Toy Generative Model for Jets

Kyle Cranmer, SM & Duccio Pappadopulo

Sebastian Macaluso
New York University

IRIS-HEP topical meeting
September 25, 2019

Toy Generative Model for Jets

Kyle Cranmer, SM & Duccio Pappadopulo

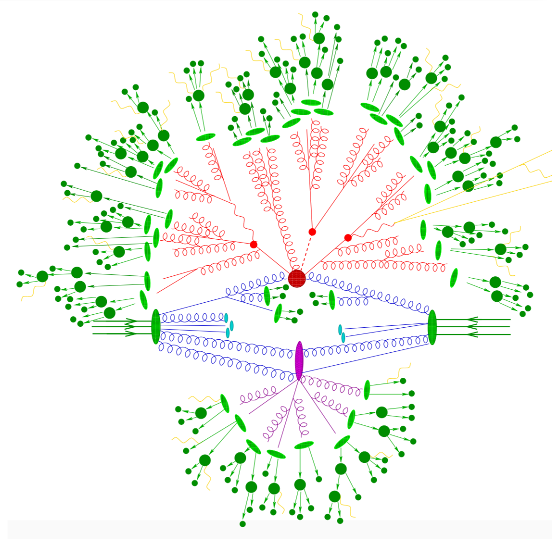
https://github.com/SebastianMacaluso/ToyJetsShower

Motivation: build a model to aid in machine learning research for jet physics.

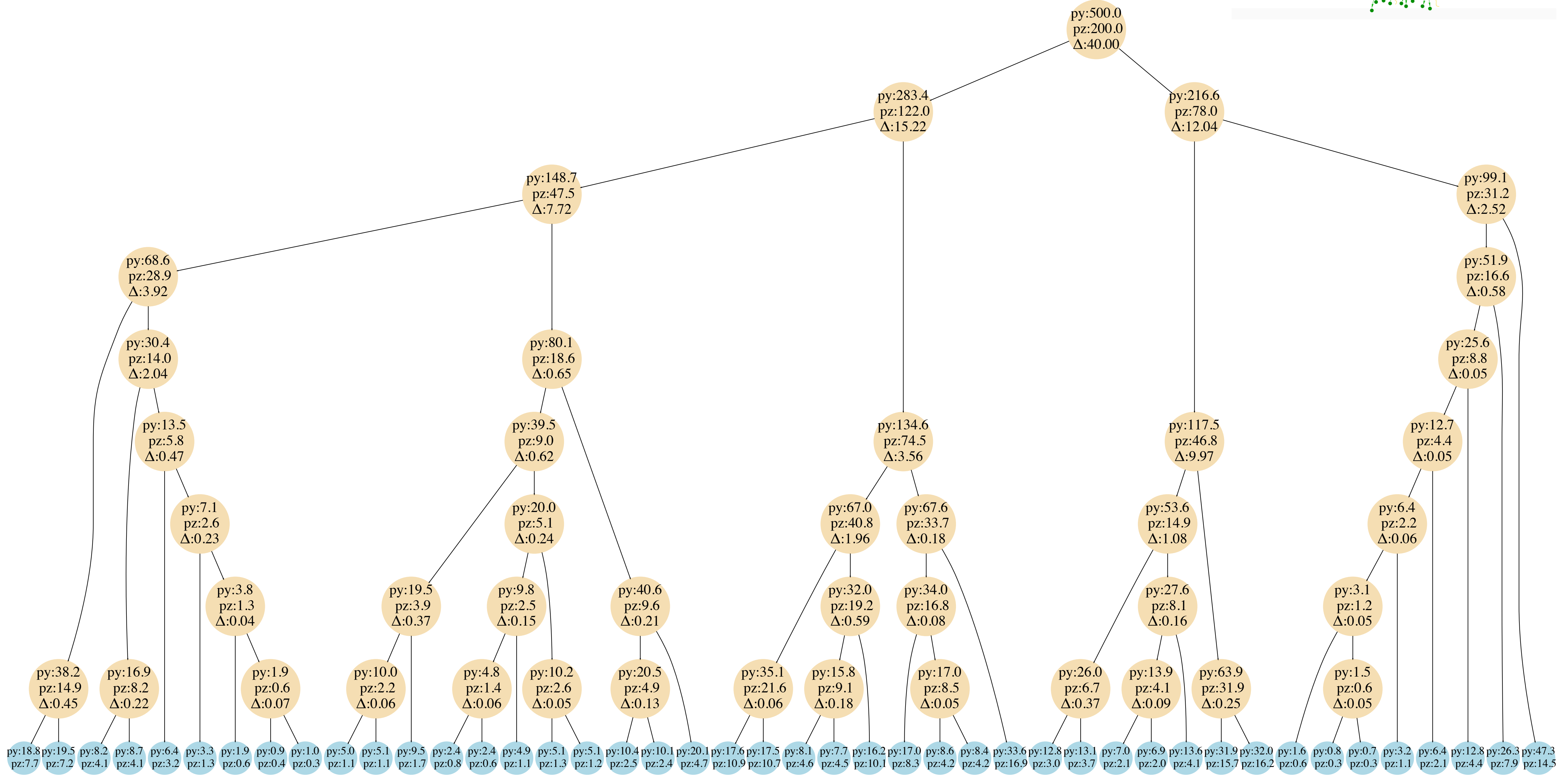
- Captures essential ingredients of full physics simulations.
- Implements an analogue to a parton shower (no hadronization effects).
- Tractable joint likelihood
- Presented in machine-learning friendly format

Sample jet

Jet constituents + showering path => jet binary tree



[Stefan Hoche 1411.4085]



Model Considerations

Recursive algorithm to generate a binary tree with jet constituents as the leaves.

Showering process: binary splittings + stopping rule.

