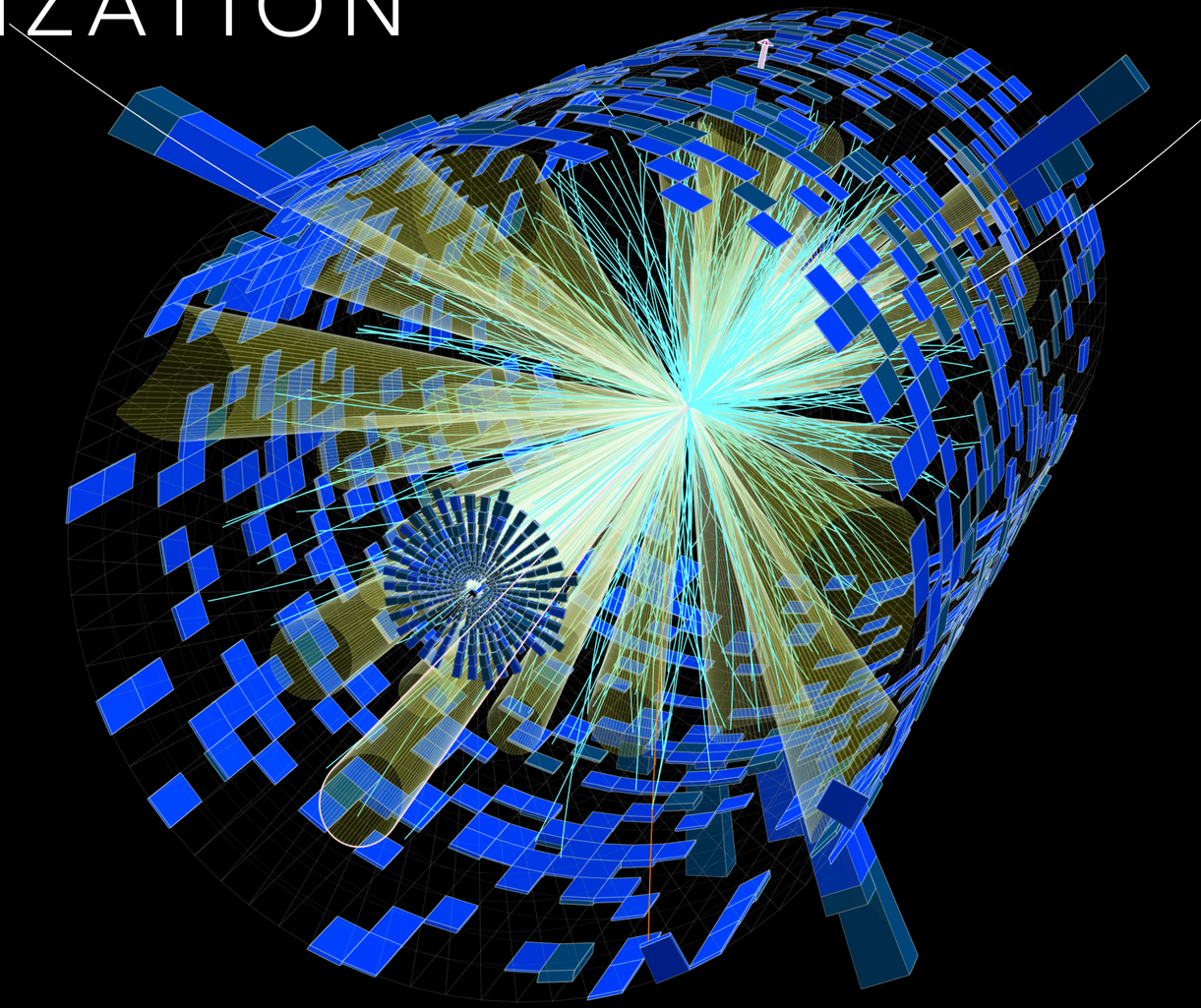
NYU Center for Data Science

Center for Cosmology and particle physics

QUARKS, HIERARCHICAL CLUSTERING, AND COMBINATORIAL OPTIMIZATION



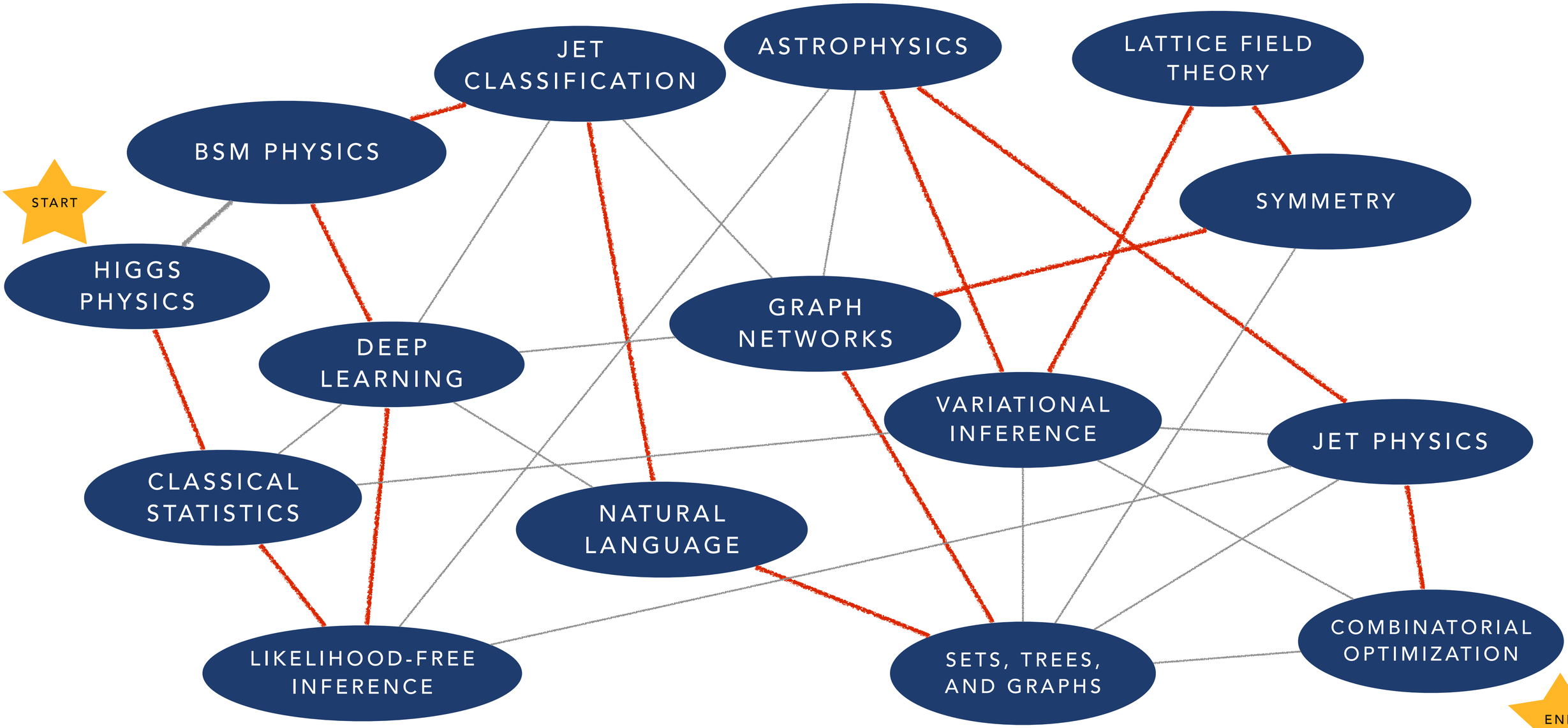
@KyleCranmer
New York University
Department of Physics
Center for Data Science
CILVR Lab

