Emerging techniques for sampling, searching, and summing over the combinatorially large space of shower histories

Johann Brehmer, Kyle Cranmer, Matthew Drnevich,
Sebastian Macaluso & Duccio Pappadopulo

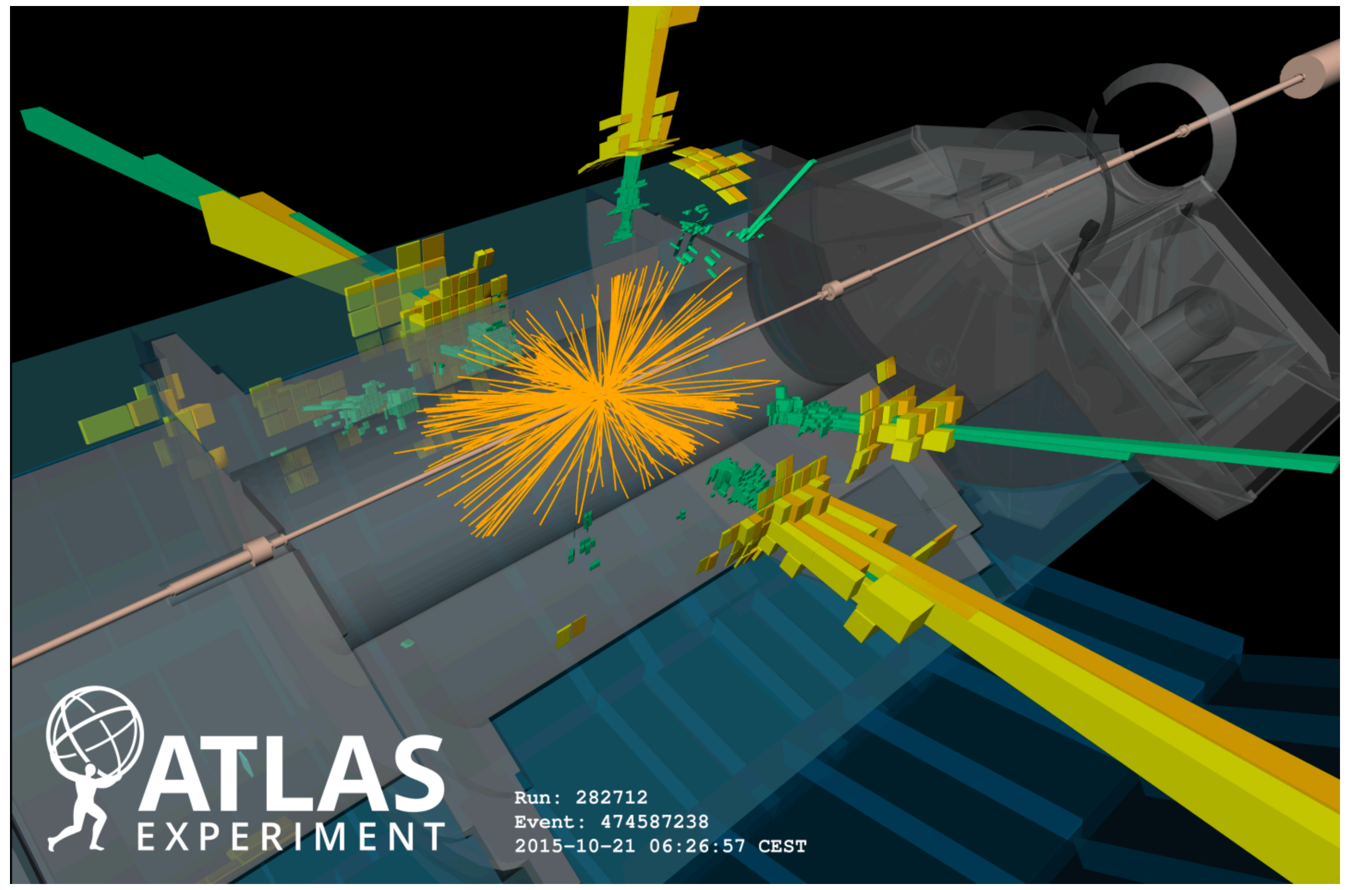
ML4Jets2021

July 6, 2021

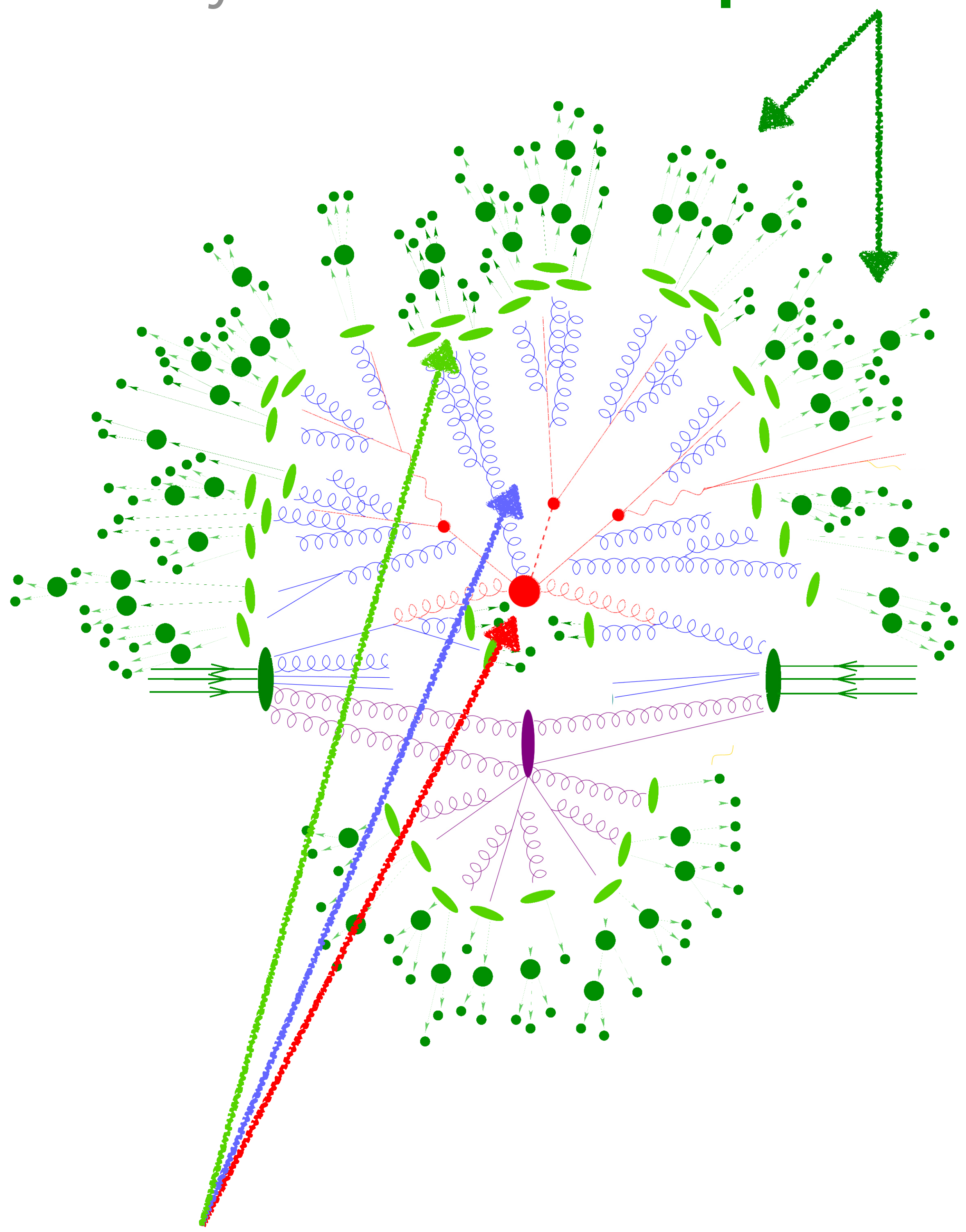
Monte Carlo event generator

- 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 final state particles (leaves).
- Many showering histories (trees) could give rise to the same set of leaves.

> It would be powerful if we could unify generation (the simulation / forward model) and inference (MC tuning, jet tagging, etc.).
> To do this, we need first to reframe jet physics in probabilistic terms.



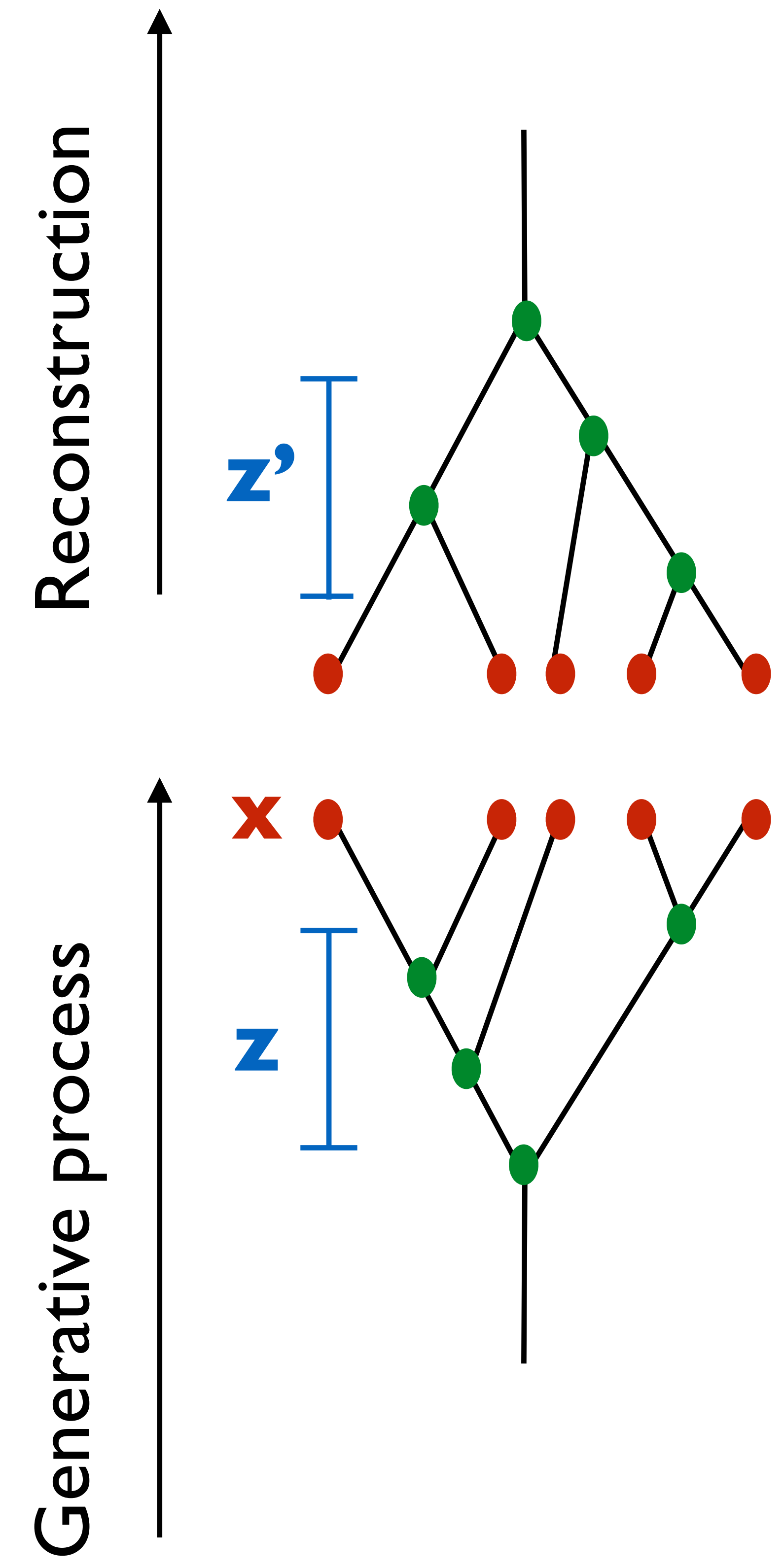
We only **observe** the **particles**



Evolution of the shower is **latent**

Jet Reconstruction

- The goal of jet reconstruction is to invert the generative process.
- Task: reconstruct the unobserved showering history (latent tree) z from observed particles x.
- In more general terms we want to find the truth level hierarchical clustering given a set of leaves (jet constituents) x.