Generation

- Momentum conservation
- Running of the splitting scale

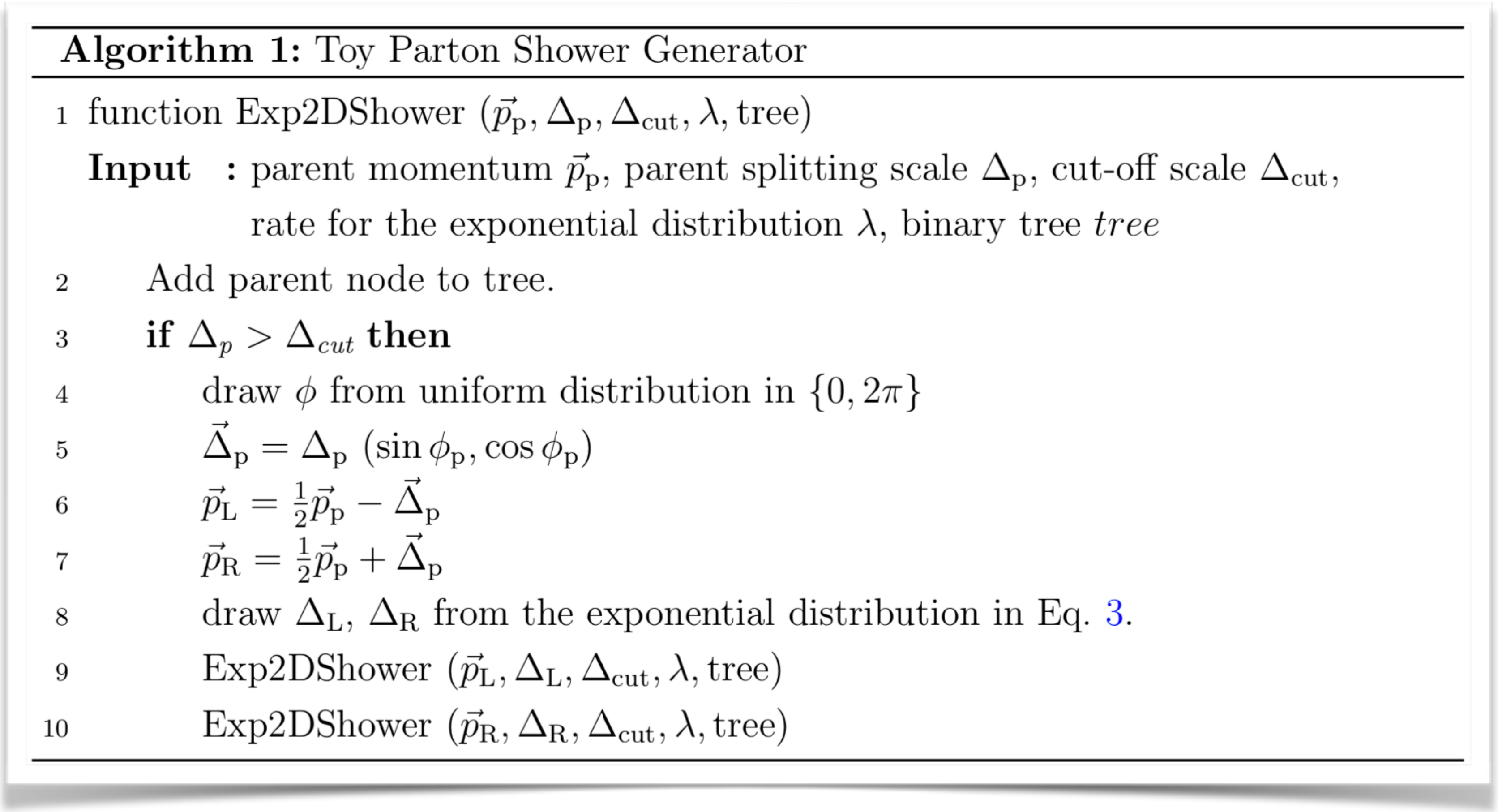
Reconstruction

- Permutation invariance
- Distance measure (e.g. k_t)

=> An analogue of the generalized k_t clustering algorithms can be defined.

Model Specification

- Stand-alone mathematical **specification** of model
- Document describing model to non-physicists
- Algorithm described in computer science style
- Python implementation with few software dependencies



Algorithm 1: Toy Parton Shower Generator

```
function Exp2DShower ($\vec{p}_{\rm p}, \Delta_{\rm p}, \Delta_{\rm cut}, \lambda$, tree)
Input : parent momentum $\vec{p}_{\rm p}$, parent splitting scale $\Delta_{\rm p}$, cut-off scale $\Delta_{\rm cut}$,
        rate for the exponential distribution $\lambda$, binary tree *tree*
    Add parent node to tree.
    if $\Delta_p > \Delta_{cut}$ then
        draw $\phi$ from uniform distribution in $\{0, 2\pi\}$
        $\vec{\Delta}_{\rm p} = \Delta_{\rm p}\ (\sin\phi_{\rm p}, \cos\phi_{\rm p})$
        $\vec{p}_{\rm L} = \frac{1}{2}\vec{p}_{\rm p} - \vec{\Delta}_{\rm p}$
        $\vec{p}_{\rm R} = \frac{1}{2}\vec{p}_{\rm p} + \vec{\Delta}_{\rm p}$
        draw $\Delta_{\rm L}$, $\Delta_{\rm R}$ from the exponential distribution in Eq. 3.
        Exp2DShower ($\vec{p}_{\rm L}, \Delta_{\rm L}, \Delta_{\rm cut}, \lambda$, tree)
        Exp2DShower ($\vec{p}_{\rm R}, \Delta_{\rm R}, \Delta_{\rm cut}, \lambda$, tree)
```

Toy Generative Model for Jets

Kyle Cranmer[1], Sebastian Macaluso[1] and Duccio Pappadopulo[2]

1 Center for Cosmology and Particle Physics & Center for Data Science, New York University, USA
2 Bloomberg LP, New York, NY 10022, USA.