Workshop Overview

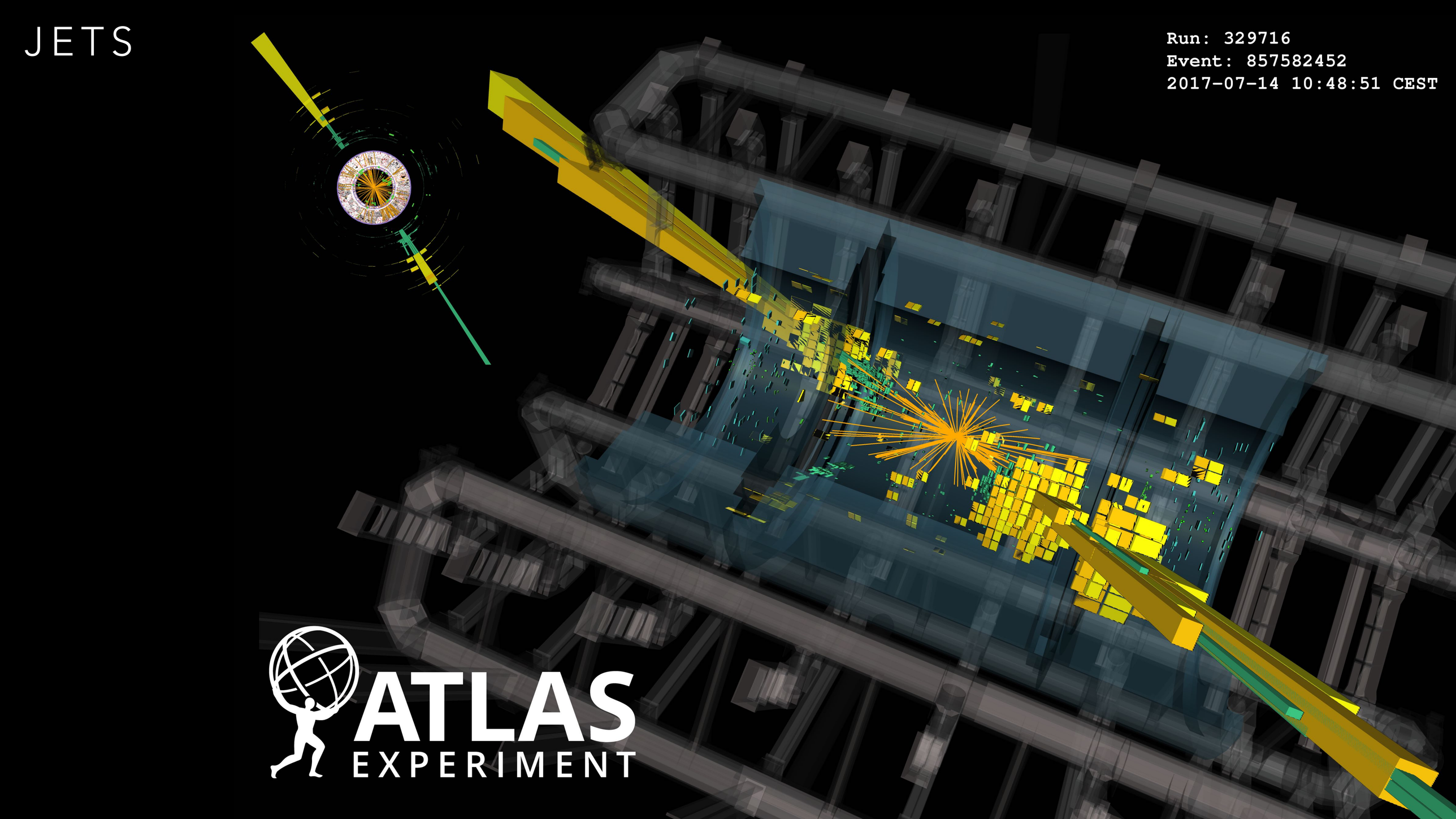
In recent years, **deep learning** has significantly improved the fields of computer vision, natural language processing and speech recognition. Beyond these traditional fields, deep learning has been expanded to quantum chemistry, **physics**, neuroscience, and more recently to **combinatorial optimization** (CO).

- DL is particularly attractive to address CO problems given its high flexibility, **approximate nature**, and self-learning paradigm.
- In other words, DL has the potential to learn universal high-quality algorithms and therefore could lead to a breakthrough in traditional CO, where algorithms are hand-crafted.
- On the other hand, **synergies between DL and CO algorithms** could lead to the possibility of taking **the best of the two domains** and deriving new algorithms, especially for applied problems.