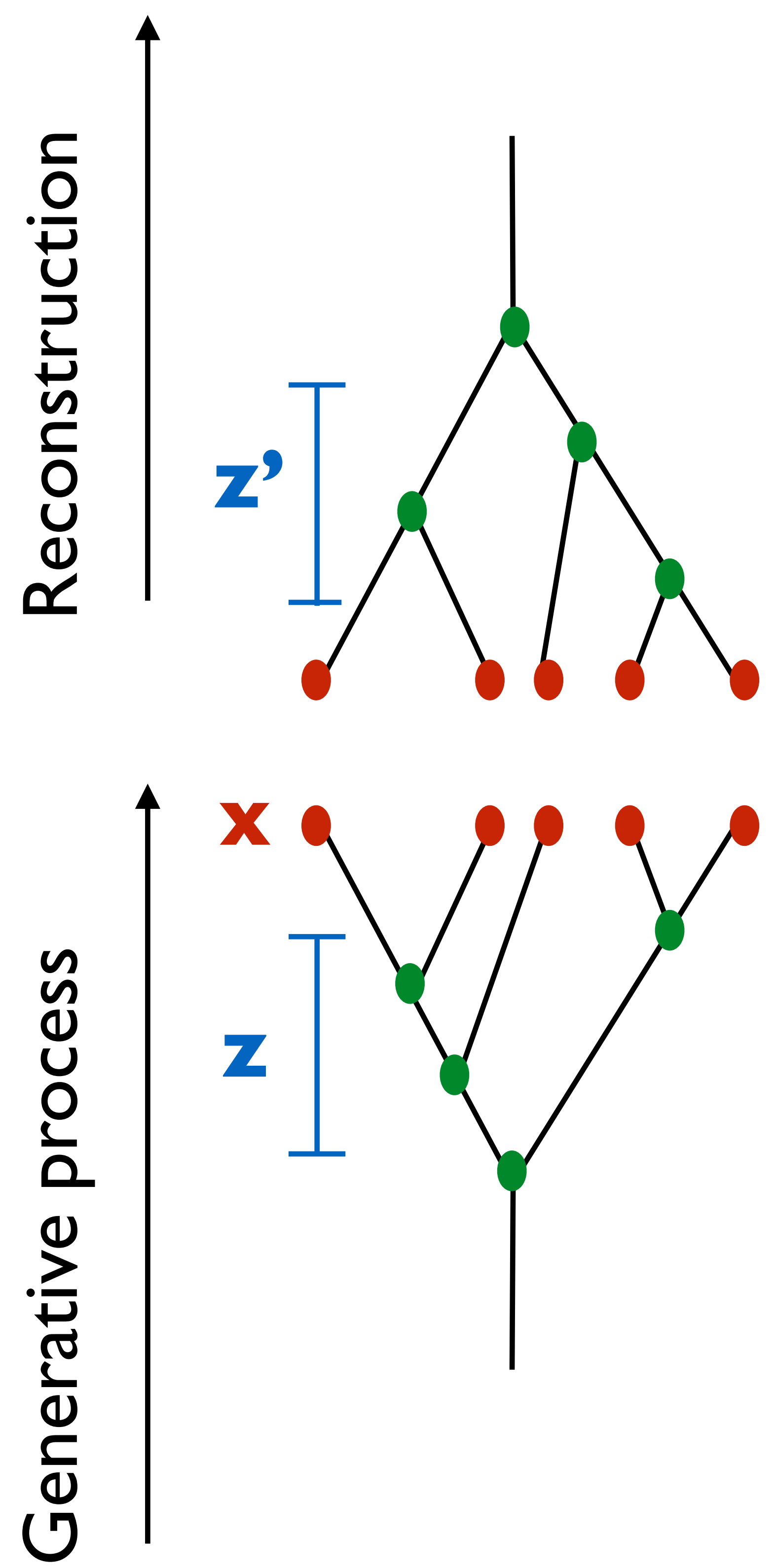


Hierarchical Clustering
Recursive partitioning of a data set into successively smaller clusters.

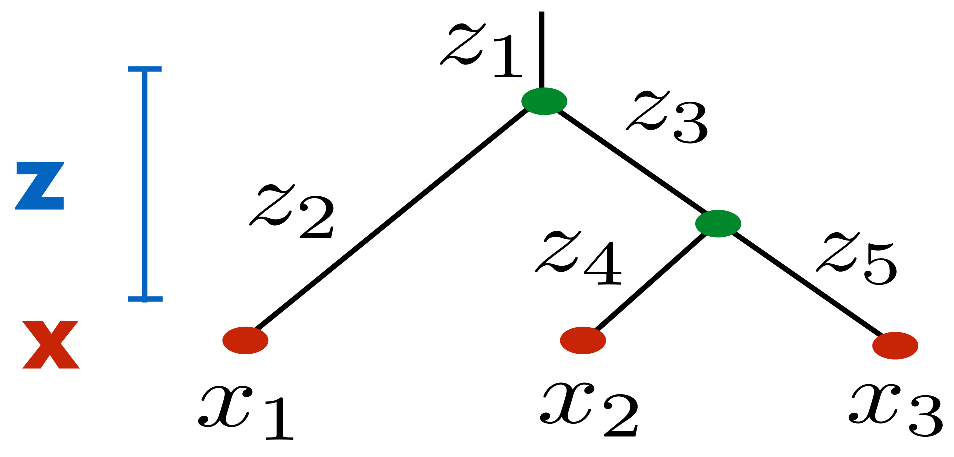
- Finding the truth level hierarchical clustering is not unique, because there are many possible showering histories that could give rise to the same leaves. But there is a notion of "best" or "most likely" hierarchy.

Reframing jet physics in probabilistic terms

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



Latent showering history



Joint likelihood:

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

Unification of generation and inference

NYU

Example tasks

- Maximum a posteriori (MAP) tree $\mathrm{ArgMax}_z\, p(z|x,\theta)$
  - Generative model describes the joint likelihood $p(x,z|\theta) \propto p(z|x,\theta)$
- Marginal likelihood

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

  - Maximum likelihood parameter $\mathrm{ArgMax}_\theta\; p(x|\theta)$ **Matthew Drnevich's talk!**
  - Likelihood ratio for different types of jets $\frac{p(x|H_1)}{p(x|H_0)}$ **Lauren Greenspan's talk!**

Parton Showers generators

PYTHIA

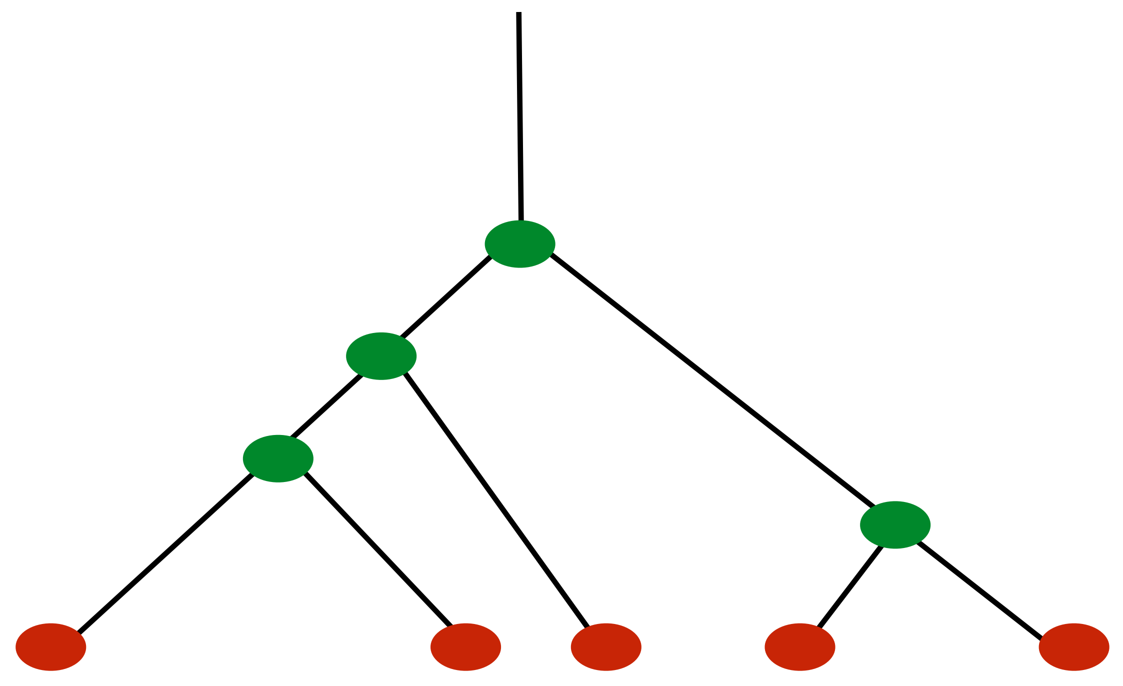
Sherpa

Herwig 7

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

- Other complications related to momentum conservation that could be addressed [Bauer & Tackmann '08].

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



To prototype we built our own simplified model!