1 Introduction

In this notes, we provide a standalone description of a generative model to aid in machine learning (ML) research for jet physics. The motivation is to build a model that has a tractable likelihood, and is as simple and easy to describe as possible but at the same time captures the essential ingredients of parton shower generators in full physics simulations. The aim is for the model to have a python implementation with few software dependencies.

Parton shower generators are software tools that encode a physics model for the simulation of jets that are produced at colliders, e.g. the Large Hadron Collider at CERN. Jets are a collimated spray of energetic charged and neutral particles. Parton showers generate the particle content of a jet, going through a cascade process, starting from an initial unstable particle. In this description, there is a recursive algorithm that produces binary splittings of an unstable parent particle into two children particles, and a stopping rule. Thus, starting from the initial unstable particle, successive splittings are implemented until all the particles are stable (i.e. the stopping rule is satisfied for each of the final particles). We refer to this final particles as the jet constituents.

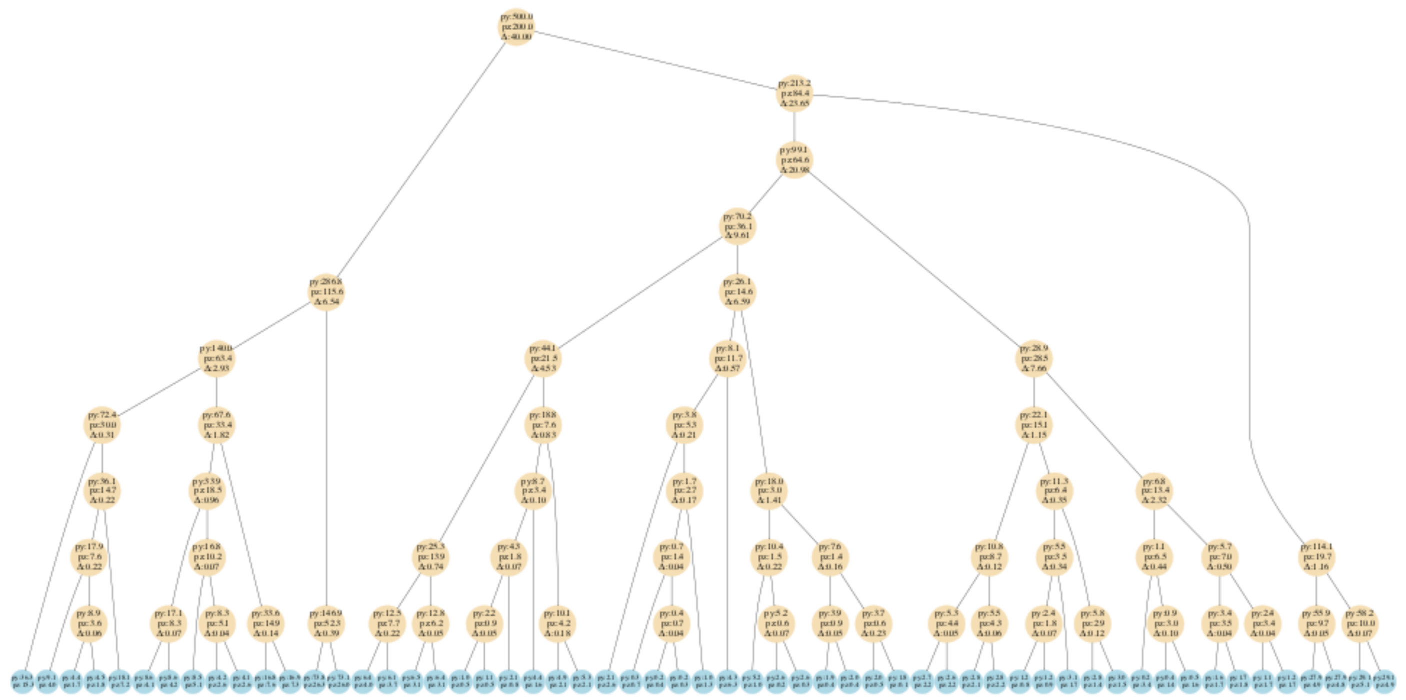
As a result of this *showering process*, there could be many latent paths that may lead to a specific jet (i.e. the set of constituents). Thus, it is natural and straightforward to represent a jet and the particular showering path that gave rise to it as a binary tree, where the inner nodes represent each of the unstable particles and the leaves represent the jet constituents.

