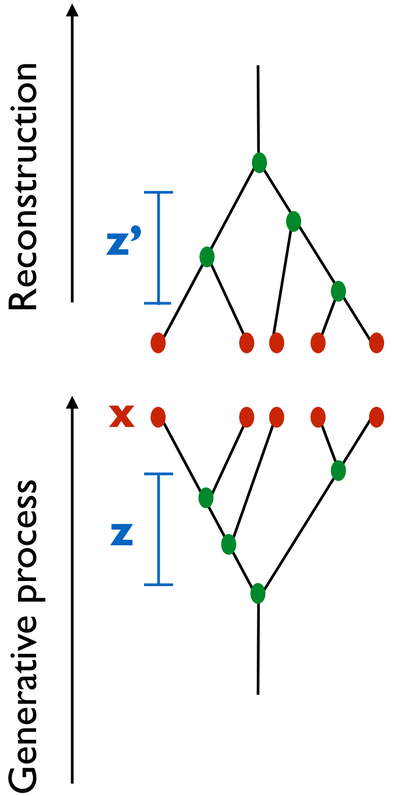
https://github.com/diana-hep/TreeNiN

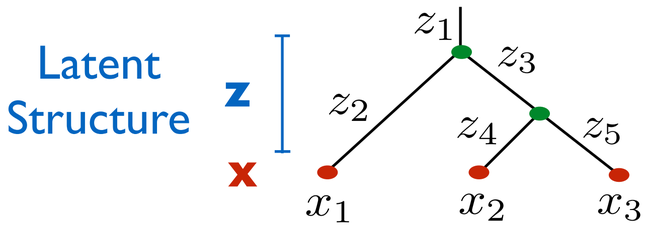


[Louppe, Cho, Becot, Cranmer '17]

Reframing jet physics in probabilistic terms

- Joint likelihood $p(x, z|\theta)$
- Maximum likelihood history: $\mathrm{ArgMax}_z\ p(x, z|\theta)$
- Marginal likelihood $p(x|\theta) = \int dz\ p(x, z|\theta)$
- Maximum likelihood parameter: $\mathrm{ArgMax}_\theta\ p(x|\theta)$
- Posterior distribution on histories: $p(z|x, \theta)$
- Posterior distribution on θ : $p(\theta|x)$





Joint likelihood:

$$p(x, z|\theta) = \prod_j p(x_j|z_{\mathrm{parent}(x_j)}, \theta) \prod_i p(z_i|z_{\mathrm{parent}(z_i)}, \theta)$$

Parton Showers generators

PYTHIA

Herwig 7

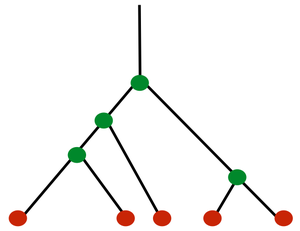
- It is hard to access the joint likelihood of a showering process.

- Typically, four-momentum conservation generates cross correlations between different splittings.

➡ Cannot write the joint likelihood as the product of single branch probabilities:

$$\mathcal{P}_{\mathrm{PS}}(\{t_i, z_i\}) = \mathcal{P}(t_1, z_1) \times \mathcal{P}(t_2, z_2) \times \cdots$$

(Analytic control could be restored [Bauer & Tackmann '08])

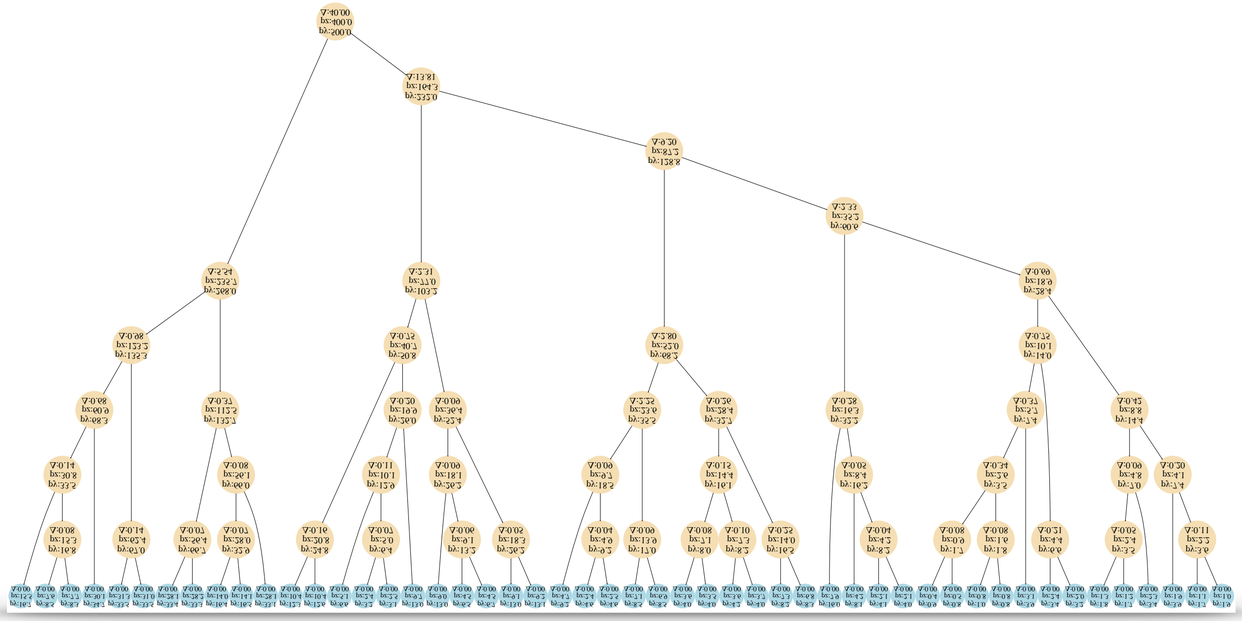


To prototype we built our own simplified model!

Ginkgo: Toy Generative Model for Jets

Kyle Cranmer, SM &
Duccio Pappadopulo

github.com/
SebastianMacaluso/
ToyJetsShower



Motivation

- Build a model to aid in ML research for jet physics.
- Facilitate exact/approximate combinatorial optimization.
- Marginalize over hierarchies (binary trees).

Generation

- Tractable joint likelihood.
- Captures essential ingredients of parton shower generators in full physics simulations.
- Implements an analogue to a parton shower (no hadronization effects).
- Python implementation with few software dependencies.

Inference

- E.g. tuning of simulation parameters (PYTHIA TUNES) to optimize a fit to the data.

Model description

Recursive algorithm to generate a binary tree with jet constituents as the leaves.
Showering process: binary splittings + stopping rule.