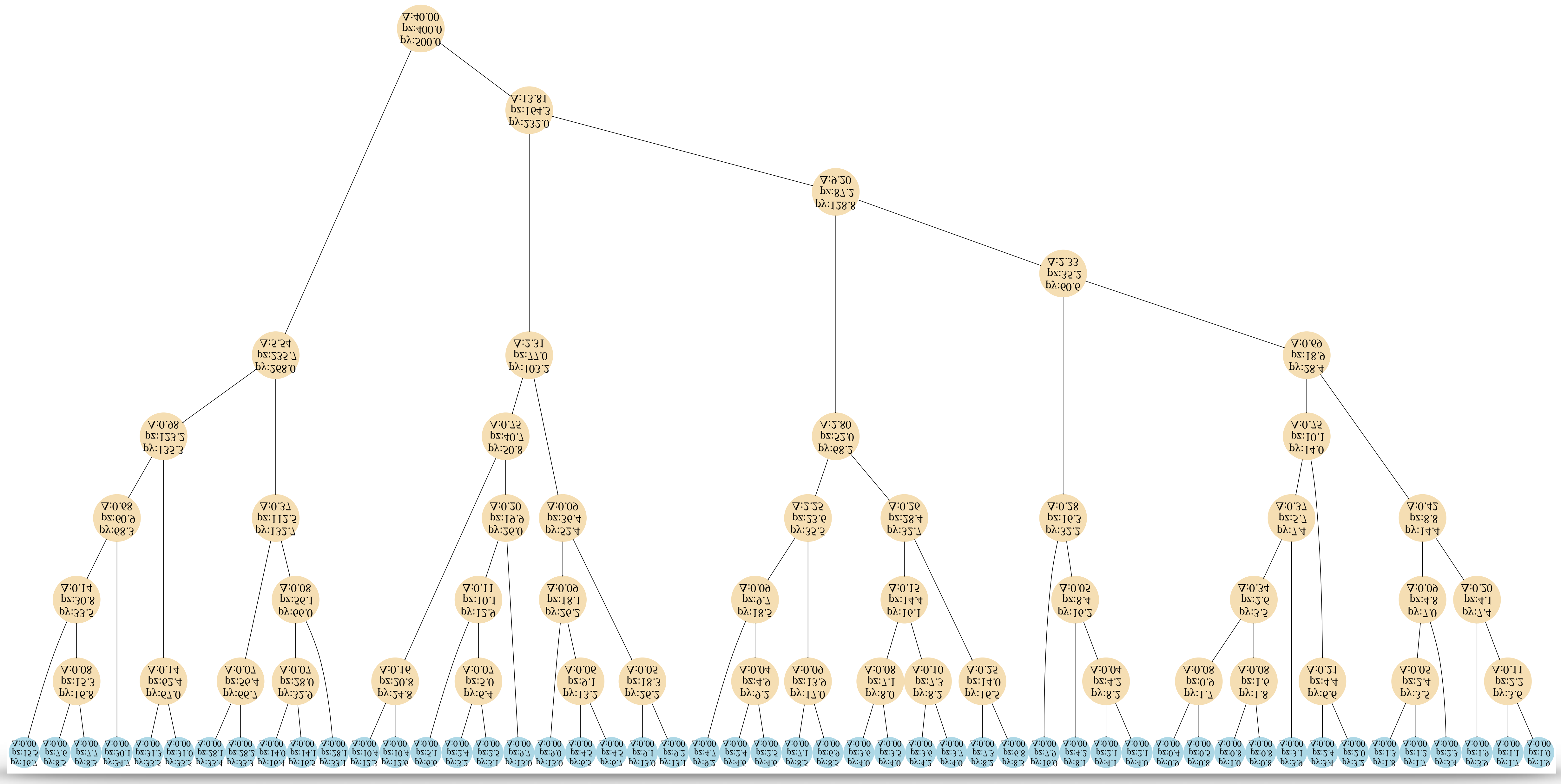
Ginkgo: Simplified Generative Model for Jets

vCHEP2021 [Cranmer, Drnevich, Macaluso, Pappadopulo, arXiv:2105.10512]

NYU

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.

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

- Marginalize over showering histories (binary trees).
- 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 $(p^{\mu}_{\mathrm{p}}, t_{\mathrm{P}}, t_{\mathrm{cut}}, \lambda, \mathrm{tree})$

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 $(p^{\mu}_{\mathrm{p}}, t_{\mathrm{L}}, t_{\mathrm{cut}}, \lambda, \mathrm{tree})$

    NodeProcessing $(p^{\mu}_{\mathrm{p}}, t_{\mathrm{R}}, t_{\mathrm{cut}}, \lambda, \mathrm{tree})$

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

Features

- Momentum conservation.
- Running of the splitting scale.

Facilitates research with:

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

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

Jet Reconstruction: Generalized kt clustering algorithms

- Idea: **sequentially cluster** jet constituents aiming to recover the showering history.
- Bottom-up (agglomerative) clustering.
- Merge closest pair based on a distance measure.
- Intuitively, particles with smaller d_{ij} have a **greater likelihood** to have come from a common parent.

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

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

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

kt-like clustering algorithms use a **heuristic**, with no explicit connection to the generative model.

The anti-k_t jet clustering algorithm #1

Matteo Cacciari (Paris, LPTHE), Gavin P. Salam (Paris, LPTHE), Gregory Soyez (Brookhaven) (Feb, 2008)

Published in: *JHEP* 04 (2008) 063 • e-Print: 0802.1189 [hep-ph]

pdf DOI cite 8,011 citations

FastJet User Manual #1

Matteo Cacciari (Paris, LPTHE and Diderot U., Paris), Gavin P. Salam (CERN and Princeton U. and Paris, LPTHE), Gregory Soyez (Saclay, SPhT) (Nov, 2011)

Published in: *Eur.Phys.J.C* 72 (2012) 1896 • e-Print: 1111.6097 [hep-ph]

pdf DOI cite 4,168 citations

Improving over sequential (agglomerative) clustering

https://github.com/SebastianMacaluso/StandardHC

- In the probabilistic language, generalized kt algorithms are greedy.

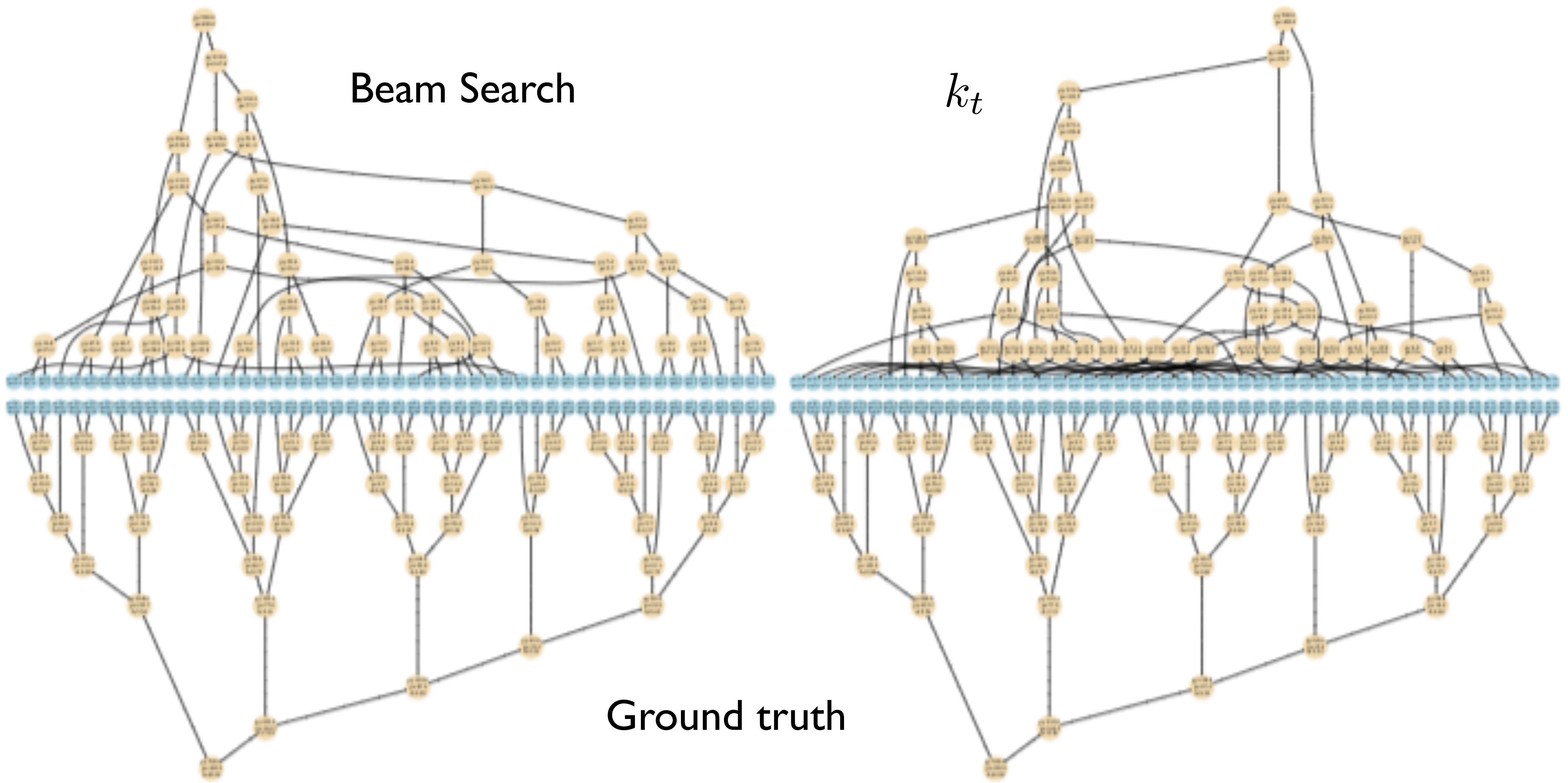
Greedy algorithms are analogue to playing chess only thinking one move at a time.

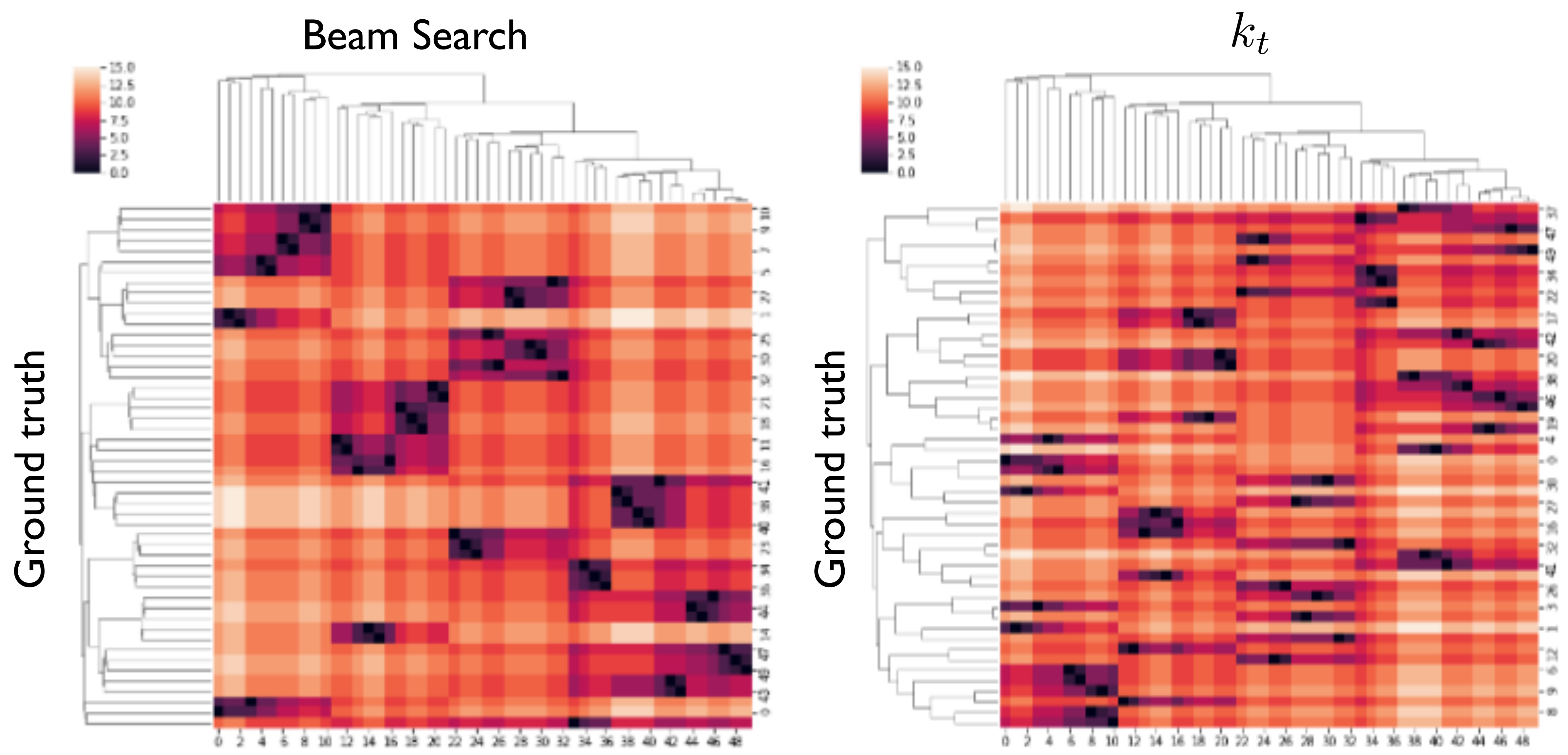
Straightforward improvements:

- Use the splitting likelihood encoded in the generative model instead of heuristics.
- Splitting likelihood gives a natural way to score the combination of multiple clusterings, i.e. their product. This allows to explore other algorithms, e.g. beam search.

Beam Search: keeps multiple possible clusterings in memory before choosing the showering history.

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





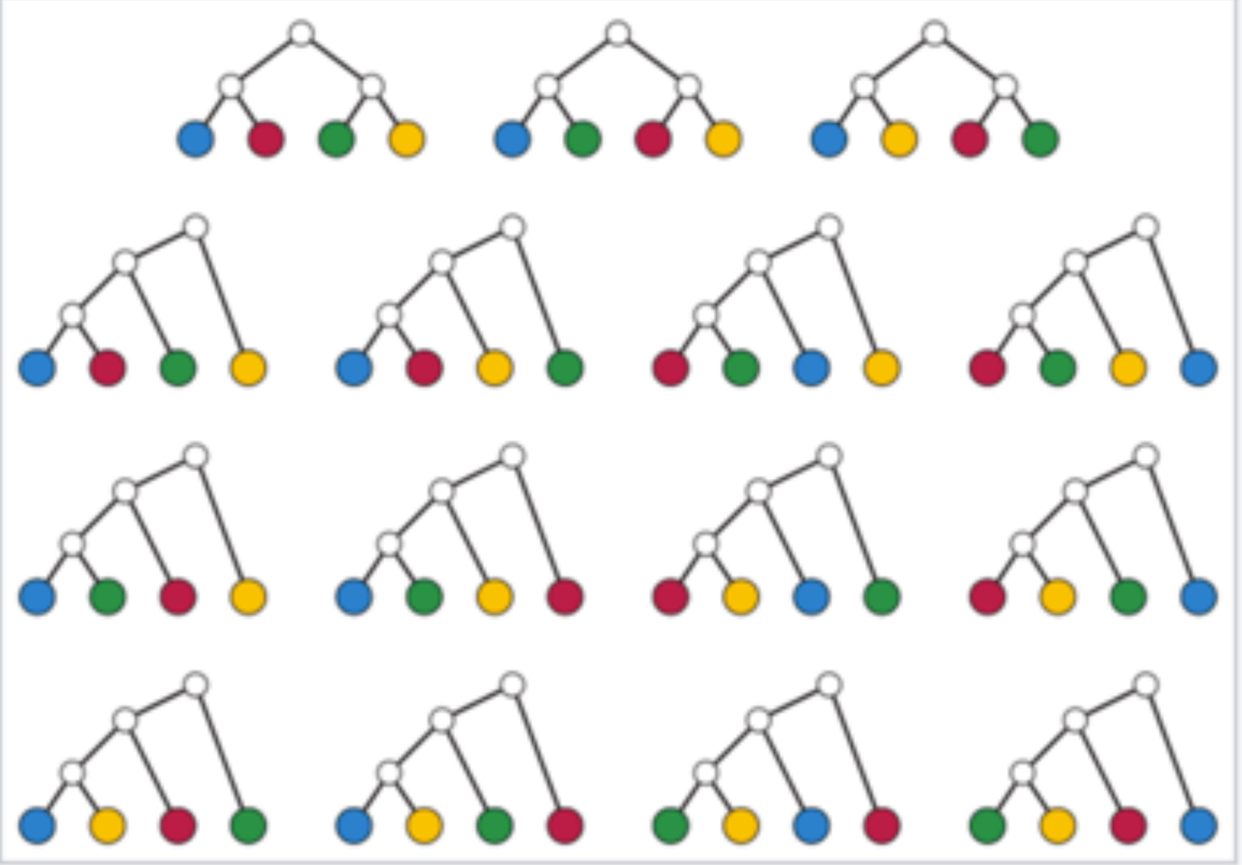
https://github.com/SebastianMacaluso/VisualizeBinaryTrees

Can we find the exact ML showering history or sum over all of them?

We aim to invert the process probabilistically

- Reconstructing the showering history is like inverting the generative model.
- The number of clustering histories is enormous! It grows like (2N-3)!! (times $2^{(N-1)}$ permutations), with N the number of jet constituents.
- Traditional jet reconstruction algorithms don't use the probability model directly.
- We use the likelihood of the showering 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 × 4 − 3)!! (see article text).

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

Hierarchical Cluster Trellis

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

NYU

We connected with Andrew McCallum's group at UMass

Craig Greenberg

Nicholas Monath

https://github.com/SebastianMacaluso/ClusterTrellis

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

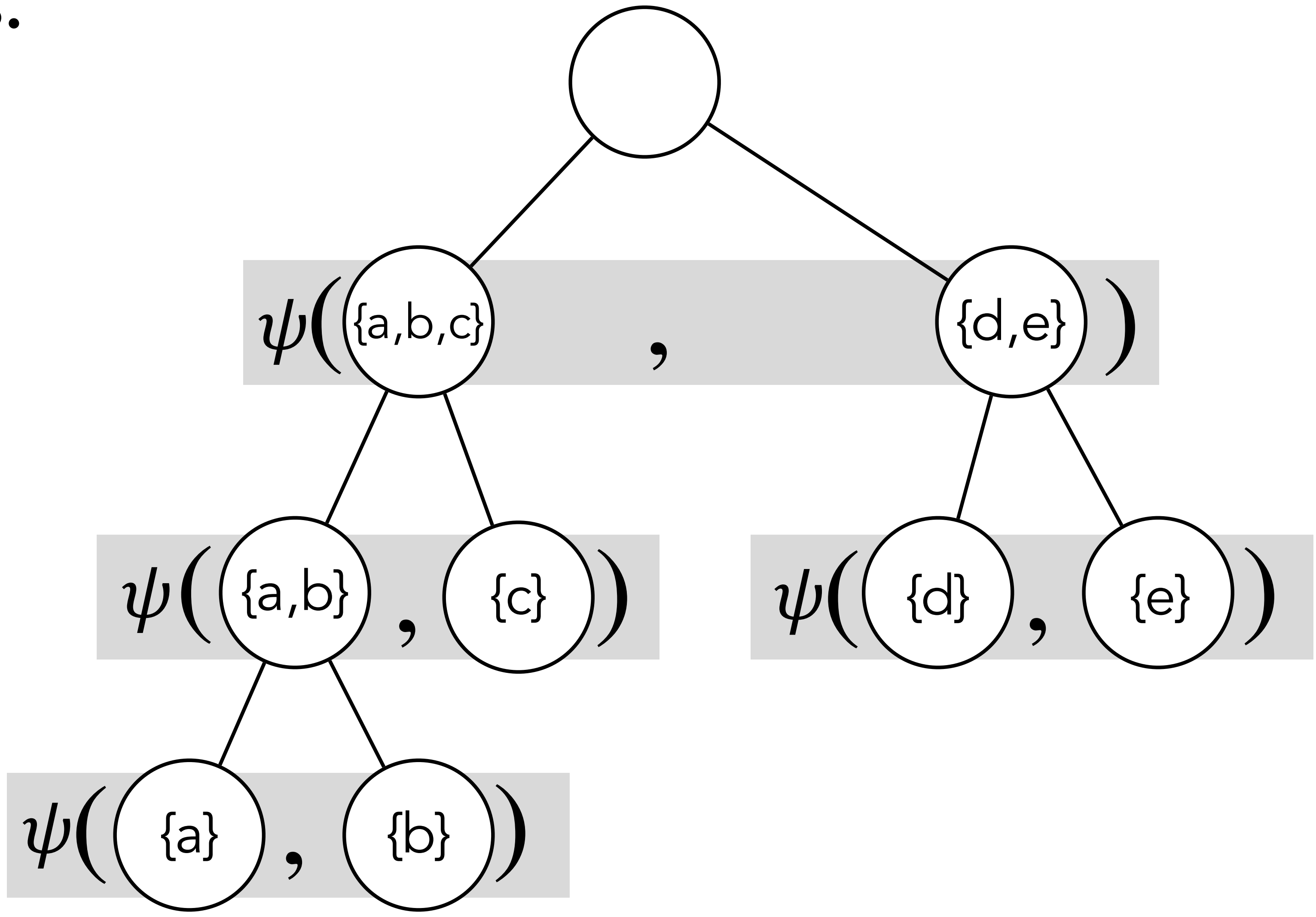
  Most meaningful, though we typically focus on the ML history.
  See **Matthew Drnevich** and **Lauren Greenspan**'s talks for implementation examples.