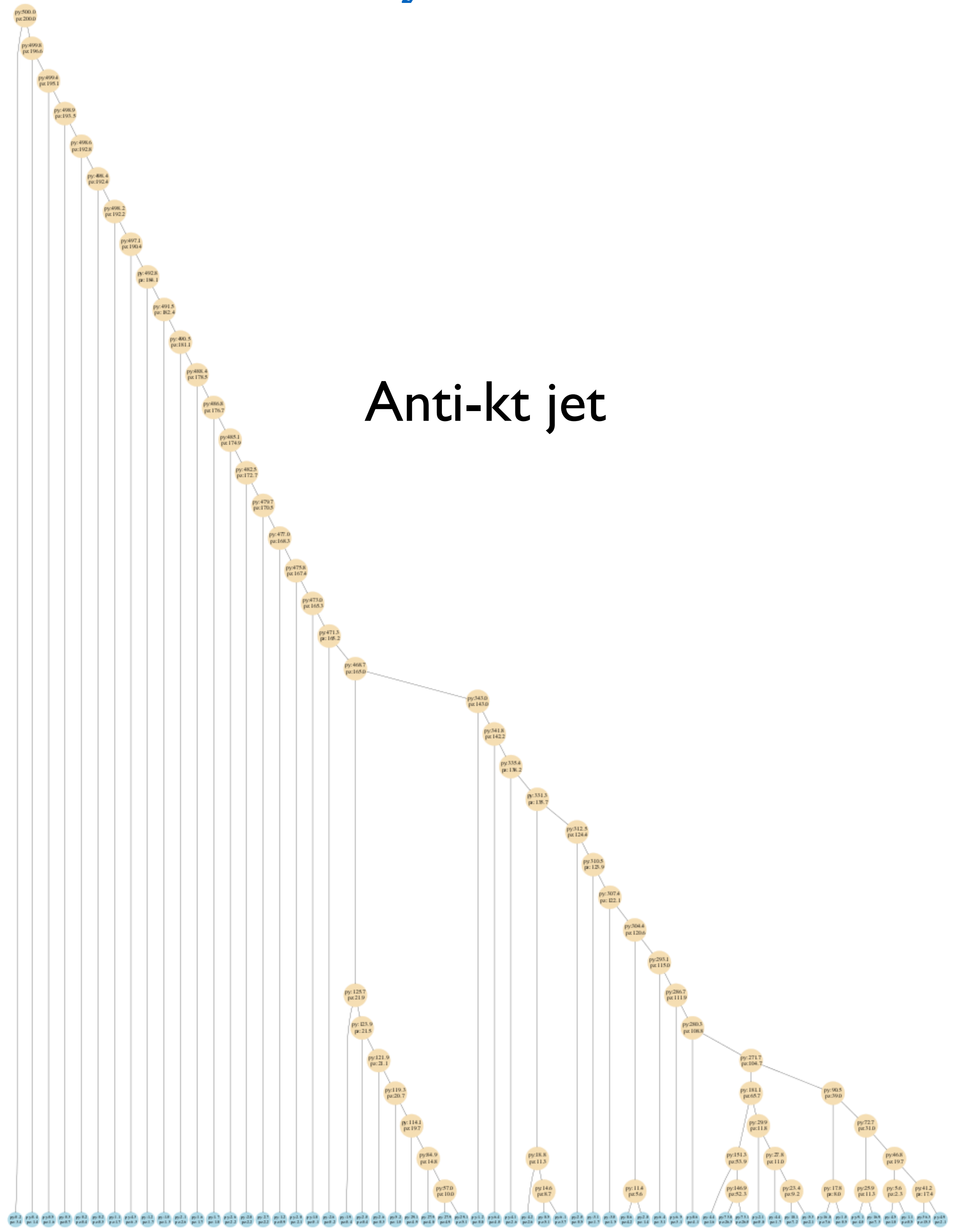
1

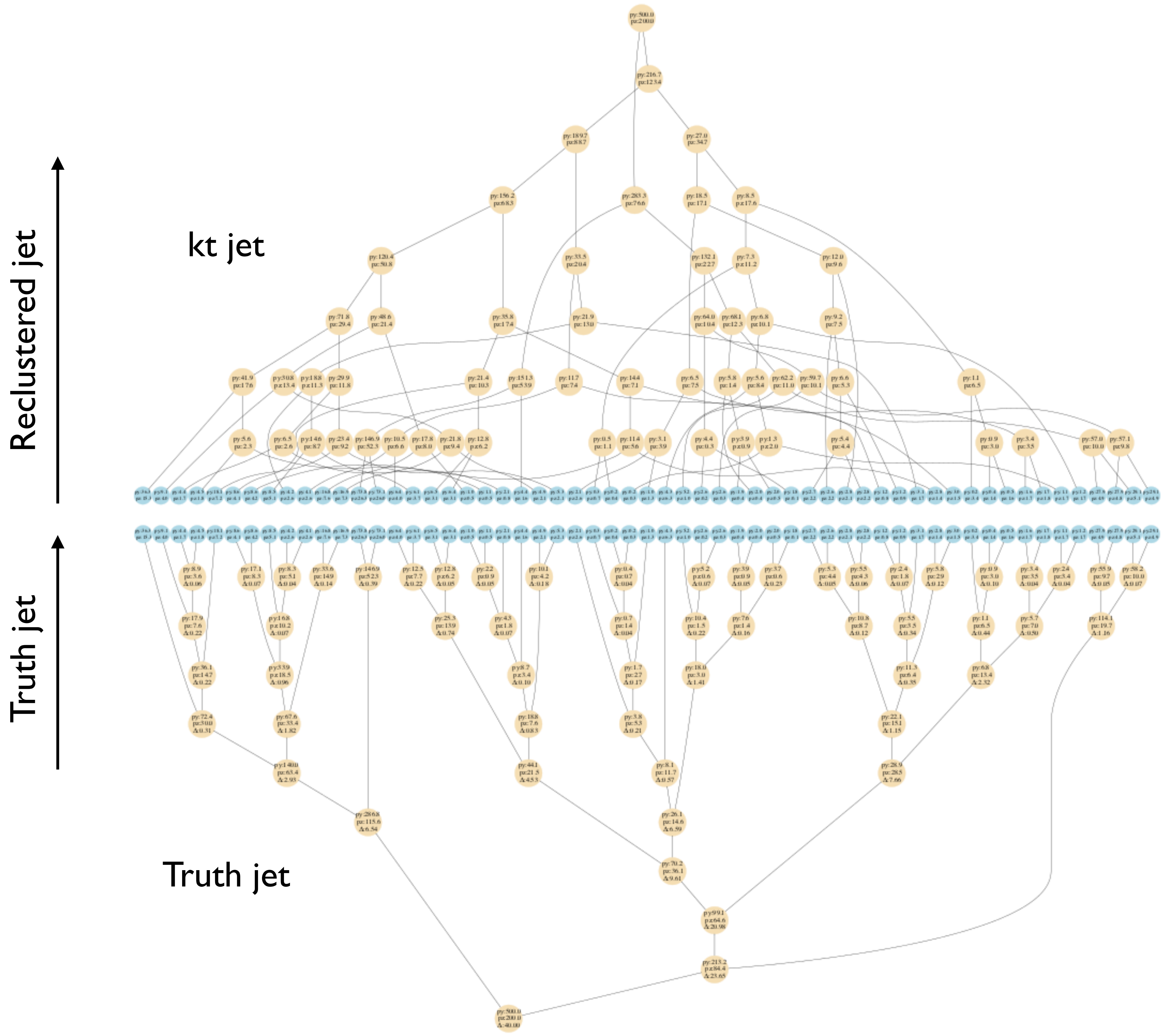
Binary Tree Visualizations