Algorithm 1: Toy Parton Shower Generator

function NodeProcessing $(p_{\mathrm{p}}^{\mu}, t_{\mathrm{P}}, t_{\mathrm{cut}}, \lambda, \mathrm{tree})$
  Input : parent momentum $\vec{p}_{\mathrm{p}}$, parent mass squared t_{p}, cut-off mass squared t_{cut}, rate for the exponential distribution λ, binary tree *tree*
  Add parent node to tree.
  if $t_p > t_{cut}$ **then**
    Sample t_L and t_R from the decaying exponential distribution.
    Sample a unit vector from a uniform distribution over the 2-sphere.
    Compute the 2-body decay of the parent node in the parent rest frame.
    Apply a Lorentz boost to the lab frame to each child.
    NodeProcessing $(p_{\mathrm{p}}^{\mu}, t_{\mathrm{L}}, t_{\mathrm{cut}}, \lambda, \mathrm{tree})$
    NodeProcessing $(p_{\mathrm{p}}^{\mu}, t_{\mathrm{R}}, t_{\mathrm{cut}}, \lambda, \mathrm{tree})$

$$t_L \sim f(t|\lambda, t_{\mathrm{P}}) = \frac{1}{1-e^{-\lambda}} \frac{\lambda}{t_{\mathrm{P}}} e^{-\frac{\lambda}{t_{\mathrm{P}}} t} \quad t_R \sim f(t|\lambda, t_{\mathrm{P}}, t_L) = \frac{1}{1-e^{-\lambda}} \frac{\lambda}{(\sqrt{t_{\mathrm{P}}} - \sqrt{t_L})^2} e^{-\frac{\lambda}{(\sqrt{t_{\mathrm{P}}} - \sqrt{t_L})^2} t}$$

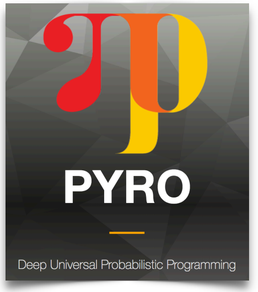
Features

- Momentum conservation.
- Running of the splitting scale.

Facilitates research implementing:

- Probabilistic programming
- Differentiable programming
- Dynamic programming
- Variational inference

The model keeps track of the augmented data based on a **PYRO** implementation.



Generation and inference unification

We aim to invert the process probabilistically

- Reconstructing the splitting history is like inverting the generative model.
- The number of clustering histories is enormous! They grow like (2N-3)!!, with N the number of jet constituents.
- Traditional jet reconstruction algorithms don't use the probability model directly.
- We use the likelihood for the history as the optimization objective.

# of leaves	Approx. # of trees
4	15
5	100
7	10 K
9	2 M
11	600 M
150	10^{300}

The fifteen different rooted binary trees (with unordered children) on a set of four labeled leaves, illustrating $15 = (2 \times 4 - 3)!!$ (see article text).

https://en.wikipedia.org/wiki/Double_factorial

Improving over sequential (agglomerative) clustering

https://github.com/SebastianMacaluso/StandardHC

- Standard techniques are based on heuristics and are greedy.

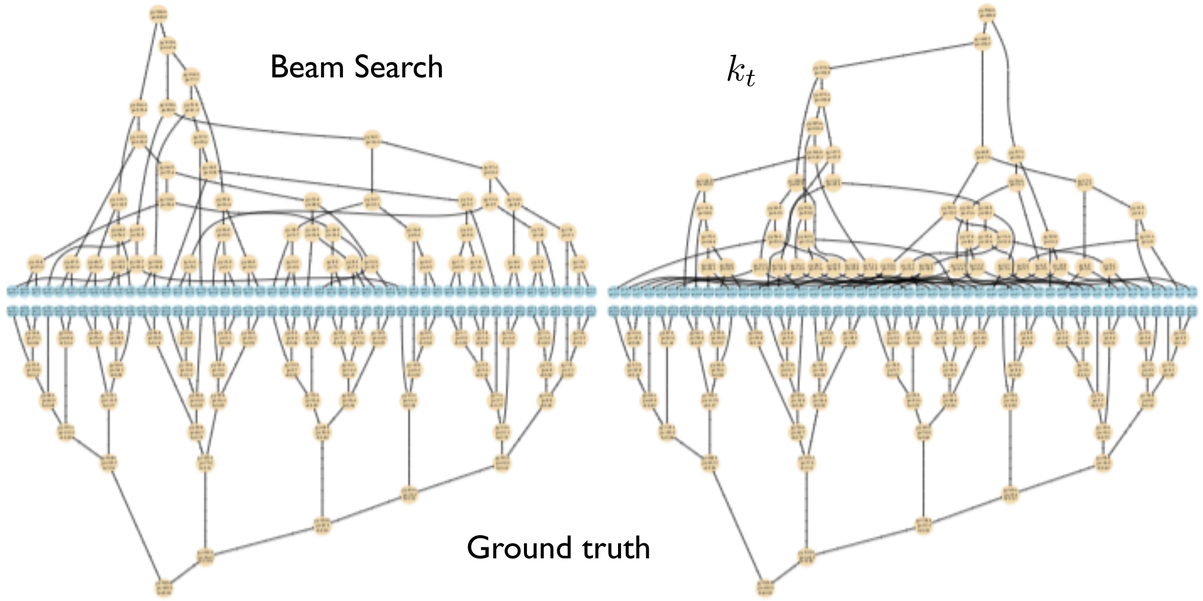
Greedy algorithms make the locally optimal choice at each step

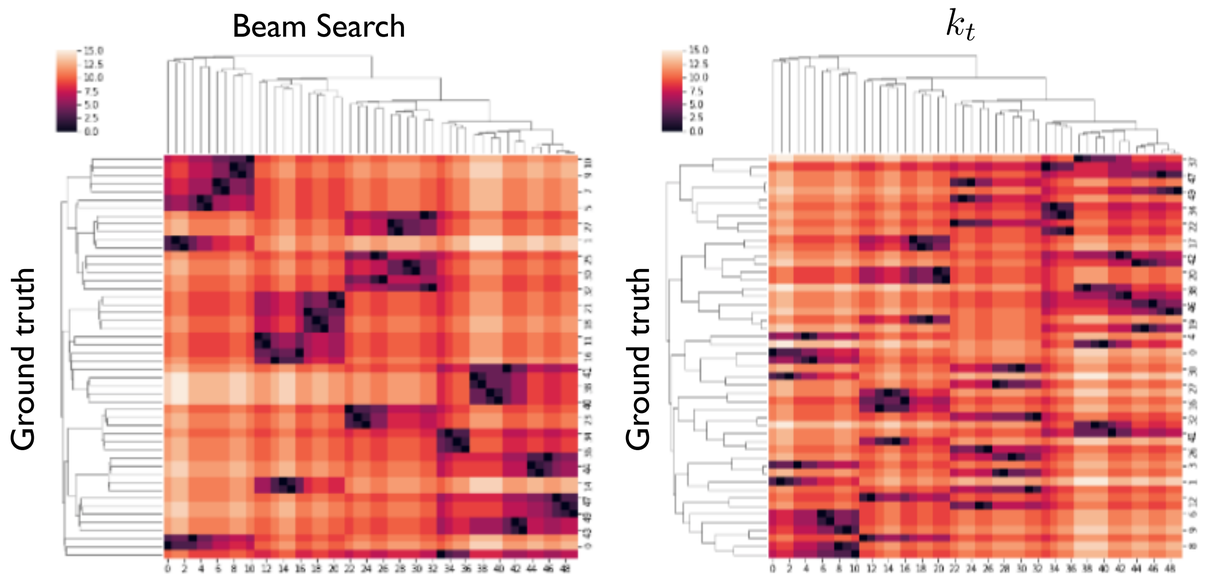
Straightforward improvements:

- Use the splitting likelihood encoded in the generative model instead of heuristics.
- Switch to other clustering algorithms, e.g. beam search to find the maximum likelihood estimate for the clustering latent structure.

$$\hat{z} = \text{argmax}_z \; p(z|x,\theta)$$

Beam Search: maximize the likelihood of multiple steps before choosing the latent path.





https://github.com/SebastianMacaluso/VisualizeBinaryTrees

Hierarchical Cluster Trellis

AISTATS 2021 [Greenberg, Macaluso, Monath, Cranmer, et al '20, arXiv: 2002.11661]