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

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

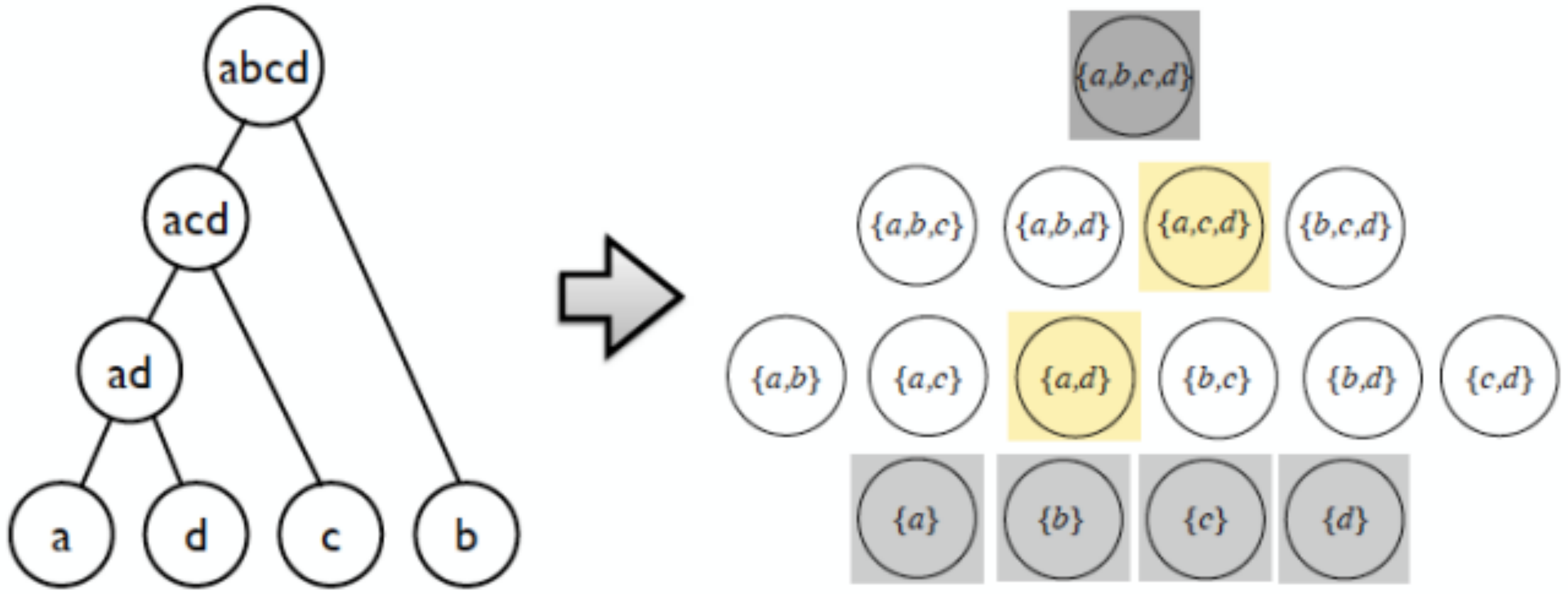
$$Z(X) = \sum_{\mathtt{H} \in \mathcal{H}(X)} \phi(X|\mathtt{H}).$$

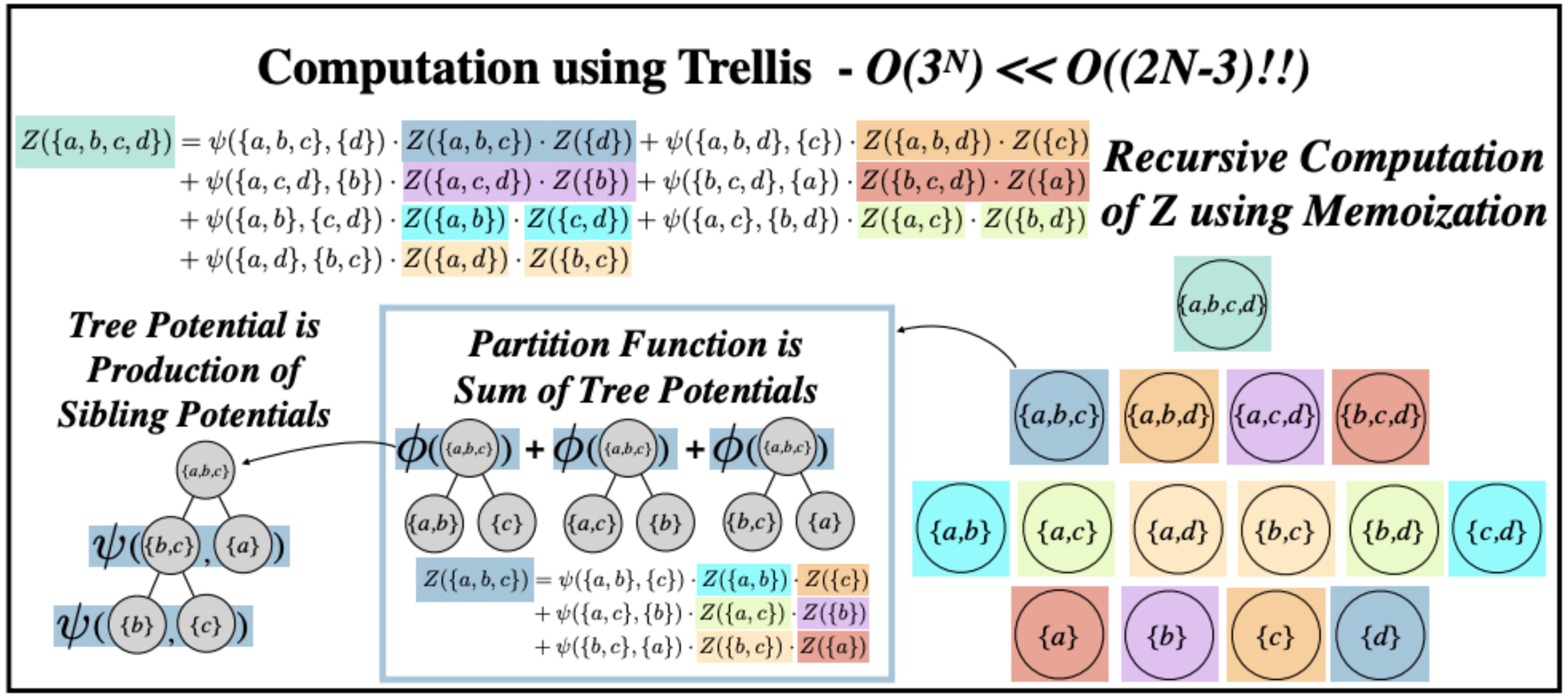
Marginal likelihood

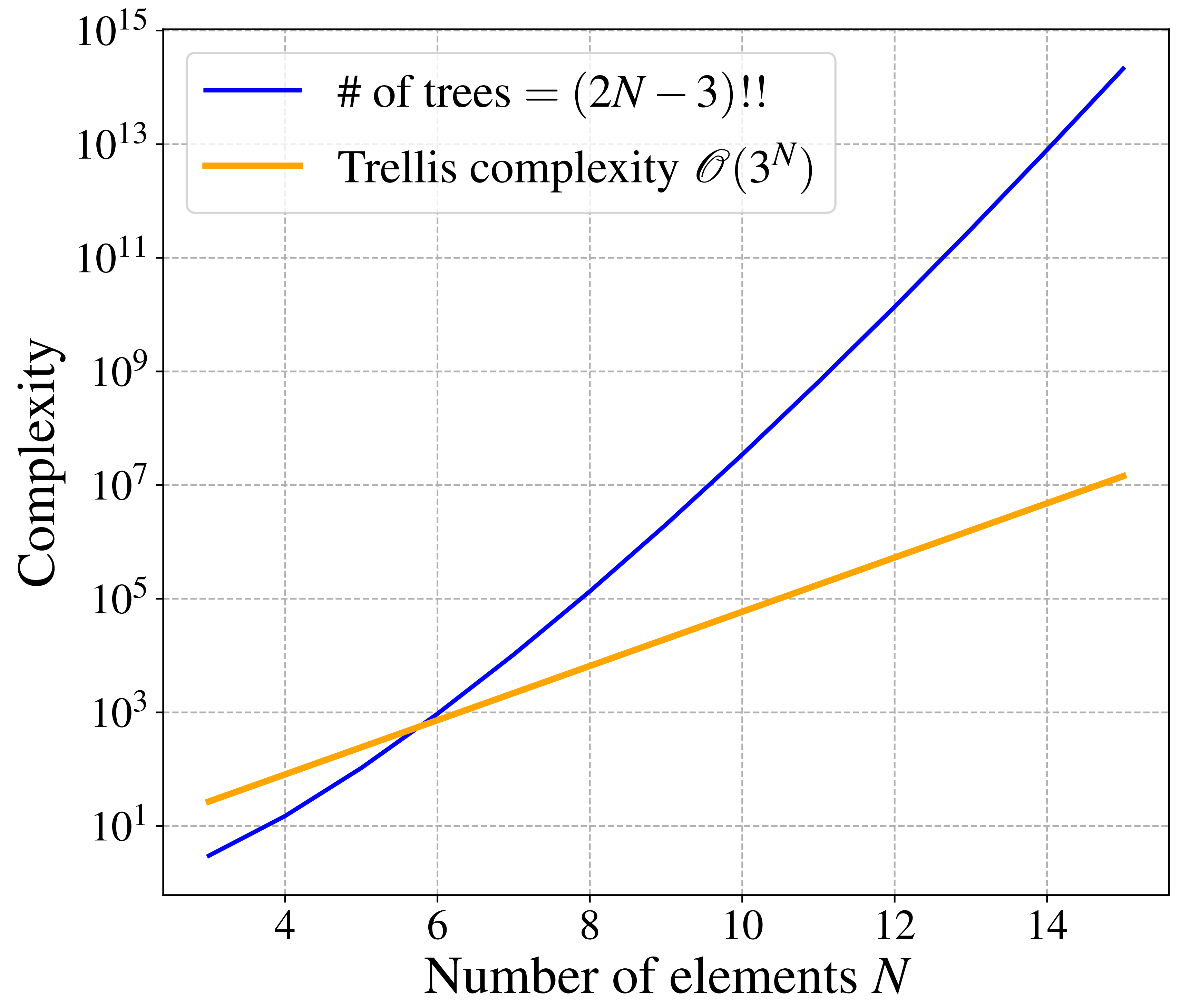


Trellis Structure

- 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 faster.
- It allows to run a dynamic programming algorithm to compute the marginal likelihood (over all possible clusterings) and the exact maximum likelihood hierarchy.