Kyle Cranmer, SM & Duccio Pappadopulo

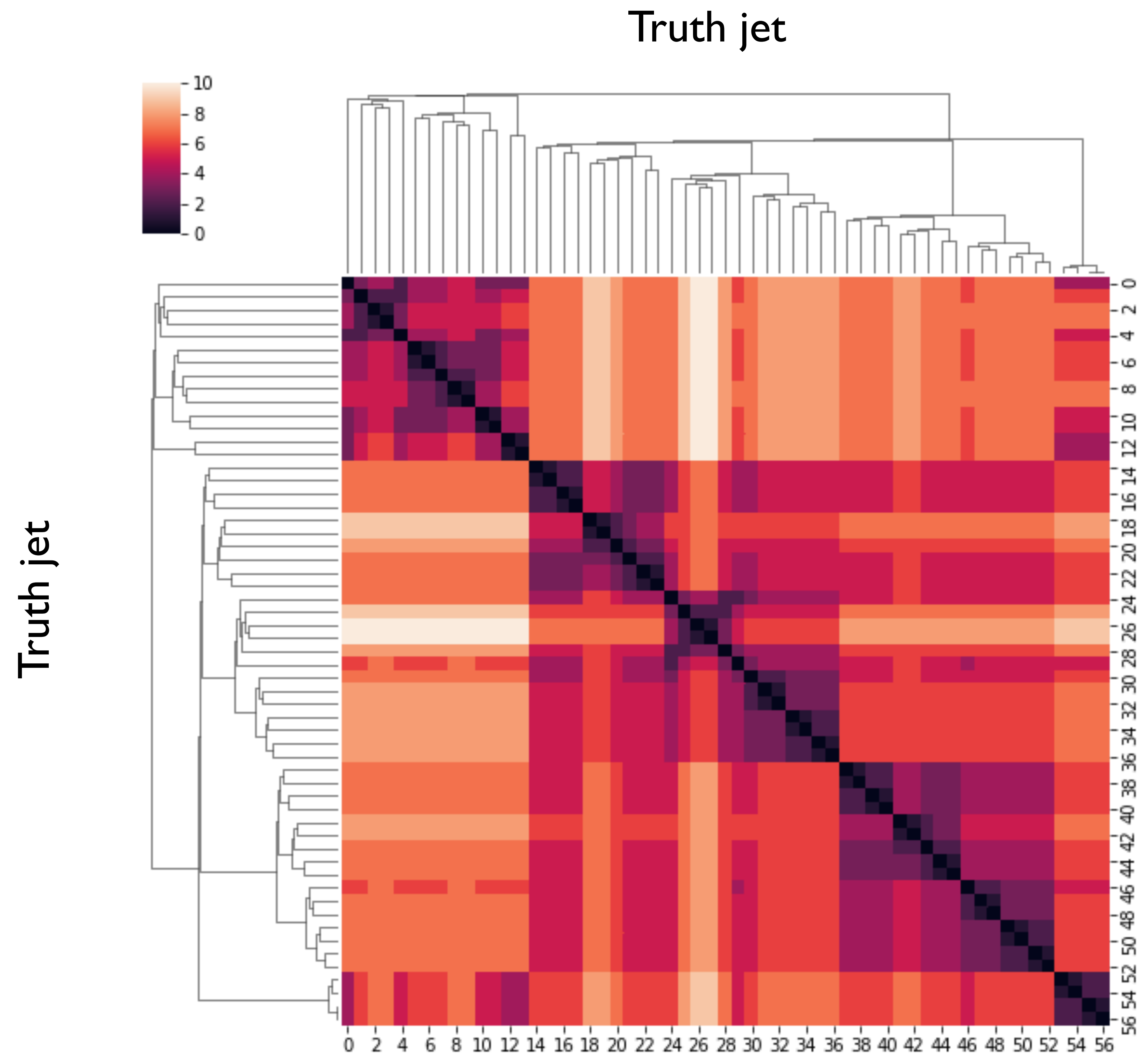
https://github.com/SebastianMacaluso/VisualizeBinaryTrees