NYU

Craig Greenberg

Nicholas Monath

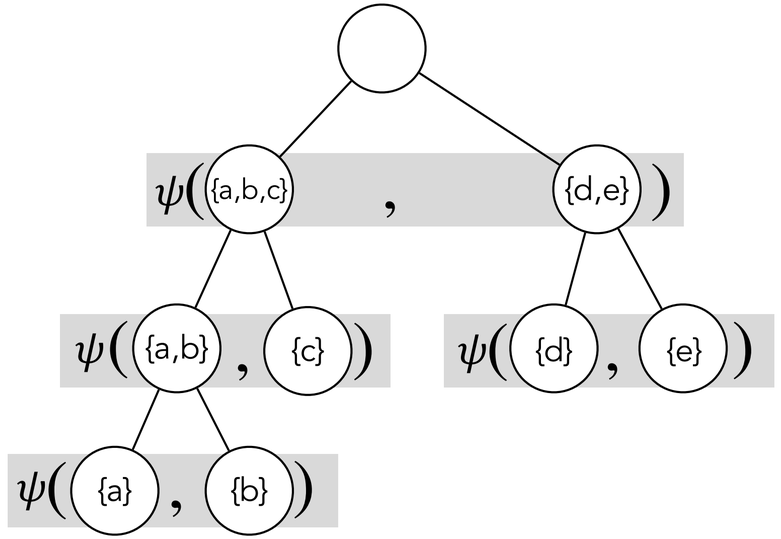
- New data structure to efficiently consider every possible hierarchical clustering.
  - Exact Maximum likelihood showering history $\text{ArgMax}_z\ p(x, z|\theta)$
  - Marginal likelihood $p(x|\theta)$ (sum of the likelihood of all the clustering histories).

https://github.com/SebastianMacaluso/ClusterTrellis

- Model needs to be defined in terms of the product of pairwise splittings:

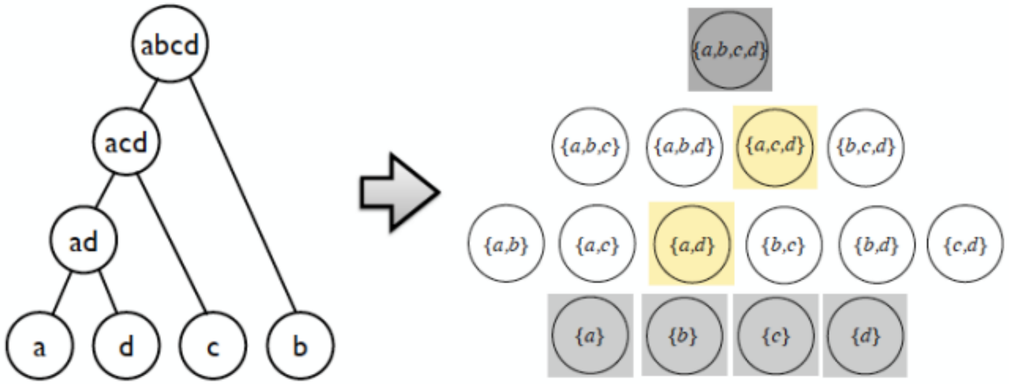
$$P(\mathtt{H}|X) = \frac{\phi(X|\mathtt{H})}{Z(X)} \text{ with } \phi(X|\mathtt{H}) = \prod_{X_L, X_R \in \text{sibs}(\mathtt{H})} \psi(X_L, X_R)$$

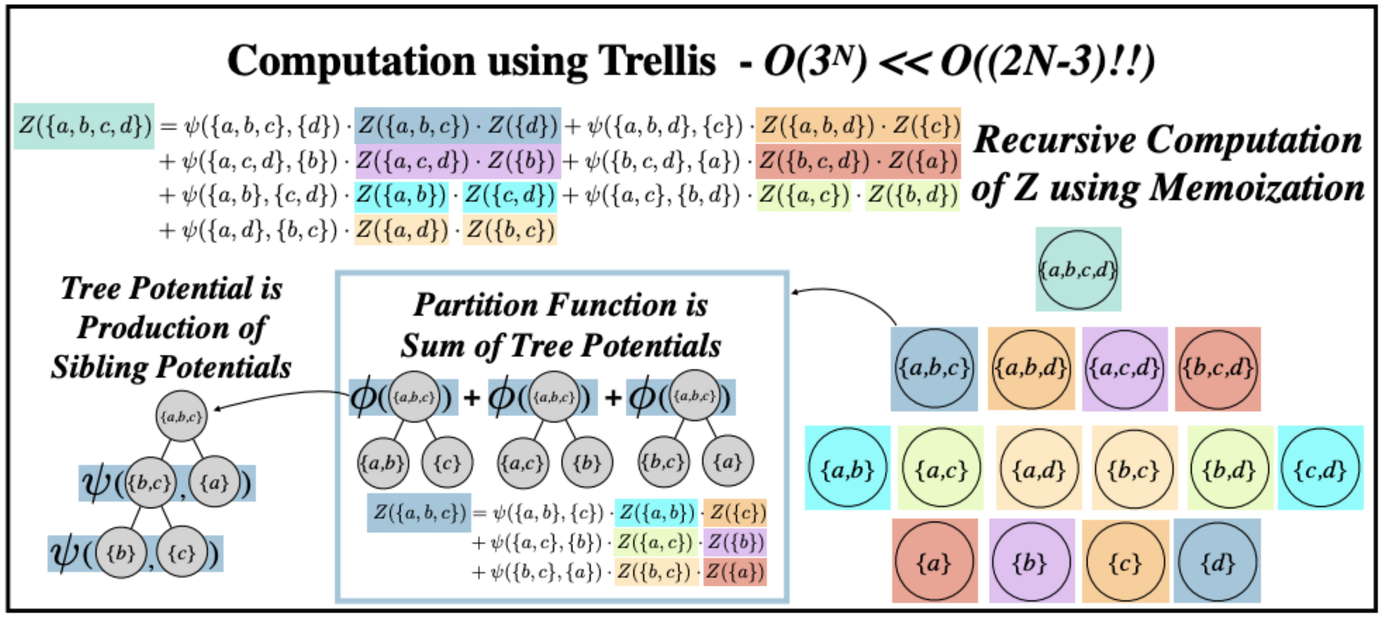
$$Z(X) = \sum_{\mathtt{H} \in \mathcal{H}(X)} \phi(X|\mathtt{H}).$$