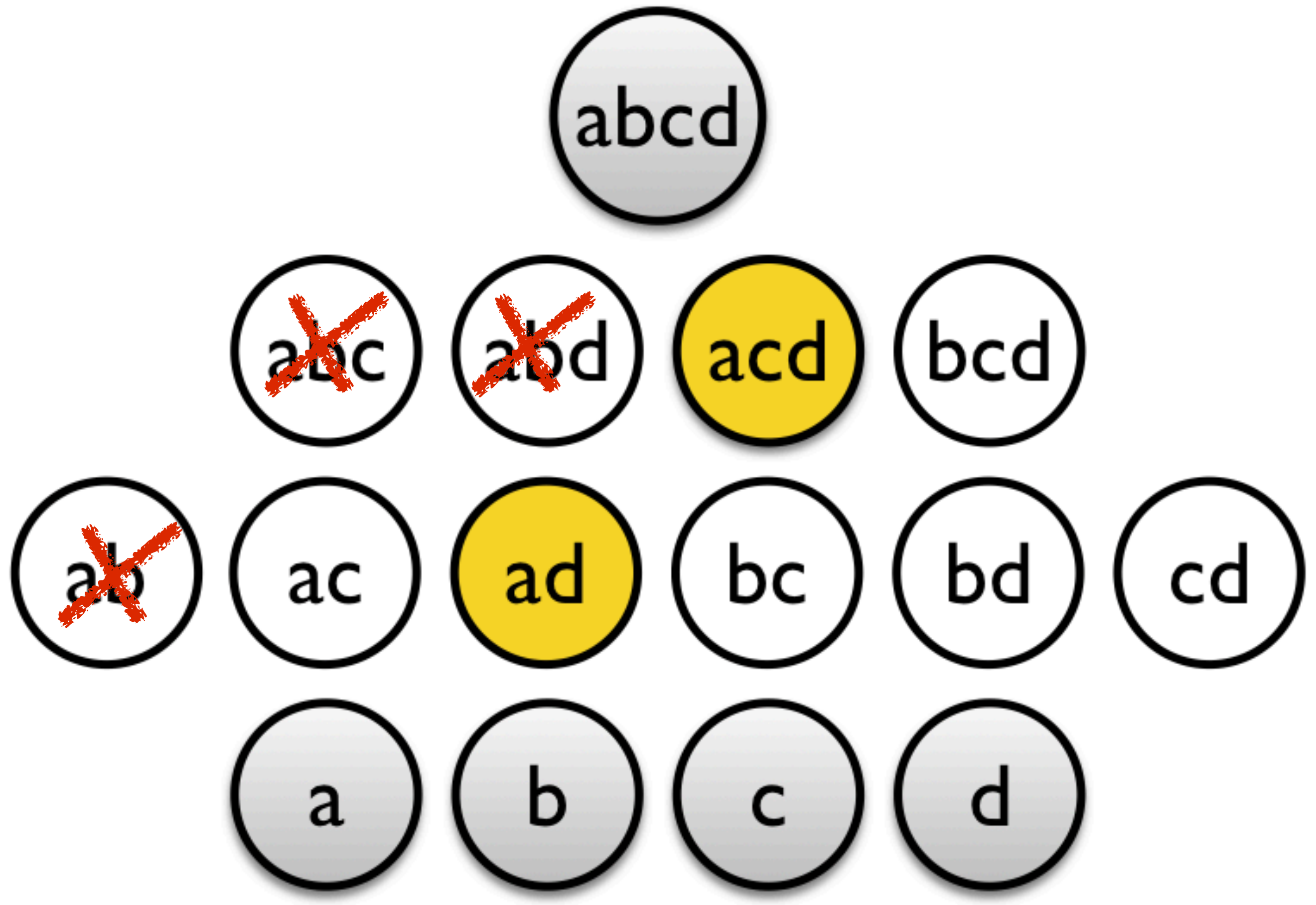






Sparse Hierarchical Cluster Trellis

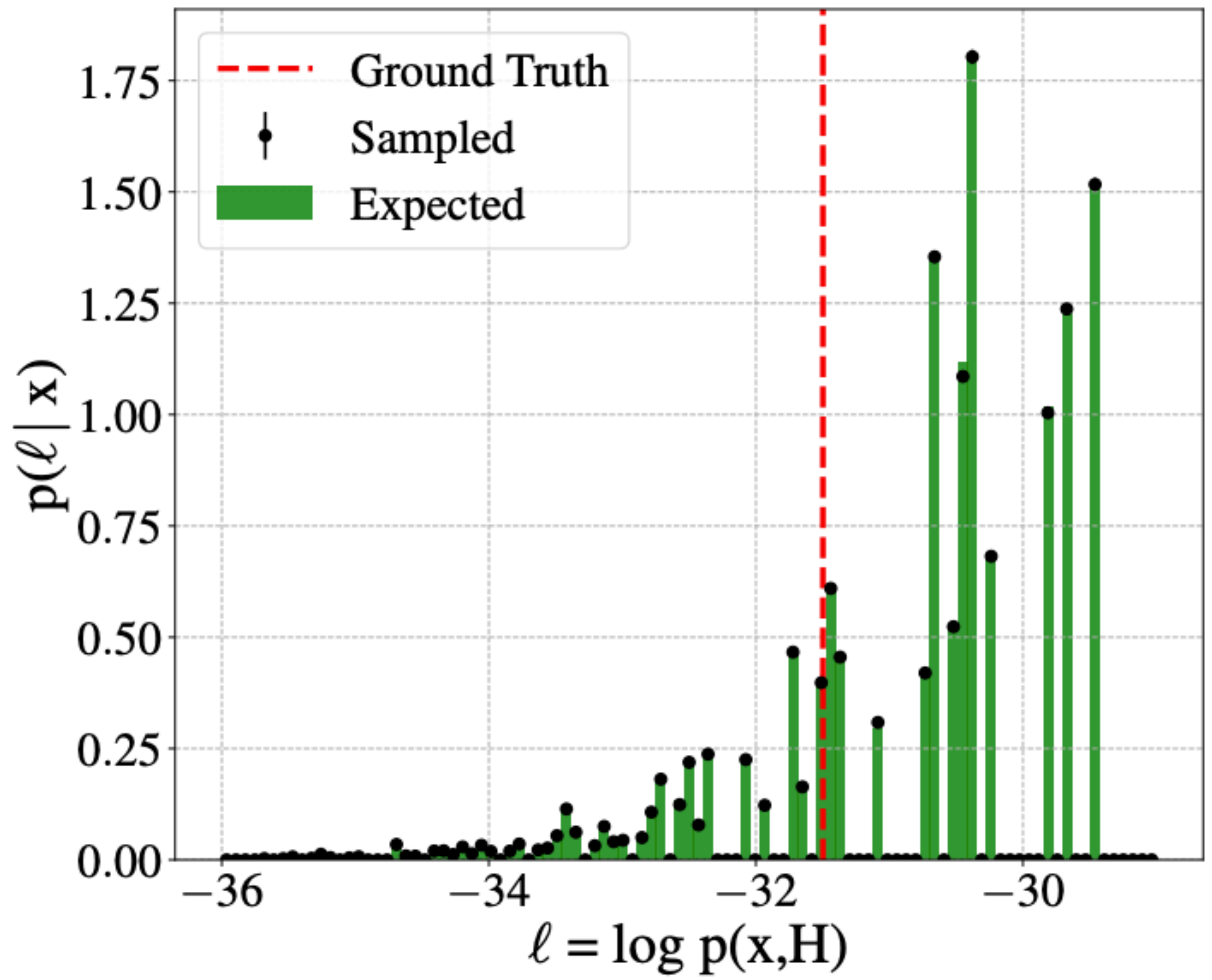
- Despite efficiency, the full trellis still grows exponentially.
- However, we can control complexity by building a sparse trellis with some nodes removed; we consider a fraction of hierarchies from the total of (2 N - 3)!!.
- Most histories likelihood is negligible compared to the maximum likelihood history (MAP tree).
- 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.
- The trellis enables to sample histories weighted by their likelihood from the true posterior distribution conditioned on a set of leaves (jet constituents) without enumerating all possible clusterings.

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



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.

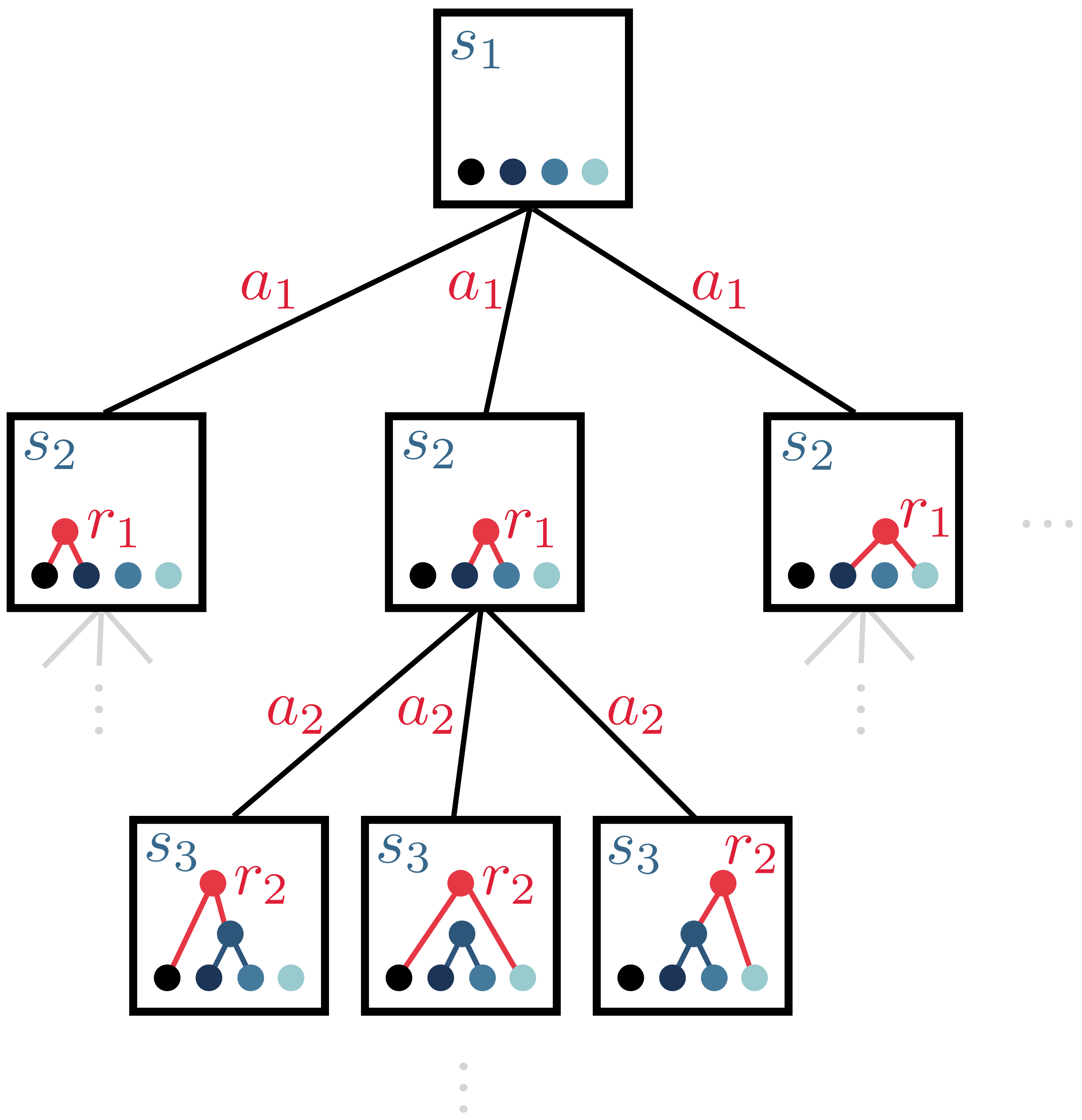
Trellis could be extended to consider 2 → 3 splittings (currently based on binary trees, 1 → 2 splittings); could make feasible the implementation of CKKW-L to a higher jet multiplicity.