Traveling ~~salesman~~ scientist



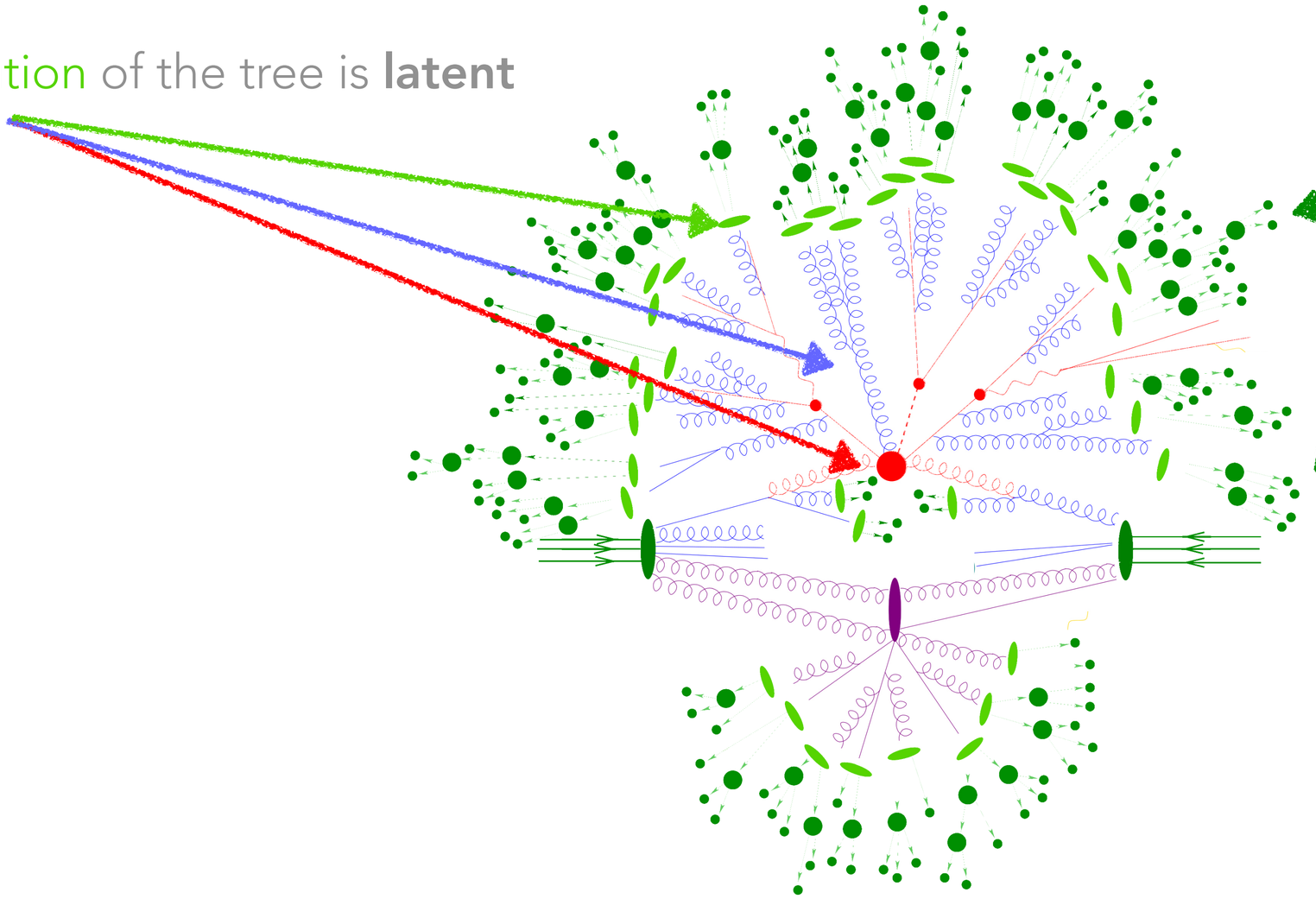
Quarks, Gluons, and Jets

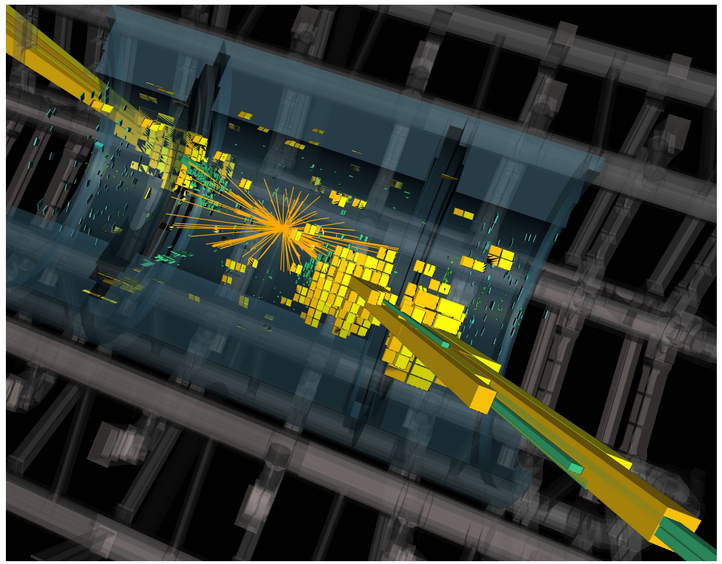

JETS
Run: 329716
Event: 857582452
2017-07-14 10:48:51 CEST
ATLAS
EXPERIMENT

Physicists' Model For Jets

Evolution of the tree is **latent**

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

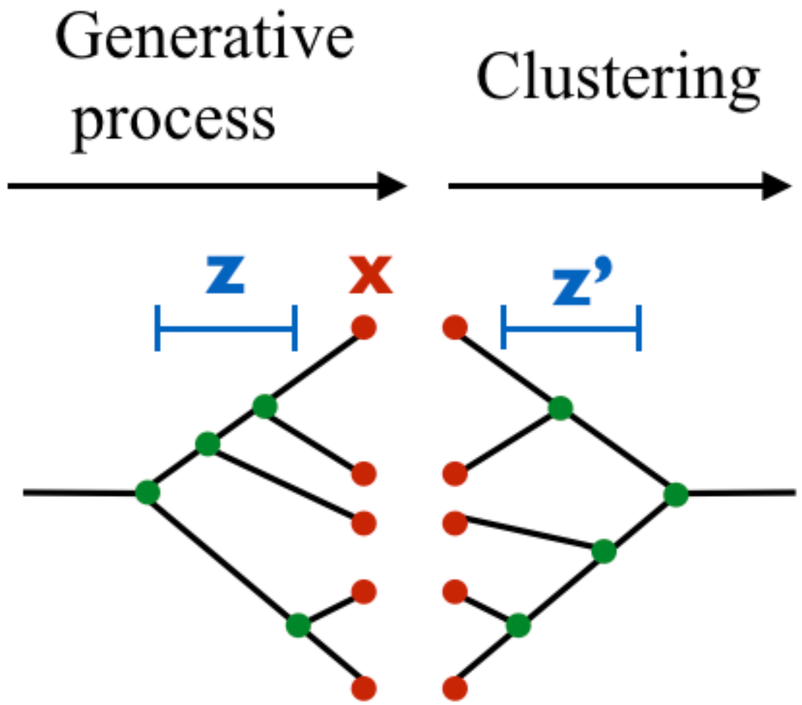




Jet Clustering

Physicists would like to reconstruct the unobserved latent tree z from observed x

- Hierarchical clustering can be seen as inverting the generative process
- Standard techniques use bottom-up / greedy / agglomerative clustering HAC
  - Similarity measure is motivated by underlying physics: QCD, relativity, etc.