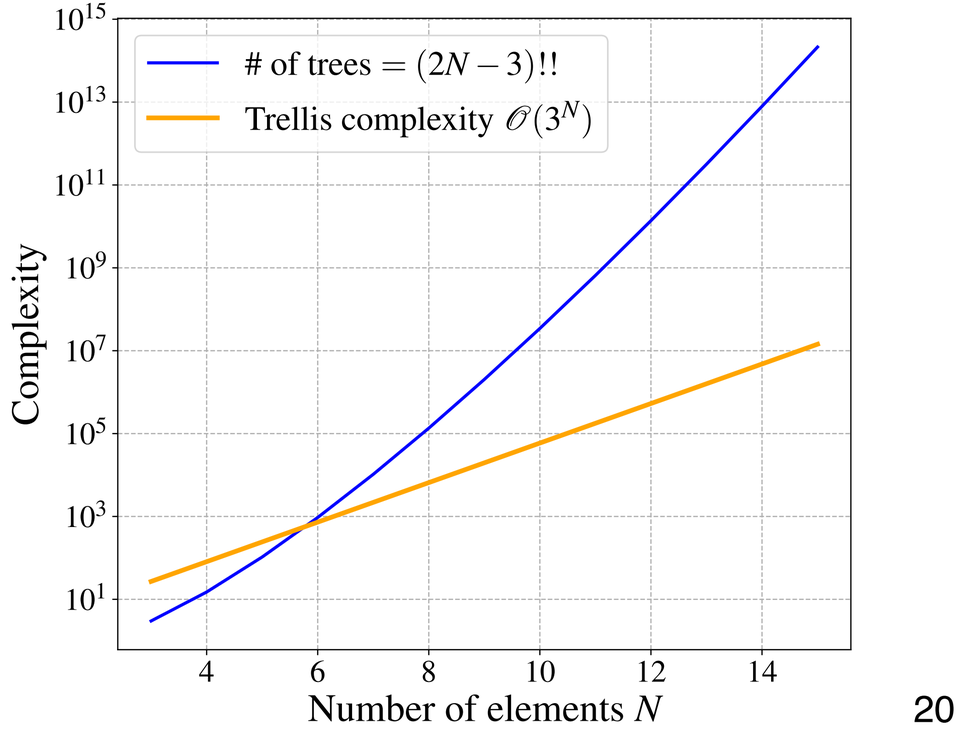


Trellis Structure

- We represent each set of elements as a node in the trellis.
- There are $2^{|x|}$ nodes for a full trellis, but the number of trees grows super-exponentially.
- It allows to run a dynamic programming algorithm to compute the marginal likelihood $p(x|\theta) = \int dz\ p(x, z|\theta)$ (over all possible clusterings) and the exact maximum likelihood hierarchy $\mathrm{ArgMax}_z\ p(x, z|\theta)$

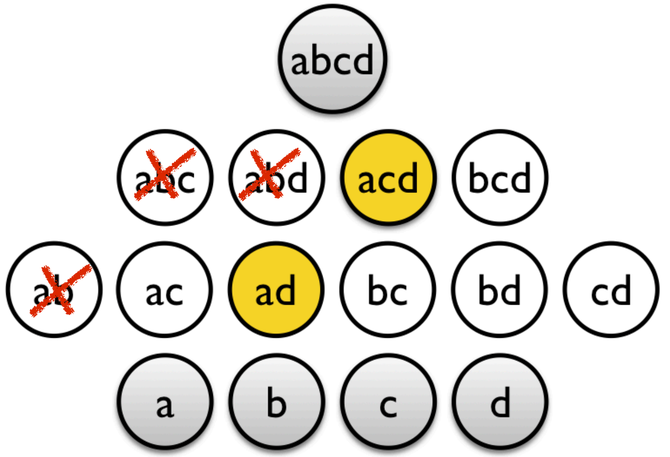






Sparse Hierarchical Cluster Trellis

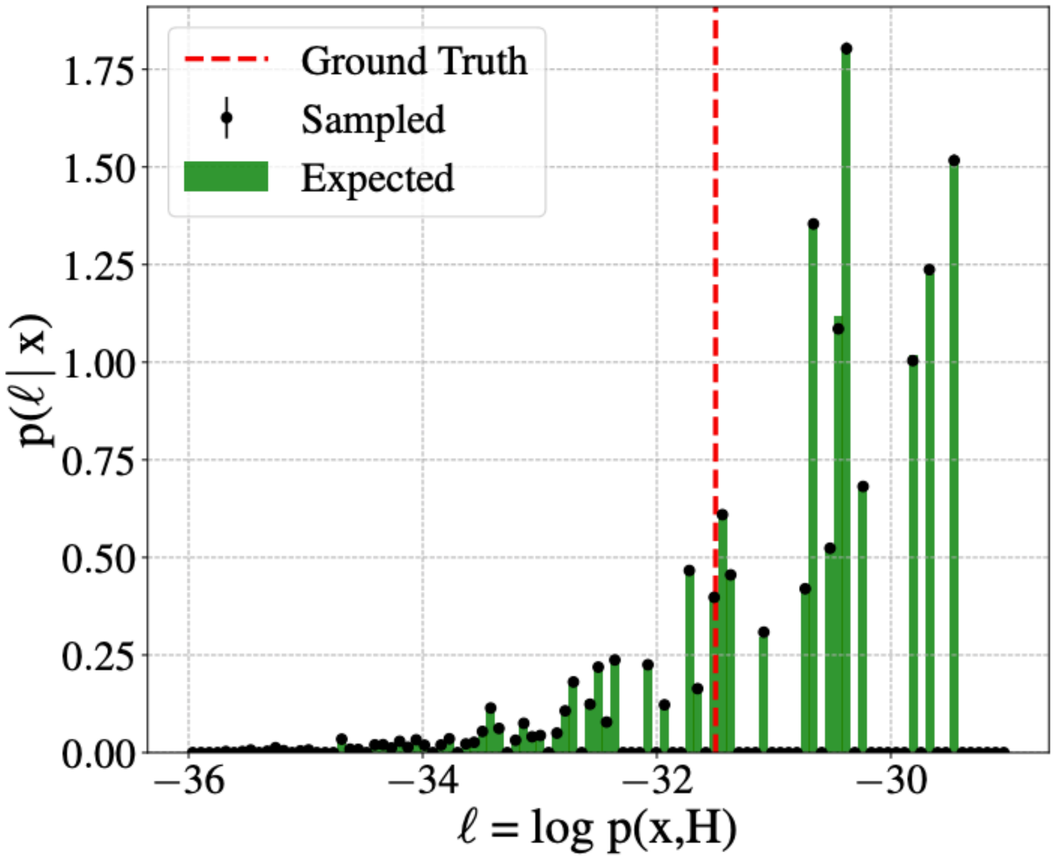
- The trellis still grows exponentially.
- We can build a sparse trellis by removing nodes, we consider a fraction of hierarchies from the total of (2 N - 3)!!.
- Many histories likelihood are negligible compared to the maximum likelihood (MLE) history.
- One can also construct a sparse trellis from samples (e.g., ground truth from a simulator, greedy, or beam search) or randomly sample pairwise splittings.



Posterior distribution - Sampling procedure

- The trellis encodes a distribution over all possible trees.
- Traversing the trellis top-bottom is similar to running the generative model.
- Trellis enables to sample histories from the true posterior distribution conditioned on a set of leaves (jet constituents) without enumerating all possible clusterings.