Hierarchical Clustering as a Markov Decision Process

Brehmer, Macaluso, Pappadopulo, Cranmer [arXiv:2011.08191]
NeurIPS 2020 ML4Physical Sciences workshop

We explored the use of Reinforcement Learning

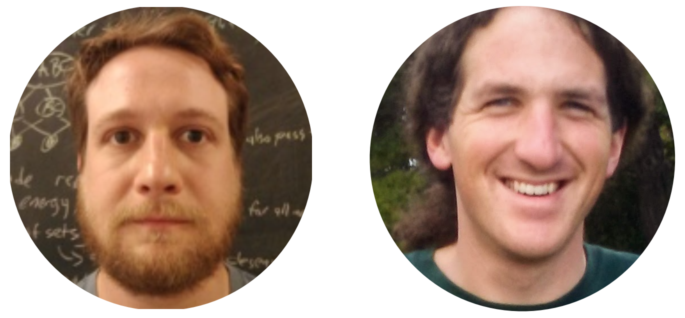
- The state space ***S*** is given by all possible particle sets at any given point during the clustering process.
- The actions ***A*** are the choice of two particles to be merged.
- The state transitions ***P*** are deterministic and update z_t to z_{t-1}
- The rewards ***R*** are the splitting probabilities.
- The MDP is episodic and terminates when only a single particle is left.



We implement Monte Carlo Tree Search (MCTS) and Imitation learning / Behavioral Cloning (BC).

A* search + trellis

UAI 2021 [Greenberg, Macaluso, Monath, Cranmer, McCallum, et al '21]

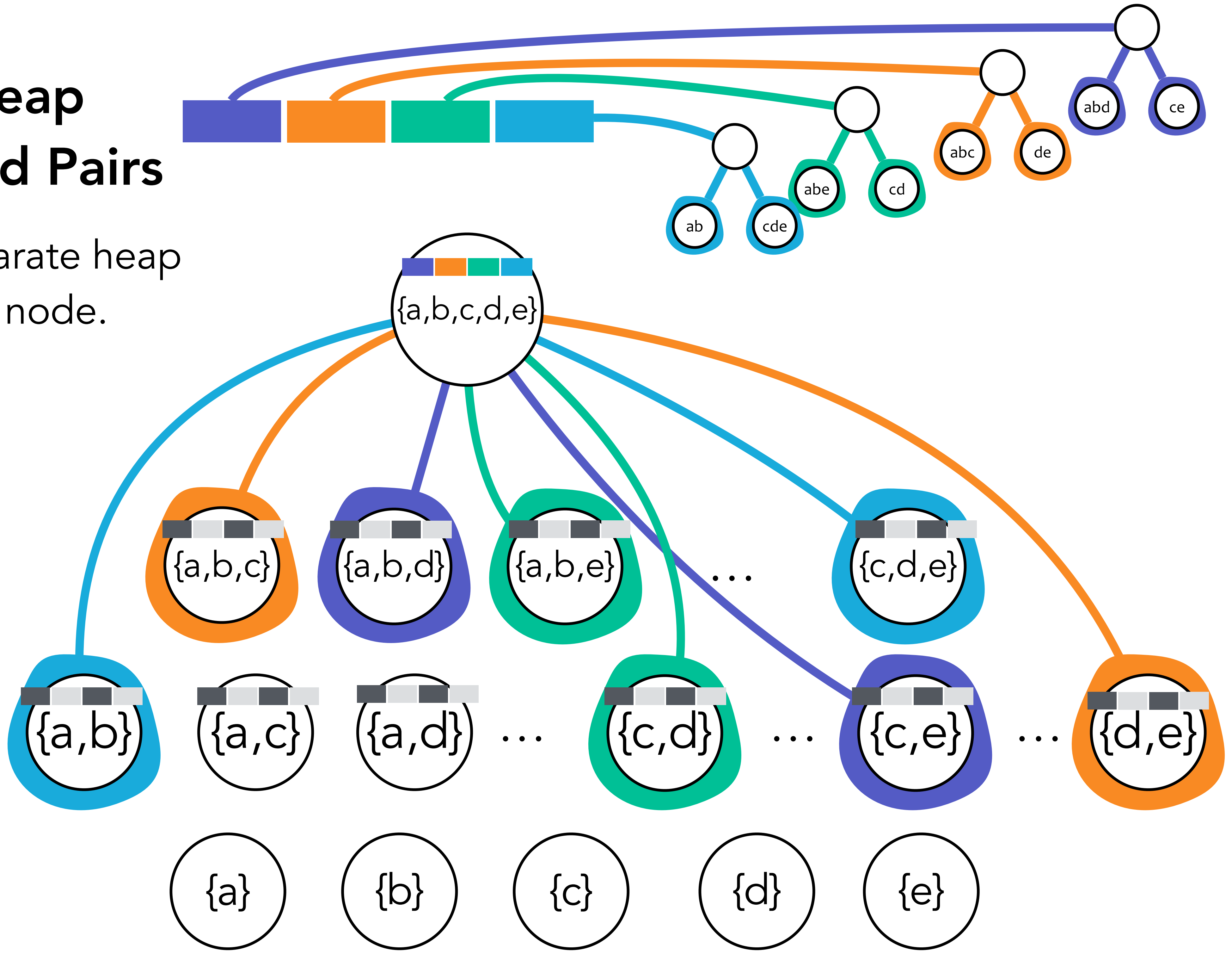


Craig Greenberg   Nicholas Monath

- We proposed A* search on the trellis to find the MLE hierarchy.
- **Best-first search** algorithm that reframes **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.

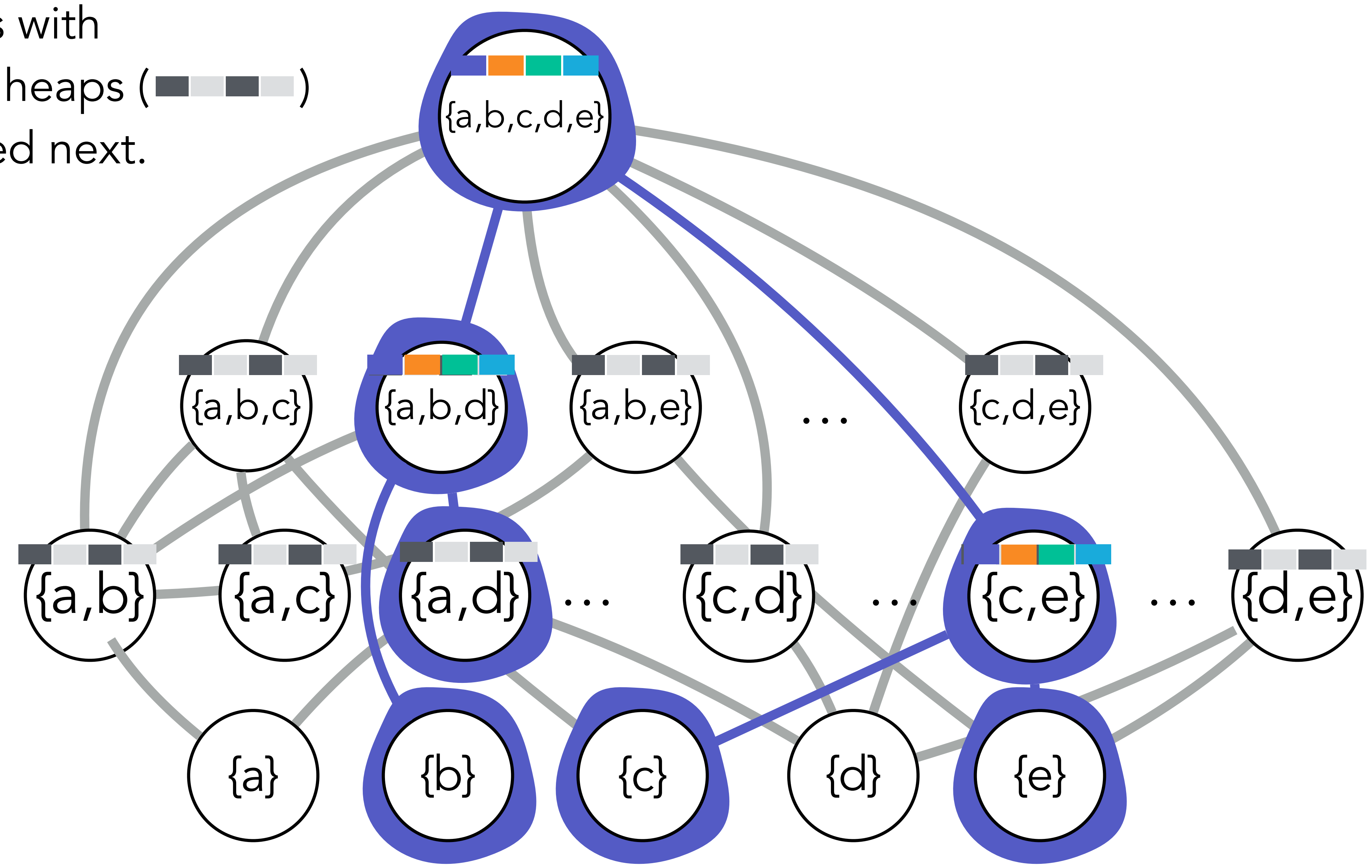
Min-Heap Over Child Pairs

Store a separate heap at each node.



Partial Hierarchical Clustering State

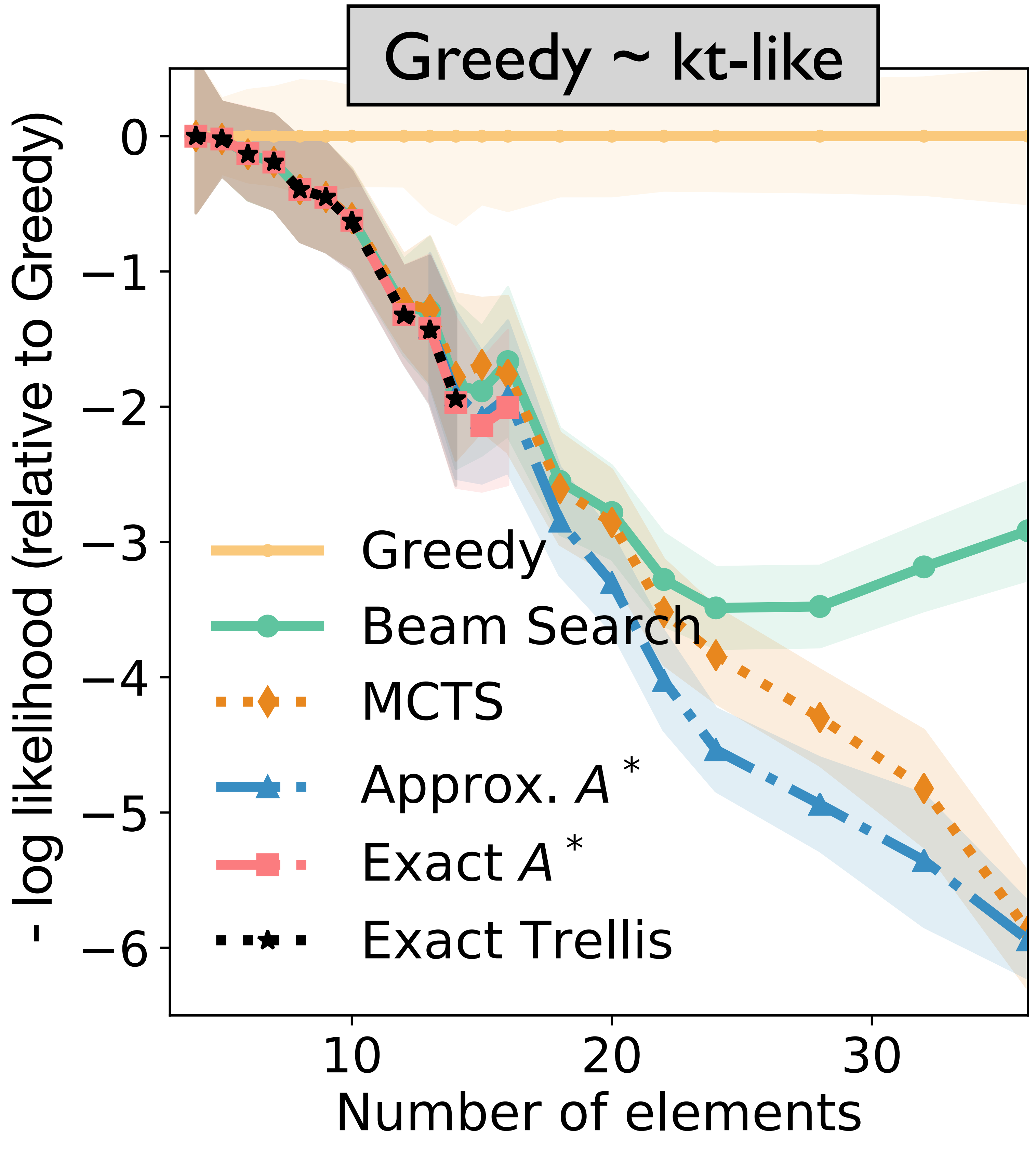
Leaf nodes with un-initialized heaps ( ) to be explored next.