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 7,676 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 3,945 citations

Dispelling the N^3 myth for the k_t jet-finder #1

Matteo Cacciari (Paris, LPTHE), Gavin P. Salam (Paris, LPTHE) (Dec, 2005)

Published in: *Phys.Lett.B* 641 (2006) 57-61 • e-Print: hep-ph/0512210 [hep-ph]

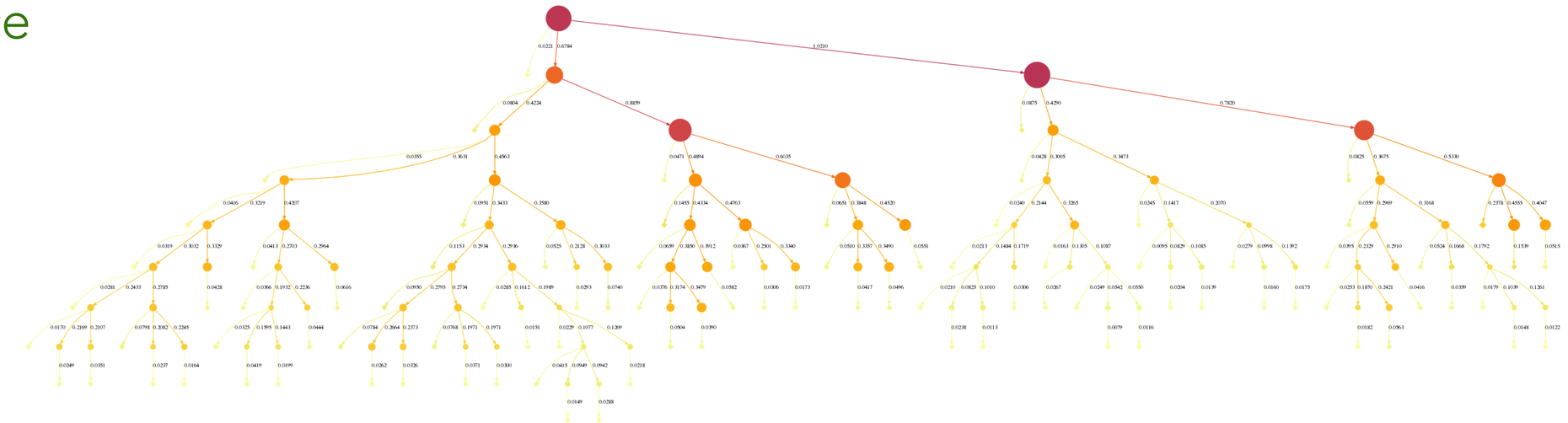
pdf DOI cite 1,988 citations

Impact of clustering on downstream tasks

Optimal solutions for many down-stream tasks would be ~trivial if an oracle could give us the correct / ground-truth tree, but

- Several trees could potentially lead to the same set of leaves, so we need to think probabilistically

Traditional approaches for downstream tasks (eg. classification, regression, etc.) interpret the HAC tree as an estimate for ground truth, and additional work is needed to characterize how properties of the clustering algorithm impact downstream performance



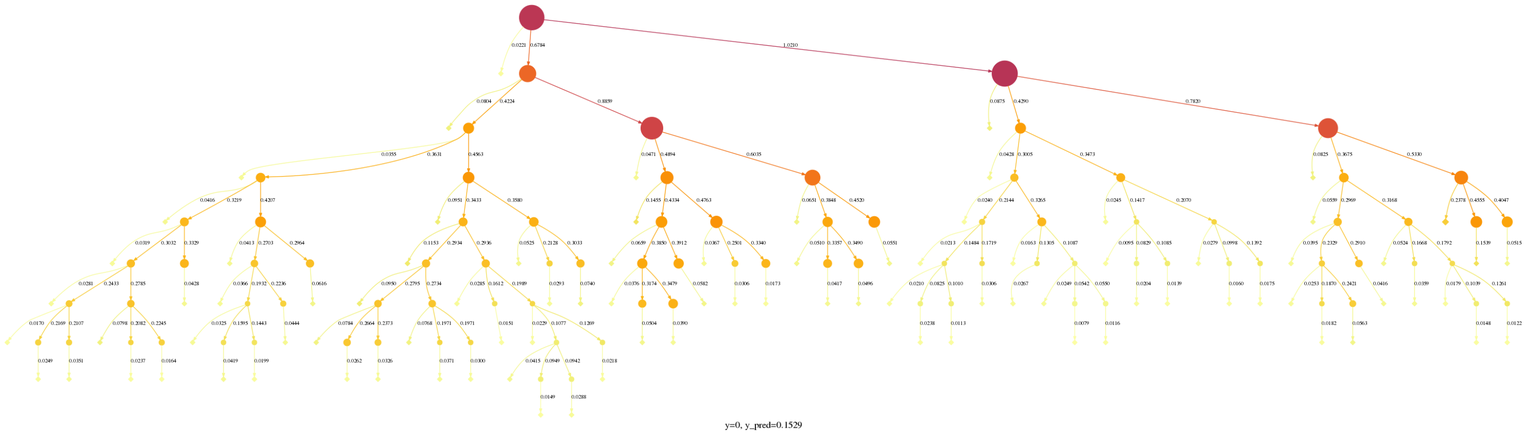
Deep Learning

Deep learning has been used successfully for solving downstream tasks without explicitly inferring the latent state