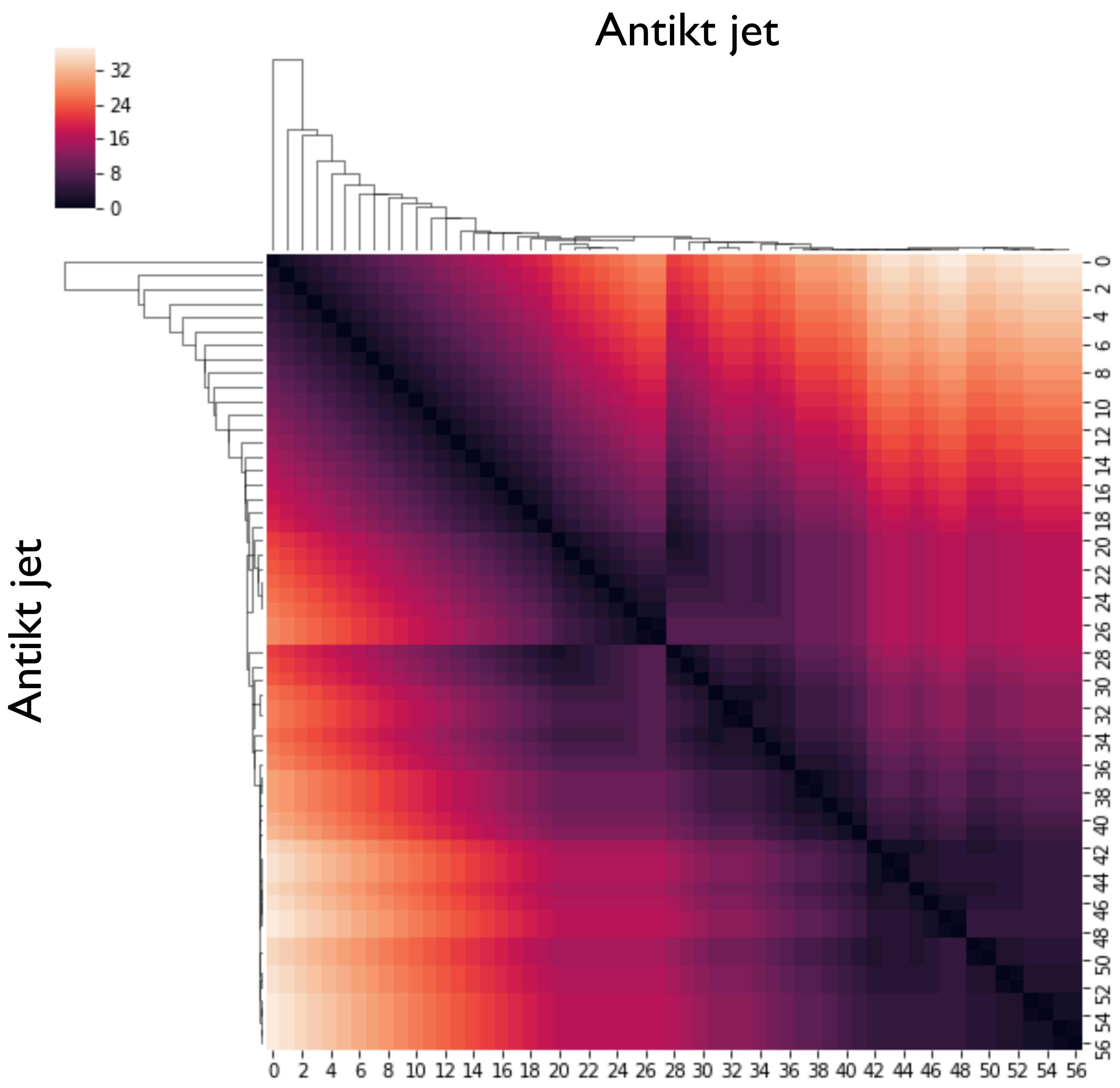



Reclustered jet
kt jet
Truth jet
Truth jet

Heat dendrogram cluster maps



Heat dendrogram cluster maps



Example Benchmark Task

Kyle Cranmer, SM & Duccio Pappadopulo

Assuming the toy model, attempt to find the most likely splitting history

- Reconstructing the splitting history is like inverting the generative model
- This is how we think of the kt algorithm
- **Hard**! There are a combinatorial number of histories to consider
- Usually jet reconstruction algorithms don't use the probability model directly
- We use the likelihood for the history as the optimization **objective** for benchmark

Reconsidering the kT algorithm

Generalized kt clustering algorithms

Current implementations of the generalized kt clustering algorithms sequentially choose the closest pair according to the d_{ij} distance measure.

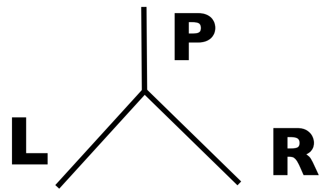
Intuitively, small d_{ij} means high likelihood the pair came from the same parent.

Viewed this way, kt is a greedy algorithm for finding most likely history.

Greedy likelihood-based algorithm

Choose the nodes pairing that locally maximizes the likelihood at each step given by