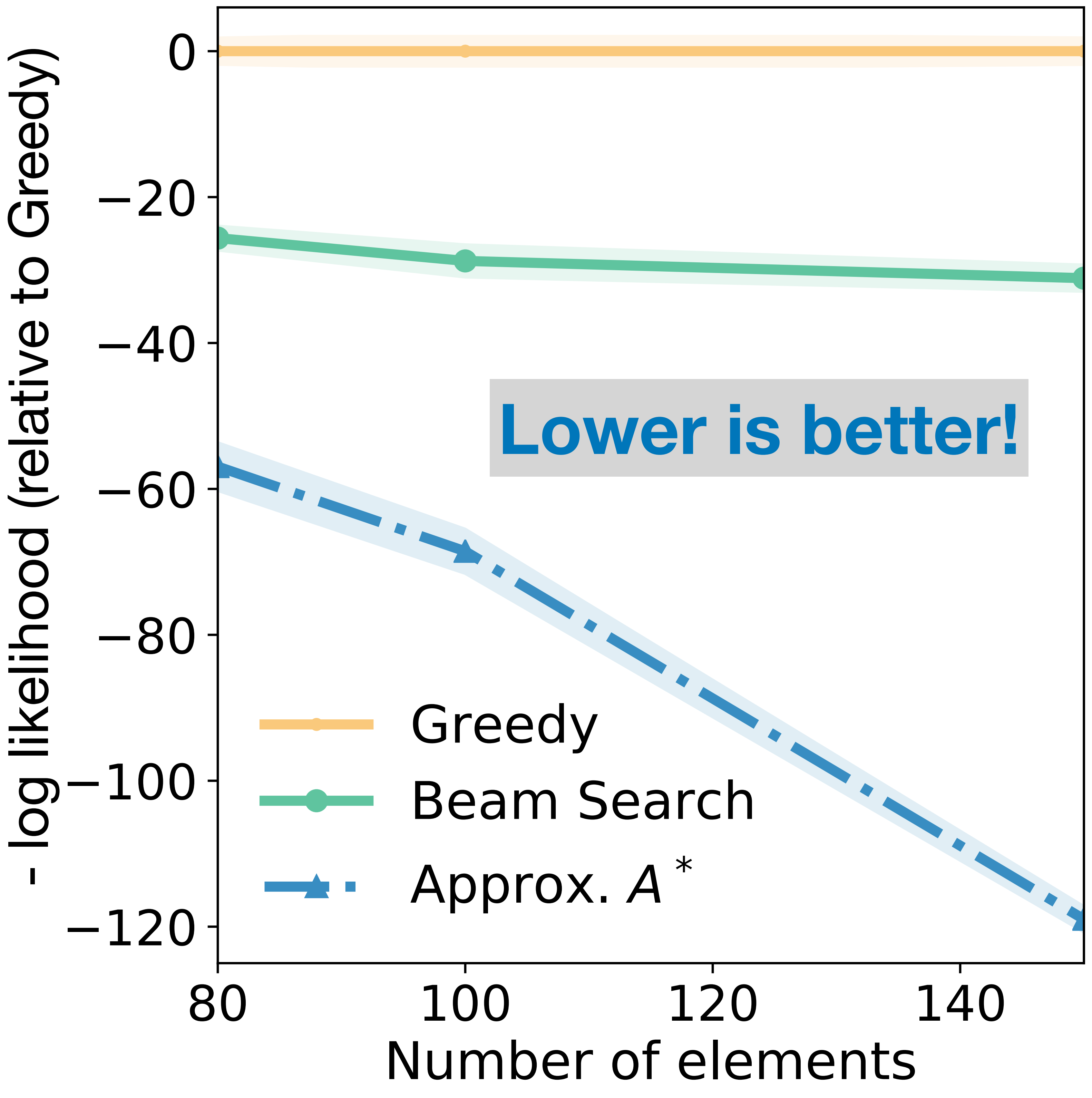


Comparison for maximum likelihood showering history

https://github.com/SebastianMacaluso/HCmanager

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





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

Probabilistic programming

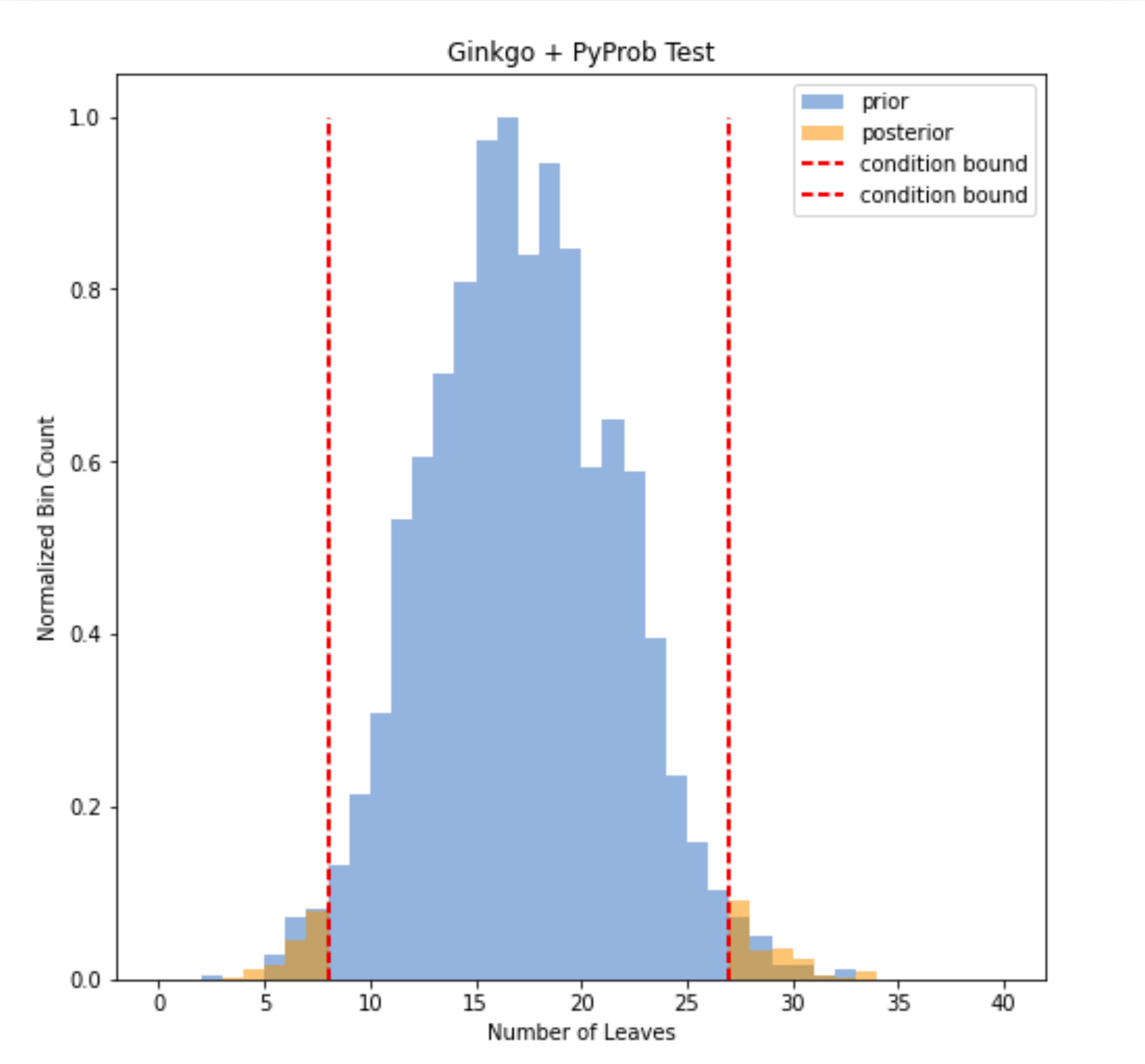
[Drnevich, Cranmer, Baydin, Macaluso, vCHEP 2021]

NYU

- Generating sufficient background in signal-like tails is hard, e.g. background for boosted jet taggers.
- Monte Carlo parton showers sample jets through random number generators.
- Probabilistic programming offers a way to hijack the random number generators and sample from this complicated region of phase space, e.g. importance sampling.
- Applied to Sherpa

NeurIPS 2019 [Baydin, Heinrich, Louppe, Cranmer, et al, arXiv:1807.07706]
SC 2019 [Baydin, Louppe, Cranmer et al, arXiv:1907.03382]

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

NYU

- Ultimately it would be very powerful if we could unify generation (the simulation / forward model) and inference (MC tuning, jet tagging, etc.). To do this, we need first to reframe jet physics in probabilistic terms.
- Introduced a simplified generative model to facilitate research into new computational techniques for jet physics.
- New implementations of likelihood-based clustering algorithms: greedy and beam search that provide a principled alternative to the generalized k_t clustering algorithms.
- Hierarchical cluster trellis and A* algorithms to exactly obtain the maximum likelihood showering history (approximate versions greatly improve over baselines). Also, applications in cancer genomics and possibly phylogenetic trees.
- New implementations of reinforcement learning based algorithms such as MCTS for jet clustering.
- Cluster trellis allows to marginalize and sample from the true posterior distribution of showering histories which could ameliorate bottlenecks in physics simulations with large jet multiplicity.
- Probabilistic programming allows to efficiently sample the tails of backgrounds in signal-rich regions of phase space.

Thanks for your attention!

Backup

Backup