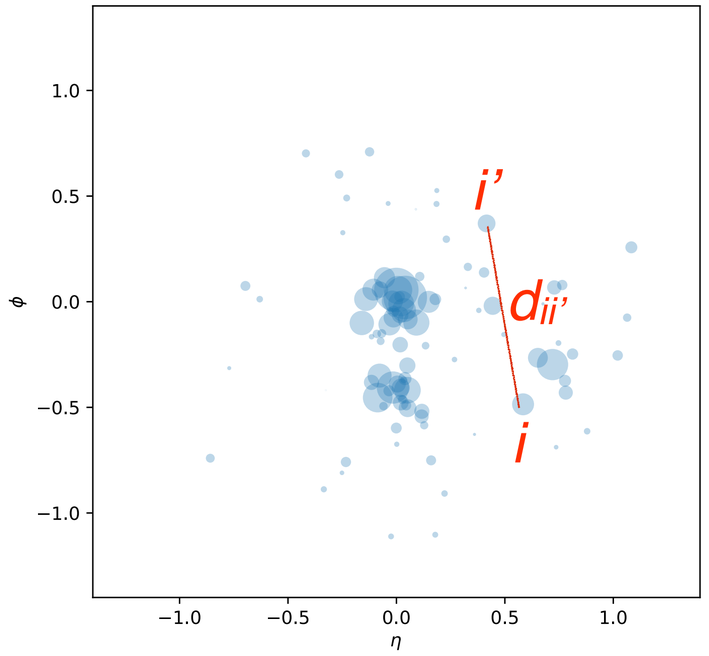
- Rich research area for networks working on structured data like sets, trees, and graphs
- Some hybrid deep learning algorithms exploit physics knowledge directly (eg. Include HAC clustering internally)

I've been interested in hybrid physics-DL models that treat latent state probabilistically

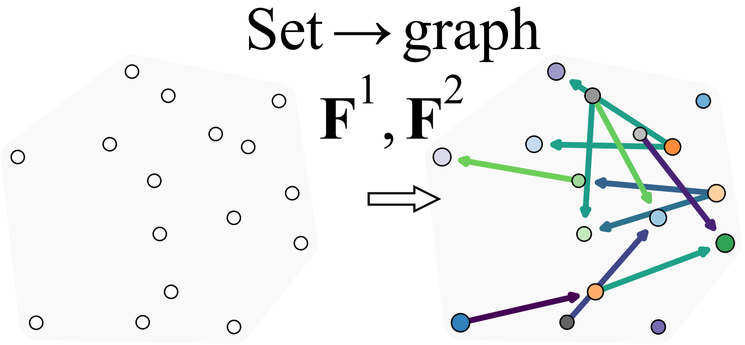
Louppe, Cho, Becot, KC [arXiv:1702.00748]



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



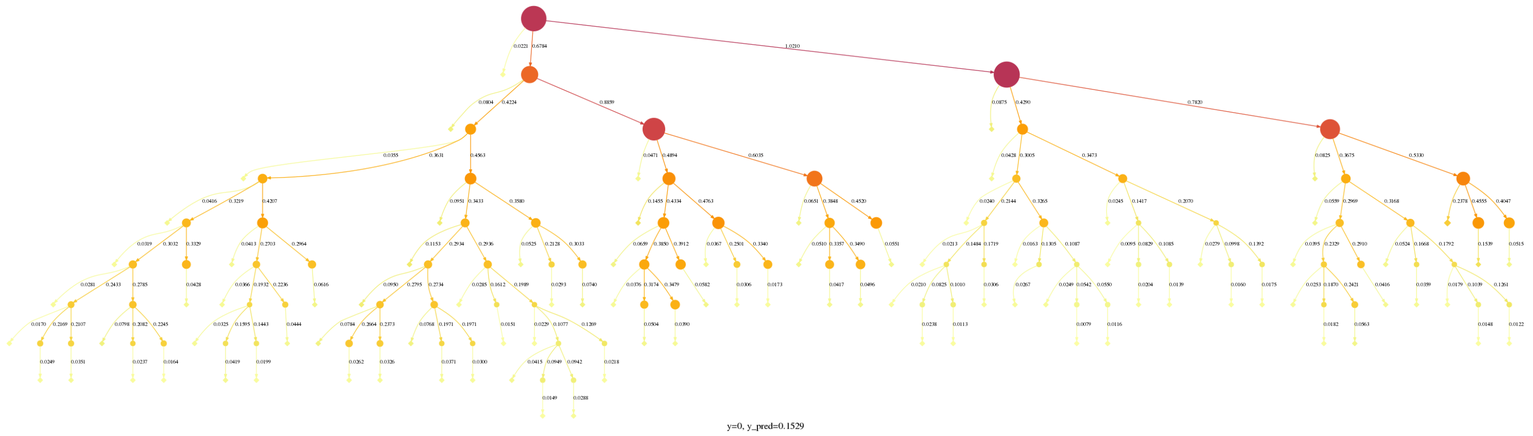
Henrion, Louppe, Cho, KC, Bruna [NeurIPS DL4Physics 2017]



Serviansky, KC, et. al. [arXiv:2002.08772]

J. Shlomi, P. Battaglia, and J.-R. Vlimant, "Graph neural networks in particle physics", (2020). In Preparation.

Generative Model



In reality the best we can hope for is to invert the process probabilistically

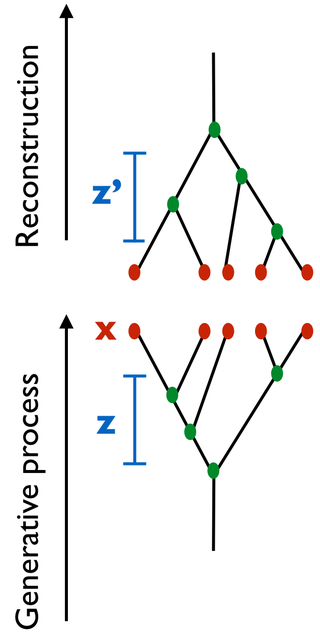
- Forward model is a Markov process with a relatively simple probability model for each splitting parametrized by θ

$$p(\text{tree}\,|\,\theta) = \prod_{\text{nodes}} \text{p(left, right}\,|\,\text{parent}, \theta)$$

Inference on θ would be new

Reframe jet physics in statistical terms

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



Generative Model

GInkgo by Sebastian Macaluso and Duccio Pappadopulo, and KC
https://github.com/SebastianMacaluso/ginkgo

Ginkgo is a simplified jet simulator written in pyro to facilitate research on exact / approximate optimization and marginalization over hierarchical clusterings (the latent tree) using

Sebastian Macaluso

- Probabilistic Programming
- Differentiable Programming
- Dynamic Programming
- Variational Inference