$$p(\Delta_{\mathrm{L}}, \Delta_{\mathrm{R}} | \Delta_{\mathrm{p}})$$

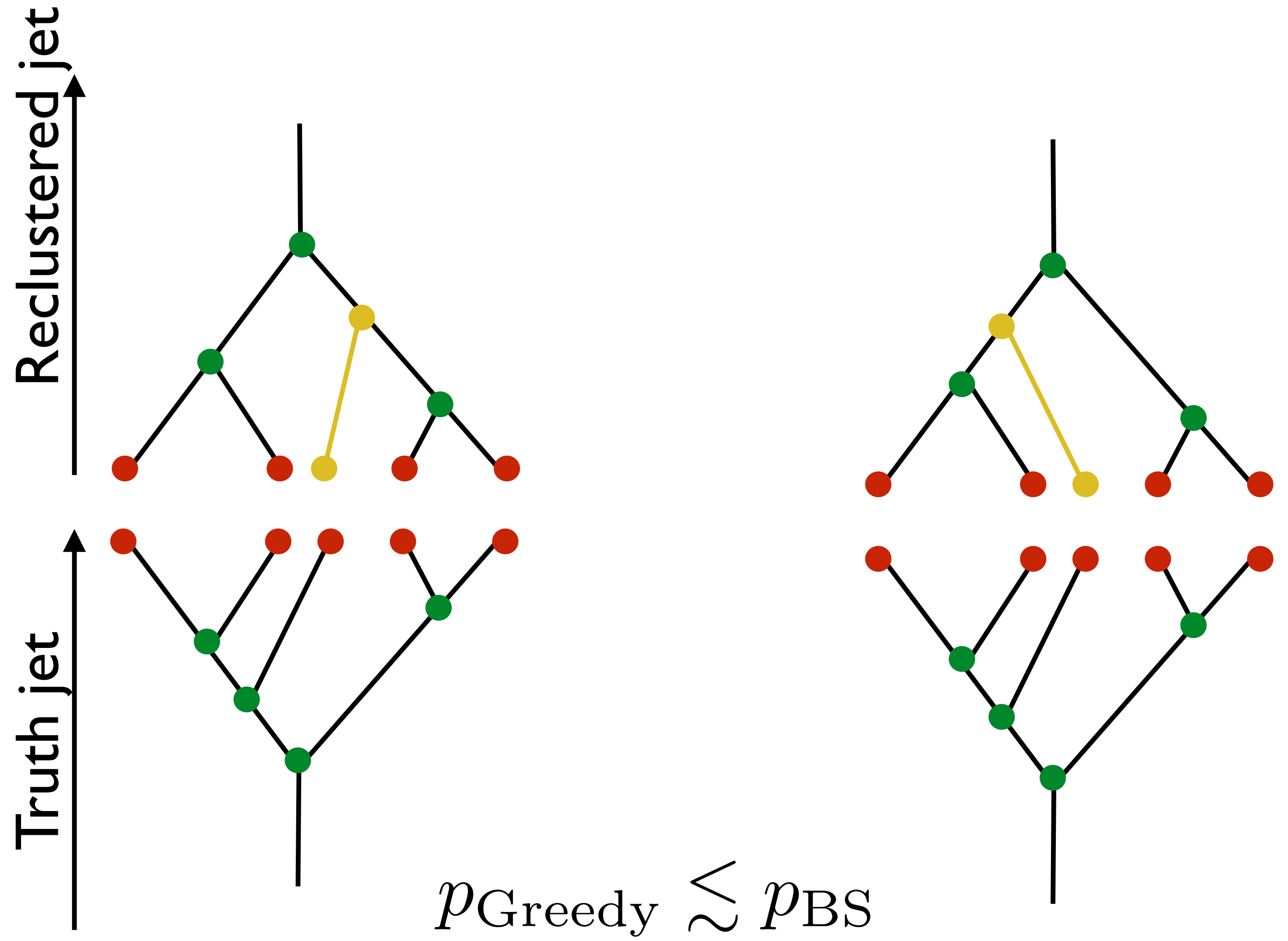


Our implementation improves the N^3 brute force time complexity to N^2 following the approach based on nearest neighbors introduced in FastJet [hep-ph/0512210, G. Salam and M. Cacciari LPTHE-06-02].

Likelihood-based clustering

Greedy
Joint likelihood from locally maximizing the likelihood at step.

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



$p_{\mathrm{Greedy}} \lesssim p_{\mathrm{BS}}$

Joint likelihood:

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

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

Given an algorithm, z is fixed.

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

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

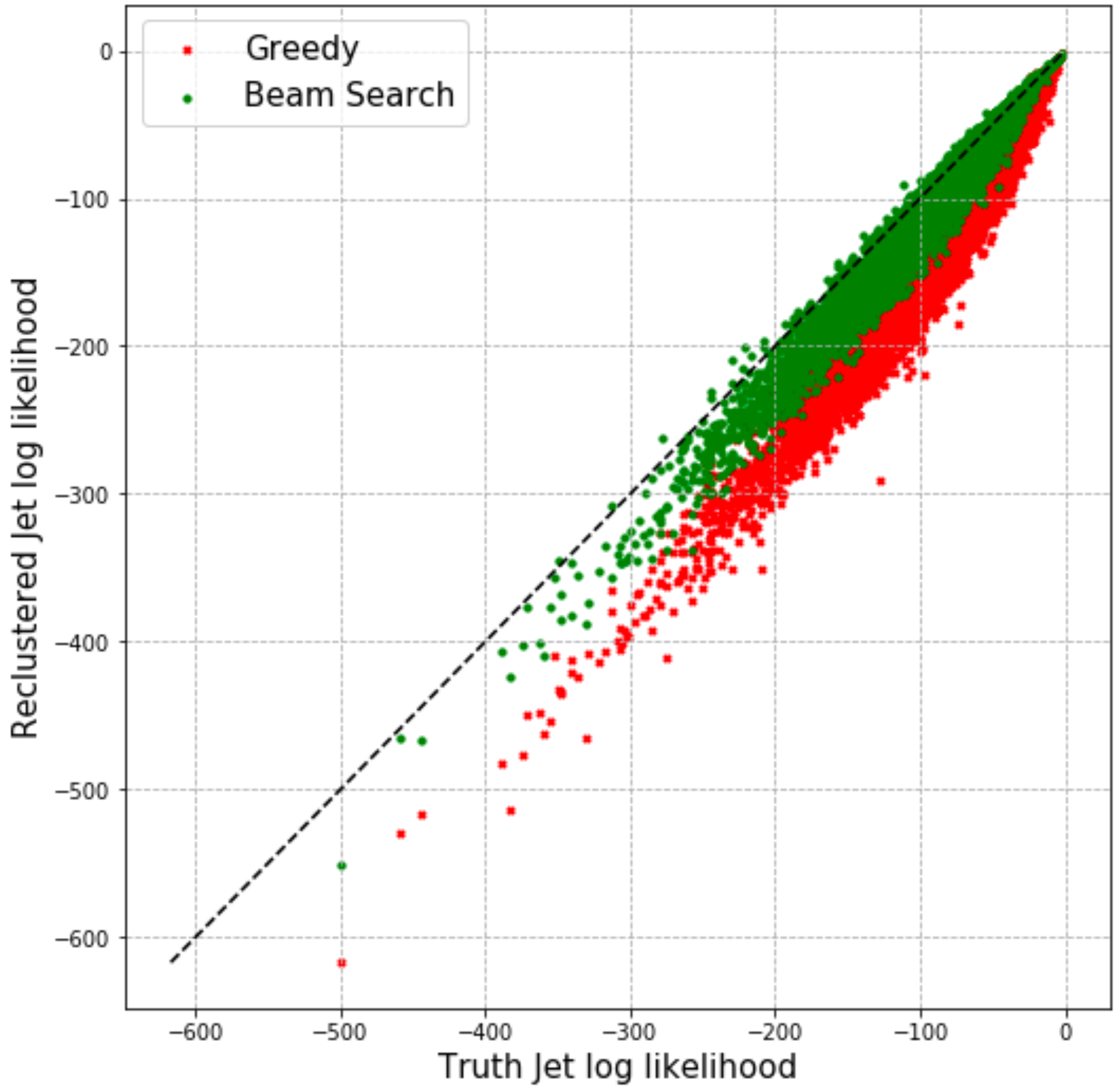
$\boxed{\hat{z}_{\mathrm{MLE}} = \mathrm{argmax}_z \; p(x|z)}$