$$p(z|x,\theta) = \frac{p(x|z,\theta)\ p(z|\theta)}{p(x|\theta)}$$

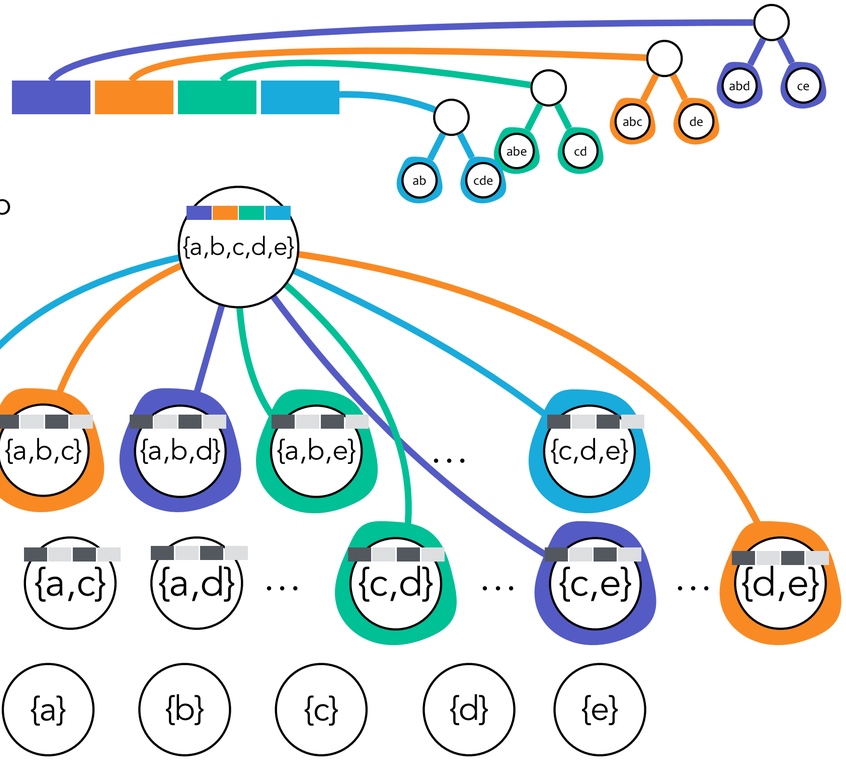


A* search + trellis

- We proposed A* search on the trellis to find the MLE hierarchy.
- Best-first search algorithm that redefines clustering as search.
- Based on a heuristic to determine the most promising path.
- Trellis compactly encodes states in the search space and allows a top-down exploration.
- Compactly represents the search frontier as nested priority queues.
- Approximate version based on a sparse trellis and/or limiting the number of trees explored.

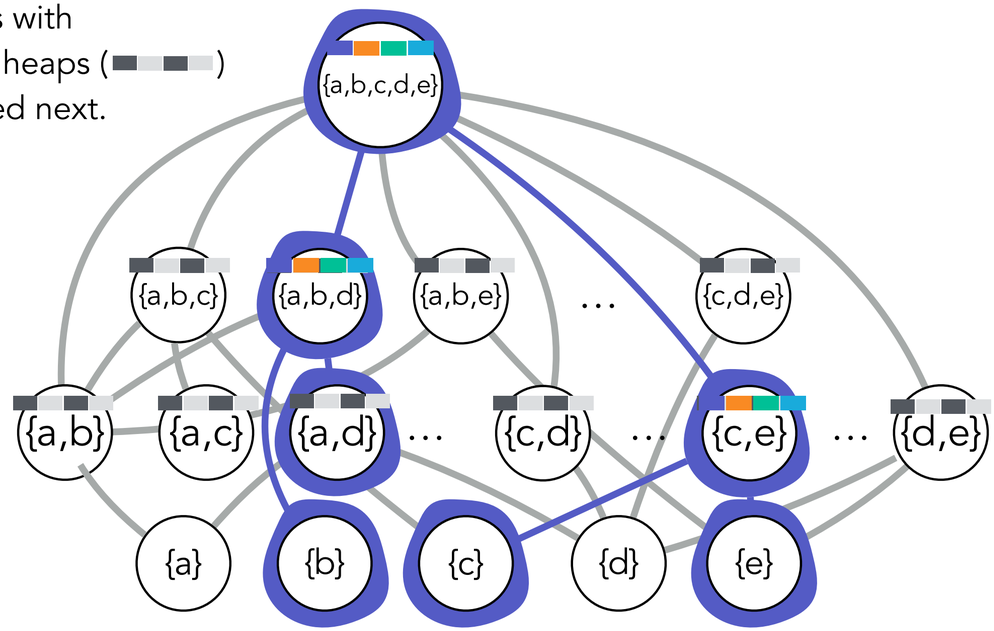
Min-Heap Over Child Pairs

Store a separate heap at each node.



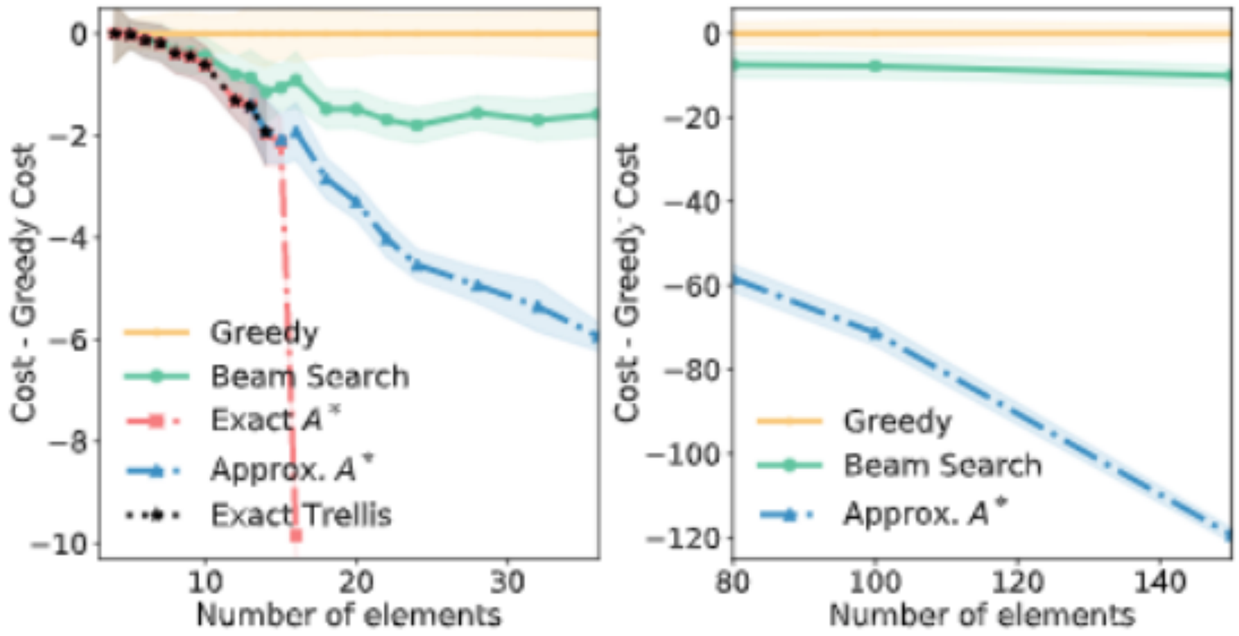
Partial Hierarchical Clustering State

Leaf nodes with un-initialized heaps ( ) to be explored next.