Captures essential physics:

- Energy / Momentum conservation
- Dokshitzer–Gribov–Lipatov–Altarelli–Parisi—like evolution

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

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

Going beyond agglomerative clustering

With this probabilistic view, interesting to revisit standard agglomerative clustering

- Similarity heuristic: $d_{ii'}^{\alpha} = \min(p_{ti}^{2\alpha}, p_{ti'}^{2\alpha}) \frac{\Delta R_{ii'}^2}{R^2}$
- Generative model for jets: $p(\text{tree}\,|\,\theta) = \prod_{\text{nodes}} \text{p}(\text{left}, \text{right}\,|\,\text{parent}, \theta)$
- If $p_{\text{node}}(\text{left}, \text{right}\,|\,\text{parent}, \theta)$ is monotonic with heuristic, then agglomerative clustering is a greedy algorithm to find an approximate MAP estimate

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

Immediately suggests we can do better with beam search or some other search algorithm

Combinatorial Optimization

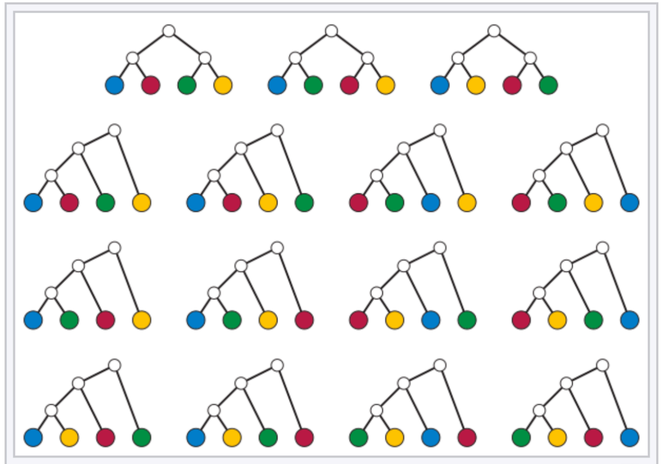
The immediate problem is that the search space (the number of binary trees on unordered leaves) is enormous!! $(2N-3)!!$

- Fortunately, when Sebastian Macaluso was at UMass we connected with Nicholas Monath, Craig Greenberg, and Andrew McCallum's group

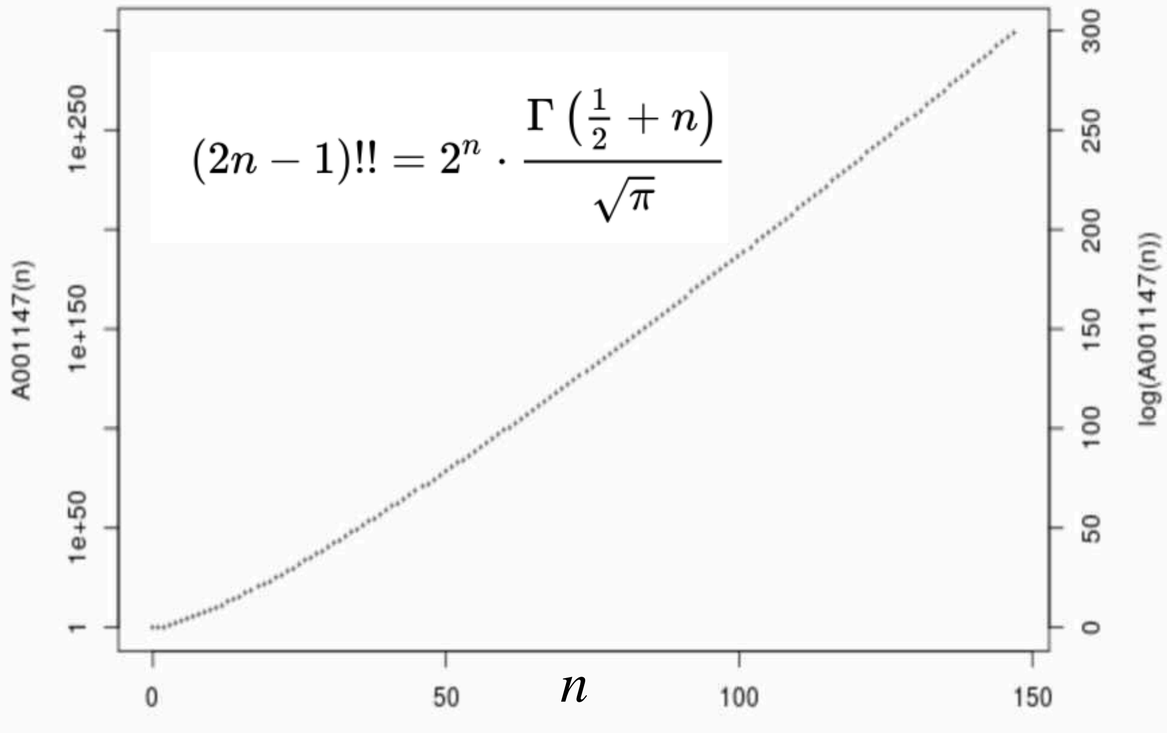
Number of clustering histories for N leaves grows as

$$a(N) = (2N-3)!!$$

# 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).



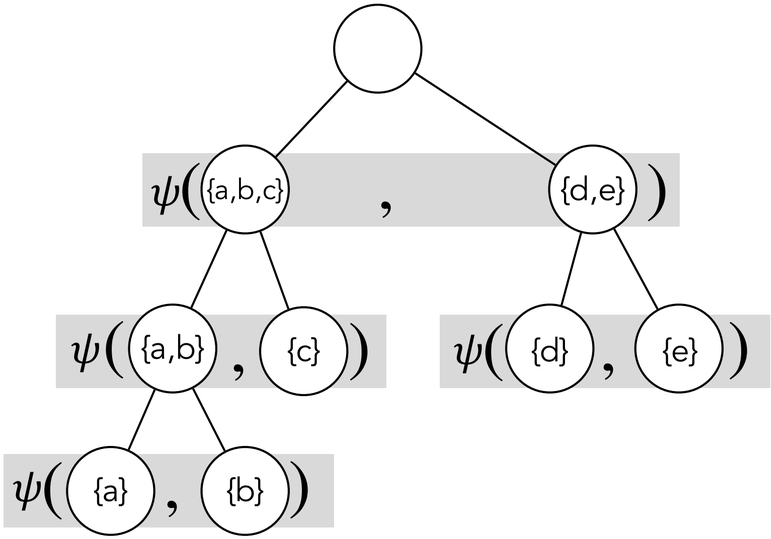
The Trellis Data Structure

The trellis

We consider cost / energy / likelihood functions that are products over siblings

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

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



This includes:

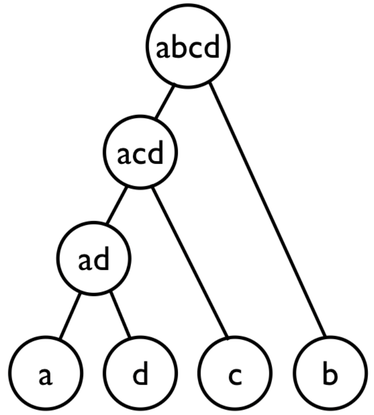
- the likelihood for tree-like Markov models like our physics use case,
- a cost function used in cancer genomics,
- the Dasgupta cost

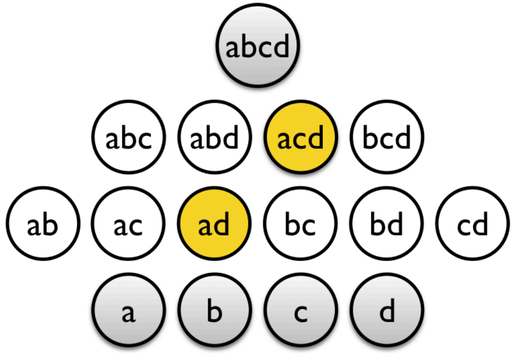
AISTATS 2021 [arXiv:2002.11661]

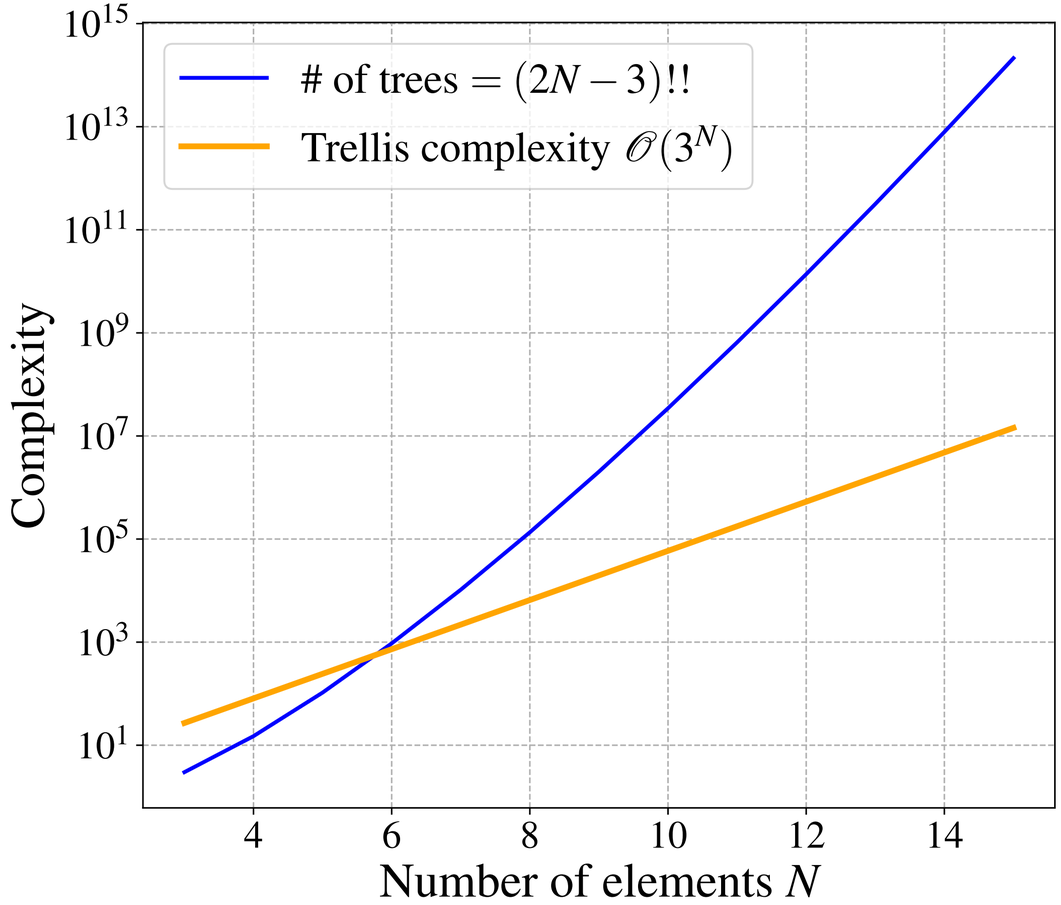
The trellis

A trellis is constructed, where nodes correspond to all subsets of elements

- The full trellis is large ($2^{|X|}$), but the number of trees is super-exponentially larger than the number of nodes in the trellis







Data Structures & Algorithms for Exact Inference in Hierarchical Clustering

Craig S. Greenberg*,[1,2], Sebastian Macaluso*,[3], Nicholas Monath[1], Ji-Ah Lee[4], Patrick Flaherty[4], Kyle Cranmer[3], Andrew McGregor[1], and Andrew McCallum[1]

[1]College of Information and Computer Sciences, University of Massachusetts Amherst, USA
[2]National Institute of Standards and Technology, USA
[3]Center for Cosmology and Particle Physics & Center for Data Science, New York University, USA
[4]Department of Mathematics and Statistics, University of Massachusetts Amherst, USA


Sebastian Macaluso


Craig Greenberg


Nicholas Monath

AISTATS 2021 [arXiv:2002.11661]

Dynamic programming

The trellis is an efficient data structure for storing memoized values of the the partition function $Z(V)$ and $\phi(V \mid H^*)$ for the MAP clustering H^* for each of subset of elements V

- This admits a **dynamic programming** algorithm to efficiently compute the **exact** partition function and MAP over all possible clusterings