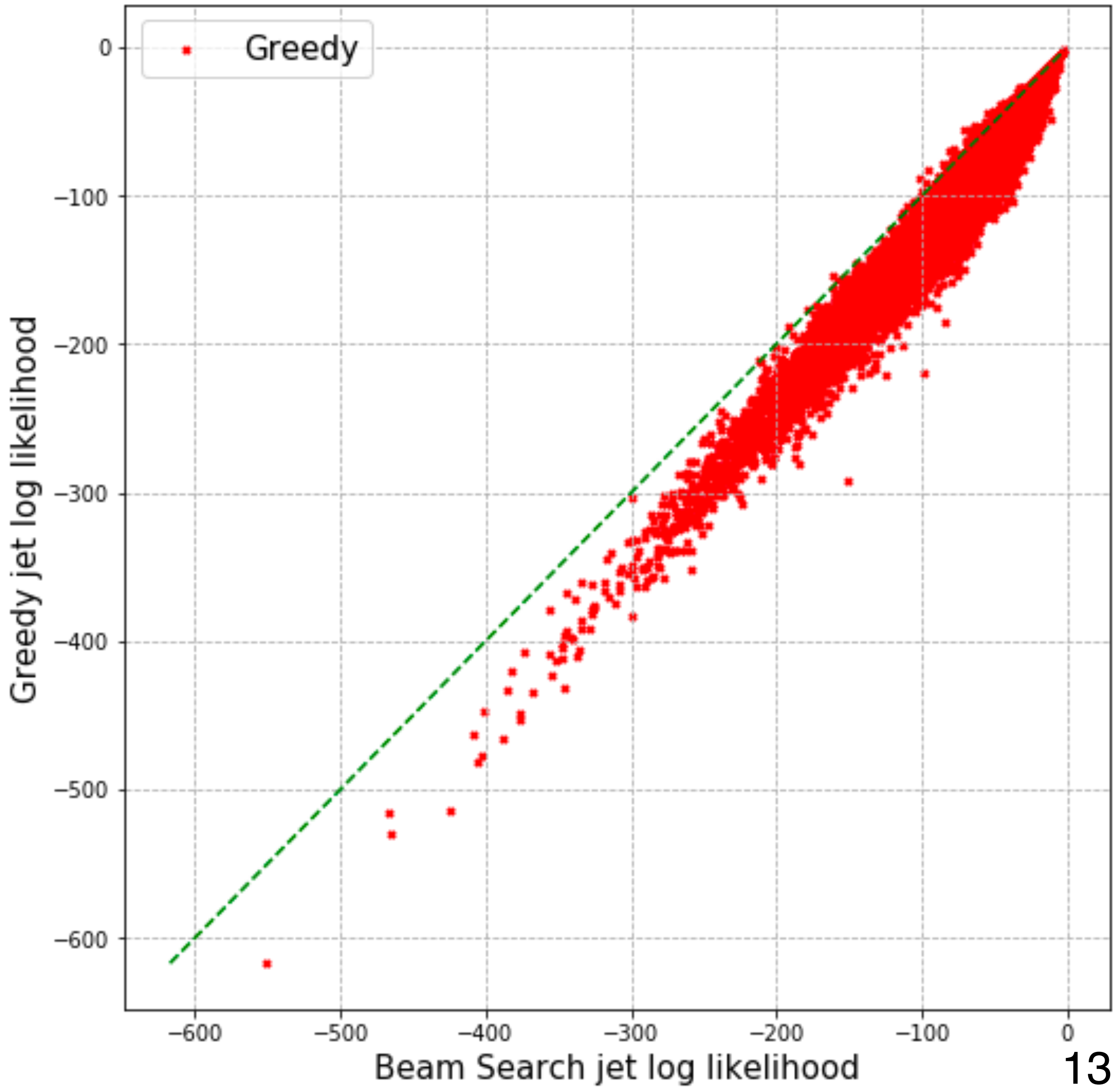
Viterbi algorithm
Computationally "infeasible" for jets with ~ 100 constituents

Jets log likelihood

Mean values:

$$\log p_{\text{Greedy}} = -111.2 \pm 3.1$$
$$\log p_{\text{BS}} = -85.7 \pm 2.6$$
$$\log p_{\text{Truth}} = -76.8 \pm 2.5$$





Final remarks

- Goal: Unification of generation and inference.
  - Future work: study the implementation of this technique on full physics simulators.
  - Potential collaboration with JetScape collaboration?
- Could we learn the node splitting from fitting to data and systematically improve the generative model?
- Study metrics to compare jet trees latent structures.
- Beam search is strictly better than the greedy algorithm (e.g. kt) in terms of estimating the most likely tree.
- Explore other possibilities for jet clustering algorithms and compare their performance.

Thanks for your attention!