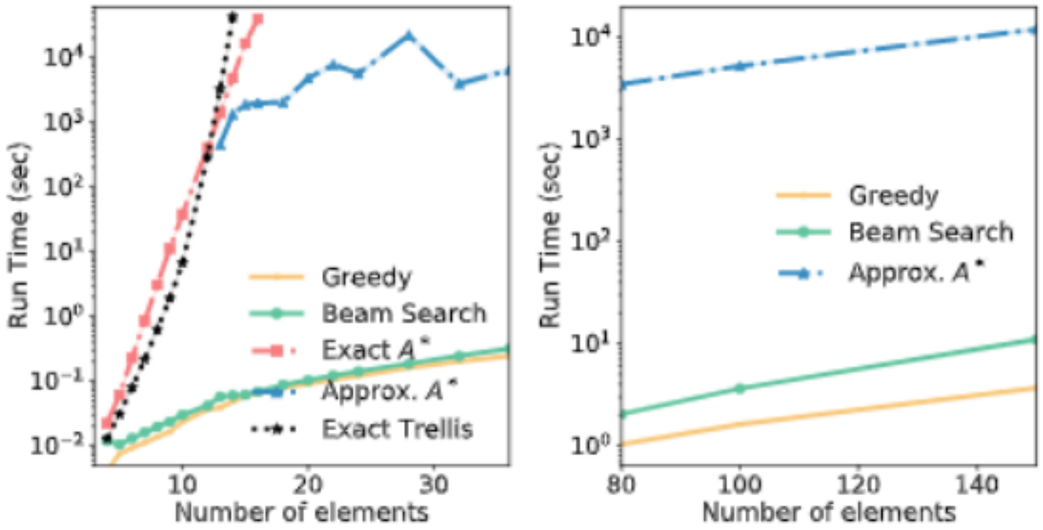
Results

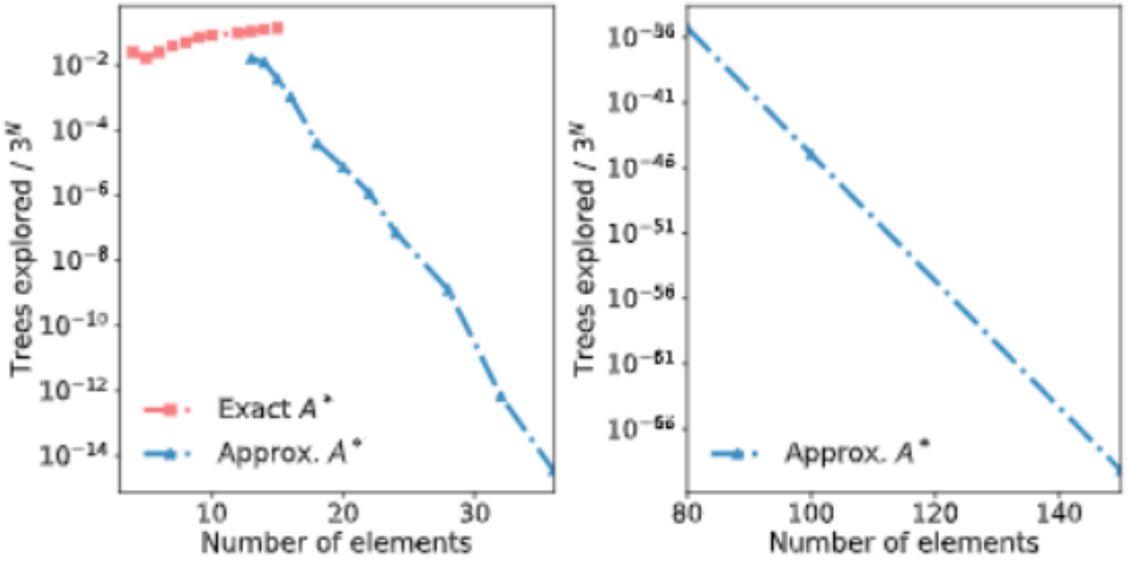
- We find the exact maximum likelihood tree for up to 16 jet constituents.
- Our approx. algorithms greatly improve over greedy and beam search baselines.



Results

A* has a controlled run time even for search spaces as large as 10^{300}.





Approximate A* explores a tiny fraction of the 3^N nodes in the trellis, which is already super-exponentially smaller than the number of trees.

Final remarks

- Reframed jet physics in probabilistic terms.
- Introduced a simplified generative model to facilitate research into new computational techniques for jet physics.
- Reframed jet clustering as a way to invert the generative model.
- New implementations of likelihood-based clustering algorithms: greedy and beam search.
- Hierarchical cluster trellis and A* algorithms to exactly obtain the maximum likelihood history (approximate versions greatly improve over baselines).
- New algorithms provide a principled alternative to the generalized k_t clustering algorithms.
- Cluster trellis allows to marginalize and sample from the true posterior distribution of clustering histories which could ameliorate bottlenecks in physics simulations with large jet multiplicity.

Thanks for your attention!

Probabilistic programming

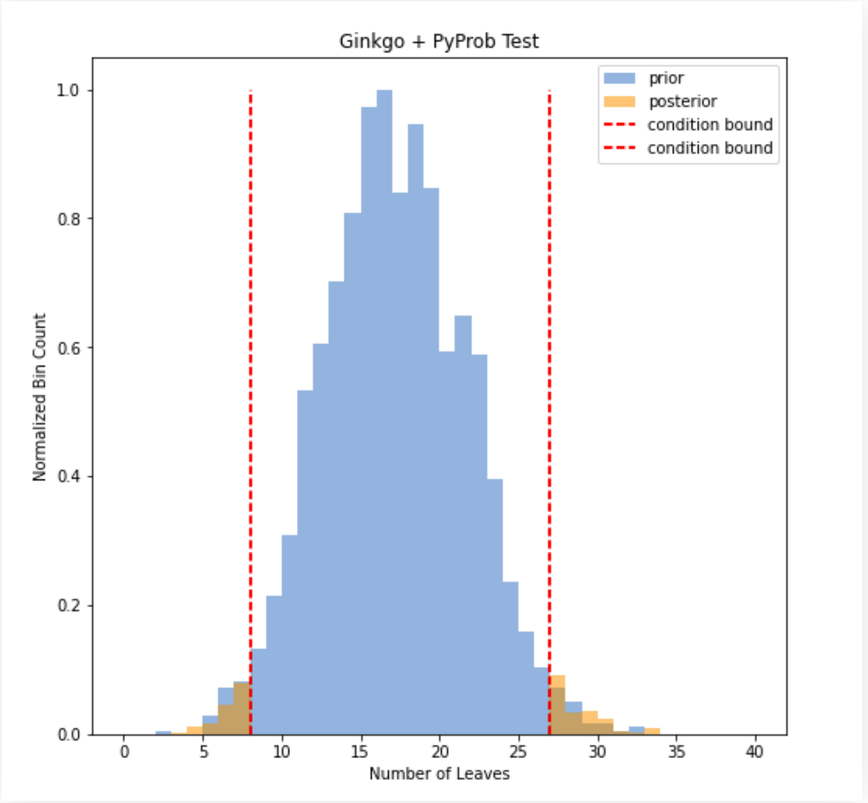
[Drnevich, Cranmer, Baydin, Macaluso ’20, work in progress]

Monte Carlo event generators sample from the distribution $p(x, z|\theta)$ over histories and jet constituents through random number generators.

Probabilistic programming allows to condition the values of random variables x or z by hijacking the random number generators.

Can bias the simulator towards the desired output, e.g. importance sampling.

- Applied to Sherpa [Baydin, Shao, Cranmer et al ’19].
- Allows to efficiently sample the tails of backgrounds in signal-rich regions of phase space.



Importance sampling on Ginkgo jets using PyProb and conditioning on the number of constituents.

Final remarks and future directions

- Many highly efficient and competitive top taggers with comparable performance!
- Reframe jet physics in probabilistic terms.
- Goal: Unification of generation and inference.
  - e.g. Reframe jet clustering as finding maximum likelihood clustering.
  - e.g. Reframe parameter tuning as maximum likelihood after marginalizing over the (latent) clustering histories.
- New likelihood-based clustering algorithms: greedy, beam search and hierarchical trellis. Beam search is strictly better than greedy algorithm (e.g. kt) in terms of estimating the most likely tree.
- Emerging computational techniques to address bottlenecks in jet physics:
  - Likelihood-free inference for model parameters tuning.
  - Hierarchical cluster trellis that enables to exactly obtain the maximum likelihood history and consider every possible clustering history.
  - Probabilistic programing to efficiently sample the tails of complicated phase space regions.

Thanks for your attention!

Generative process

Start with $\Delta_{\mathrm{P}} = \Delta_{\mathrm{root}}$

Sample $\phi \sim \mathrm{Uniform}(0, 2\pi)$ to pick a direction for Δ_{P}

Heavy resonance X jet

$$\Delta_{\mathrm{root}} = \frac{m_X}{2}$$

Split parent node

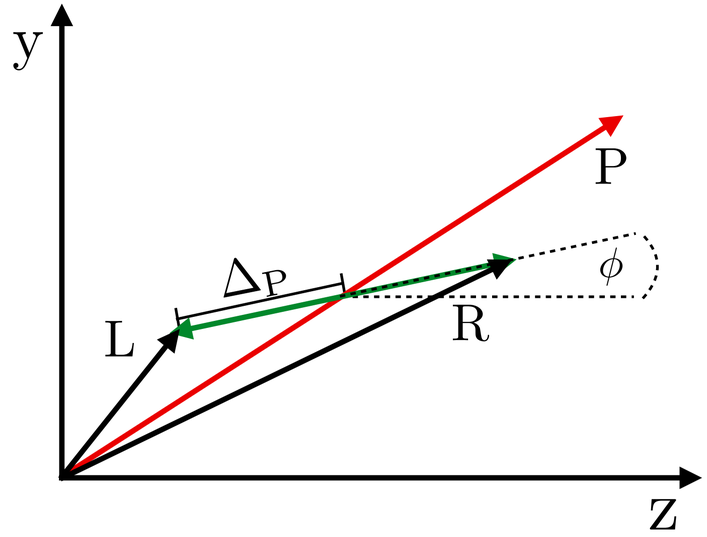
$$\vec{p}_{\mathrm{L}} = \frac{1}{2}\vec{p}_{\mathrm{p}} - \vec{\Delta}_{\mathrm{p}}$$

$$\vec{p}_{\mathrm{R}} = \frac{1}{2}\vec{p}_{\mathrm{p}} + \vec{\Delta}_{\mathrm{p}}$$

Sample independently Δ_{L} and Δ_{R}

$$\Delta \sim f(\Delta|\lambda, \Delta_{\mathrm{p}}) = \frac{\lambda}{\Delta_{\mathrm{p}}} e^{-\frac{\lambda}{\Delta_{\mathrm{p}}}\Delta}$$

Recurse over (Δ_{L},L) and (Δ_{R},R).



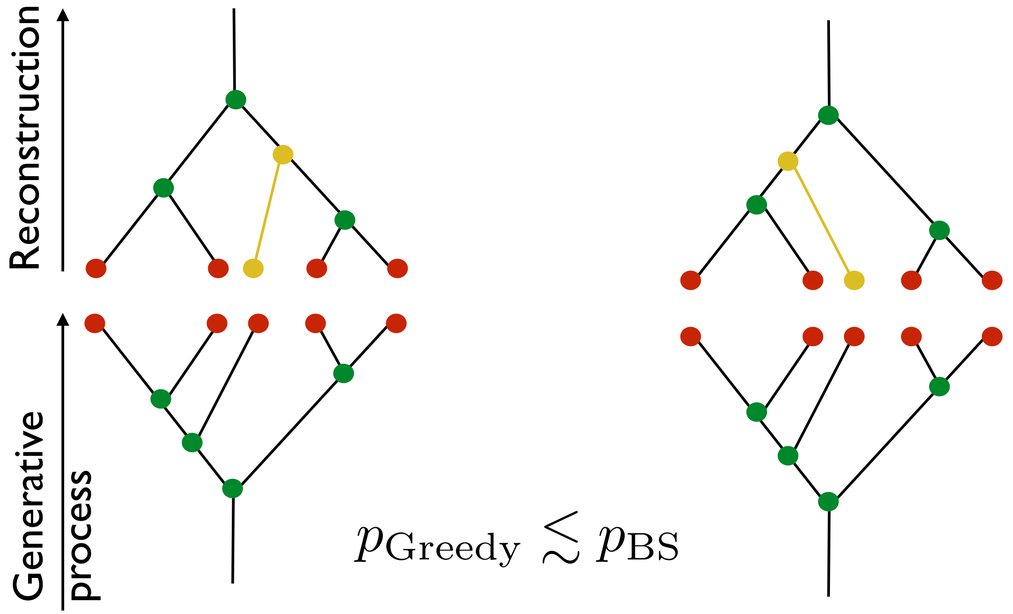
Three latent variables per particle

$\vec{p} = (p_y, p_z)$ and Δ

Likelihood-based jet clustering

Greedy
Joint likelihood from locally maximizing the likelihood at each step.

Beam Search
Maximize the likelihood of multiple steps before choosing the latent path. Maybe we could fix mistakes?



Joint likelihood:

$$p(x, z|\theta) = \prod_i p(\mathrm{L}_i, \mathrm{R}_i|\mathrm{P}_i, \theta)$$

Goal: find the latent structure that maximizes the jet likelihood (MLE).

Given an algorithm, z is fixed.

$$\hat{z}_{\text{MLE}} = \text{ArgMax}_z \; p(x, z|\theta)$$

Viterbi algorithm
Computationally "infeasible" for jets with ~ 100 constituents

$\hat{z}_{\text{Greedy}}$

$\hat{z}_{\text{BS}}$

Computational bottlenecks in Jet Physics

- **Tuning the parameters of the shower model**
  - Ideally we would tune the parameters θ of PYTHIA, Sherpa or Herwig to find the maximum likelihood $p(\theta|x)$. This typically involves a high dimensional parameter space and intractable quantities.

- **Event Generation for events with large jet multiplicity**
  - Matching algorithms, e.g. CKKW-L [arXiv:0109231, 0112284], need to consider every possible clustering history and reweigh them. This becomes infeasible for more than 6 jet configurations.

- **Simulation efficiency for jet backgrounds in signal-rich regions of phase space**
  - Enormous rate of multijets events: how to populate the tails of complicated phase space regions?

Emergent computational techniques that could address/ameliorate these problems:

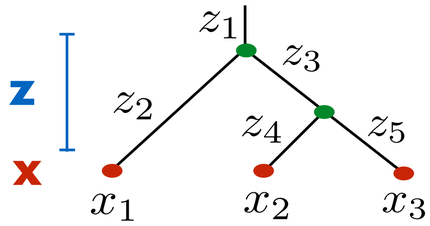
- Likelihood-free Inference of model parameters.
- Hierarchical Cluster Trellis.
- Probabilistic Programming.

Likelihood-free inference of model parameters

[Brehmer, Louppe, Pavez & Cranmer '18]
[Brehmer, Cranmer, Louppe & Pavez '18]
[Andreassen & Nachman '19]

Posterior distribution on model parameters:

$$p(\theta|x) = \frac{p(x|\theta)\ p(\theta)}{\int d\theta' p(x|\theta') p(\theta')}$$



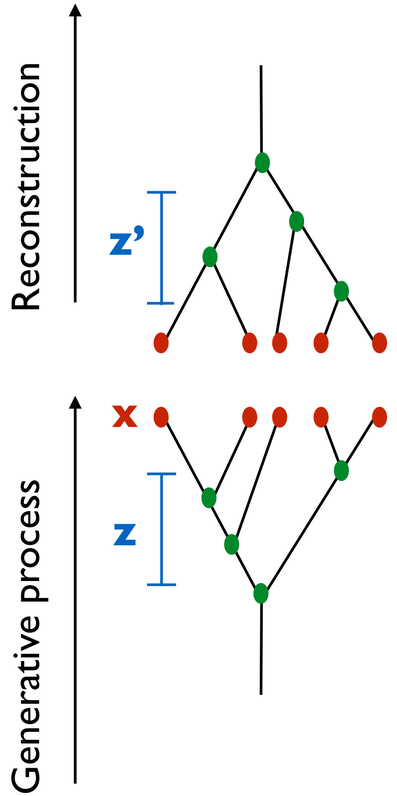
Marginal likelihood:

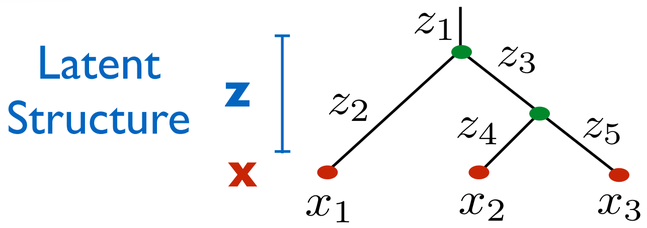
$$p(x|\theta) = \int dz\ p(x|z,\theta)\ p(z,\theta)$$ ← Intractable

Approximate $p(x|\theta)$ using machine learning without explicitly marginalizing over all latent structures z.

Reframing jet physics in probabilistic terms

- Joint likelihood $p(x, z|\theta)$
- Maximum likelihood history: $\mathrm{ArgMax}_z\ p(x, z|\theta)$
- Marginal likelihood $p(x|\theta) = \int dz\ p(x, z|\theta)$
- Maximum likelihood parameter: $\mathrm{ArgMax}_\theta\ p(x|\theta)$
- Posterior distribution on histories: $p(z|x, \theta)$
- Posterior distribution on θ : $p(\theta|x)$





Joint likelihood:

$$p(x, z|\theta) = \prod_j p(x_j|z_{\mathrm{parent}(x_j)}, \theta) \prod_i p(z_i|z_{\mathrm{parent}(z_i)}, \theta)$$

Improving over sequential (agglomerative) clustering

- Standard techniques are based on heuristics and are greedy.

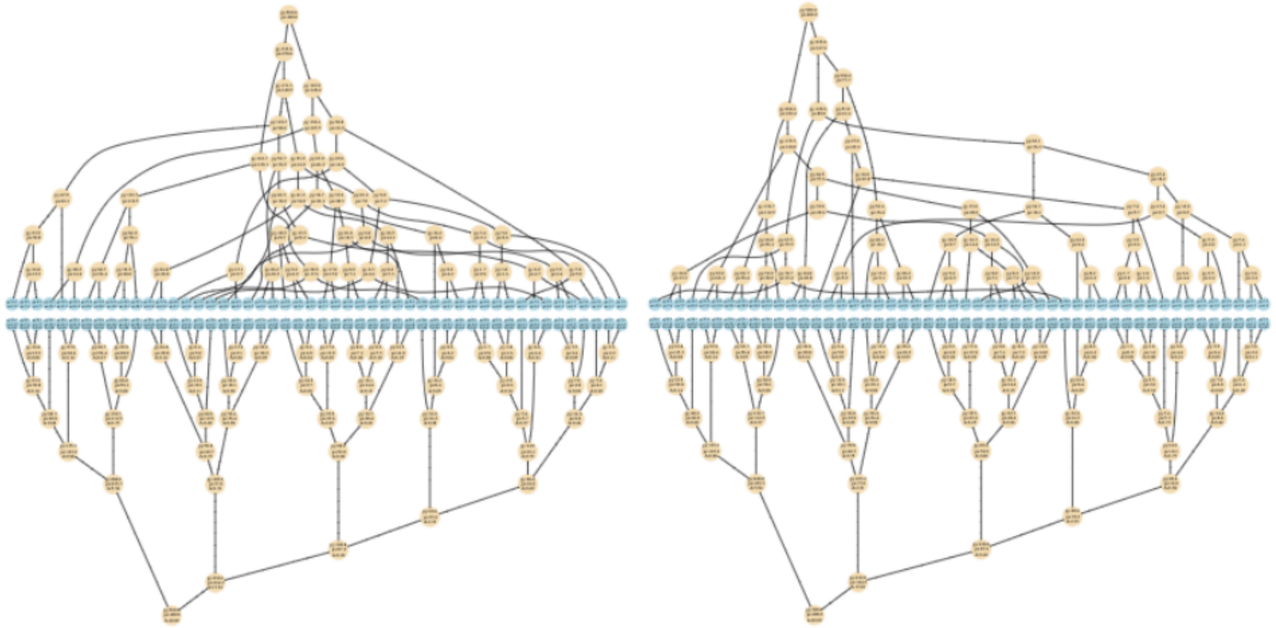
Greedy algorithms make the locally optimal choice at each step

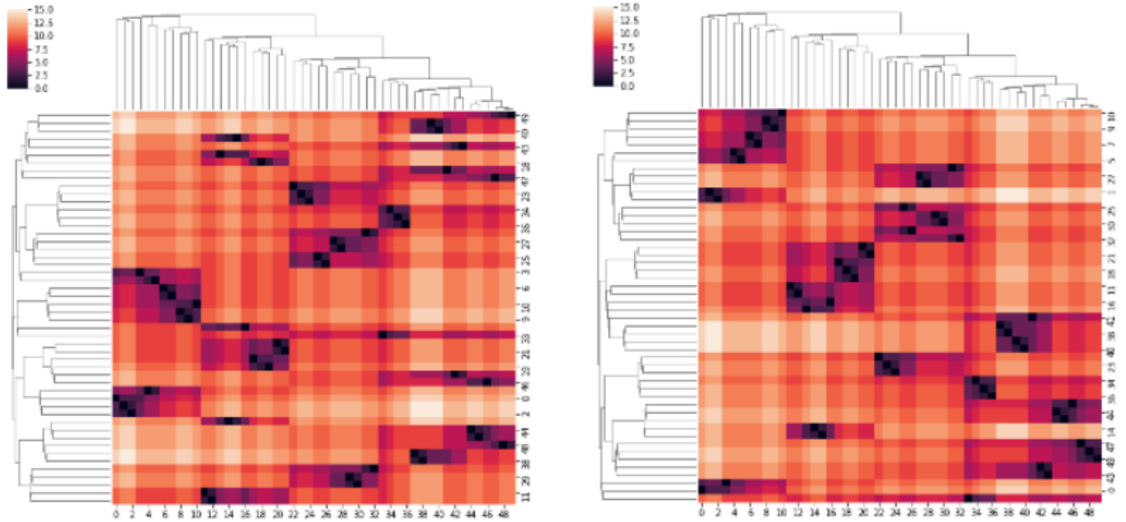
Straightforward improvements:

- Use the splitting likelihood encoded in the generative model instead of heuristics.
- Switch to other clustering algorithms, e.g. beam search to find the maximum likelihood estimate for the clustering latent structure.

$$\hat{z} = \text{argmax}_z \; p(z \mid x, \theta)$$

Beam Search: Maximize the likelihood of multiple steps before choosing the latent path.





Hierarchical Cluster Trellis

Event Generation for events with large jet multiplicity.

During simulations, when implementing the CKKW-L matching algorithm, parton final states need to be reweighted with the corresponding Sudakov form factors of each history . $p(x, z|\theta)$

Trellis could be extended to consider 2 → 3 splittings (currently based on binary trees, 1 → 2 splittings); could make feasible the implementation of CKKW-L to a higher jet multiplicity.

Shower deconstruction

[Soper & Spannowsky '11] [Soper & Spannowsky '12]

- Simplified shower model.
- Add likelihood of each clustering history over *microjets* conditioned on S or B.
- Define the likelihood ratio:

$$\chi(p_N) = \frac{P(p_N|S)}{P(p_N|B)}$$

Trellis could provide a more efficient way to consider every history over microjets and/or extend the algorithm to more microjets.