Exhaustive Computation of the Partition Function - *O((2N-3)!!)*

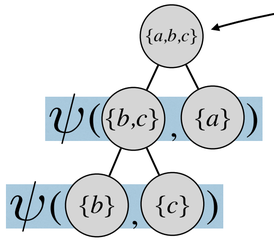
$$
\begin{aligned}
Z(\{a,b,c,d\}) = {} & \psi(\{a,b,c\},\{d\}) \cdot \psi(\{a,b\},\{c\}) \cdot \psi(\{a\},\{b\}) \\
& + \psi(\{a,b,c\},\{d\}) \cdot \psi(\{a,c\},\{b\}) \cdot \psi(\{a\},\{c\}) \\
& + \psi(\{a,b,c\},\{d\}) \cdot \psi(\{b,c\},\{a\}) \cdot \psi(\{b\},\{c\}) \\
& + \psi(\{a,c,d\},\{b\}) \cdot \psi(\{a,c\},\{d\}) \cdot \psi(\{a\},\{c\}) \\
& + \psi(\{a,c,d\},\{b\}) \cdot \psi(\{a,d\},\{c\}) \cdot \psi(\{a\},\{d\}) \\
& + \psi(\{a,c,d\},\{b\}) \cdot \psi(\{c,d\},\{a\}) \cdot \psi(\{c\},\{d\}) \\
& + \psi(\{a,b,d\},\{c\}) \cdot \psi(\{a,b\},\{d\}) \cdot \psi(\{a\},\{b\}) \\
& + \psi(\{a,b,d\},\{c\}) \cdot \psi(\{a,d\},\{b\}) \cdot \psi(\{a\},\{d\}) \\
& + \psi(\{a,b,d\},\{c\}) \cdot \psi(\{b,d\},\{a\}) \cdot \psi(\{b\},\{d\}) \\
& + \psi(\{b,c,d\},\{a\}) \cdot \psi(\{b,c\},\{d\}) \cdot \psi(\{b\},\{c\}) \\
& + \psi(\{b,c,d\},\{a\}) \cdot \psi(\{b,d\},\{c\}) \cdot \psi(\{b\},\{d\}) \\
& + \psi(\{b,c,d\},\{a\}) \cdot \psi(\{c,d\},\{d\}) \cdot \psi(\{c\},\{d\}) \\
& + \psi(\{a,b\},\{c,d\}) \cdot \psi(\{a\},\{b\}) \cdot \psi(\{c\},\{d\}) \\
& + \psi(\{a,c\},\{b,d\}) \cdot \psi(\{a\},\{c\}) \cdot \psi(\{b\},\{d\}) \\
& + \psi(\{a,d\},\{b,c\}) \cdot \psi(\{a\},\{d\}) \cdot \psi(\{b\},\{c\})
\end{aligned}
$$

Computation using Trellis - *O(3^N) << O((2N-3)!!)*

$$
\begin{aligned}
Z(\{a,b,c,d\}) = {} & \psi(\{a,b,c\},\{d\}) \cdot Z(\{a,b,c\}) \cdot Z(\{d\}) + \psi(\{a,b,d\},\{c\}) \cdot Z(\{a,b,d\}) \cdot Z(\{c\}) \\
& + \psi(\{a,c,d\},\{b\}) \cdot Z(\{a,c,d\}) \cdot Z(\{b\}) + \psi(\{b,c,d\},\{a\}) \cdot Z(\{b,c,d\}) \cdot Z(\{a\}) \\
& + \psi(\{a,b\},\{c,d\}) \cdot Z(\{a,b\}) \cdot Z(\{c,d\}) + \psi(\{a,c\},\{b,d\}) \cdot Z(\{a,c\}) \cdot Z(\{b,d\}) \\
& + \psi(\{a,d\},\{b,c\}) \cdot Z(\{a,d\}) \cdot Z(\{b,c\})
\end{aligned}
$$

Recursive Computation of Z using Memoization

Tree Potential is Product of Sibling Potentials

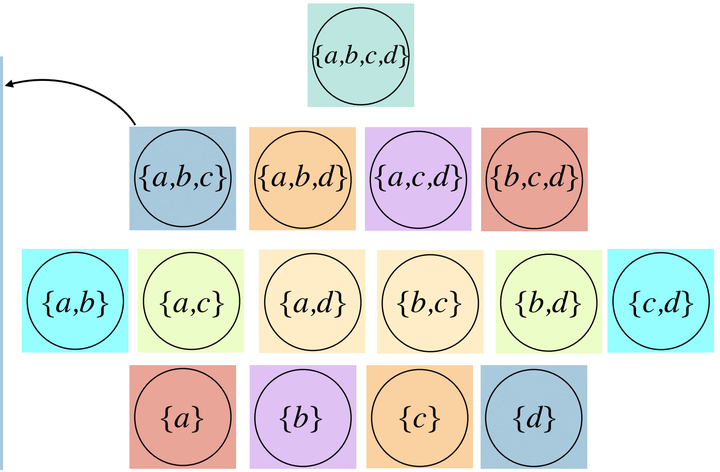


Partition Function is Sum of Tree Potentials

$\phi(\{a,b,c\}) + \phi(\{a,b,c\}) + \phi(\{a,b,c\})$

{a,b} {c} {a,c} {b} {b,c} {a}

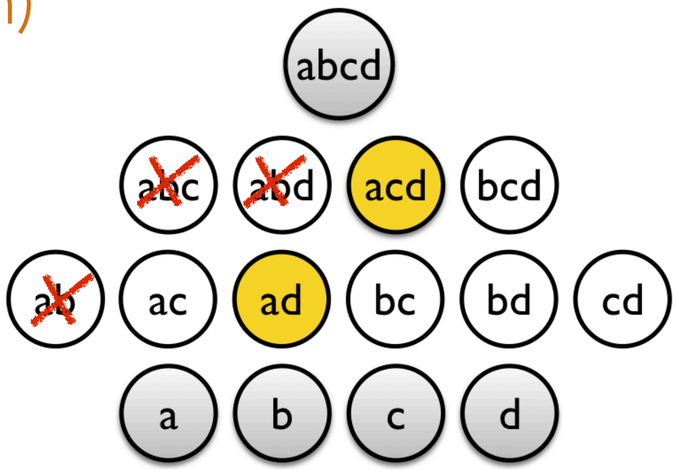
$$
\begin{aligned}
Z(\{a,b,c\}) = {} & \psi(\{a,b\},\{c\}) \cdot Z(\{a,b\}) \cdot Z(\{c\}) \\
& + \psi(\{a,c\},\{b\}) \cdot Z(\{a,c\}) \cdot Z(\{b\}) \\
& + \psi(\{b,c\},\{a\}) \cdot Z(\{b,c\}) \cdot Z(\{a\})
\end{aligned}
$$

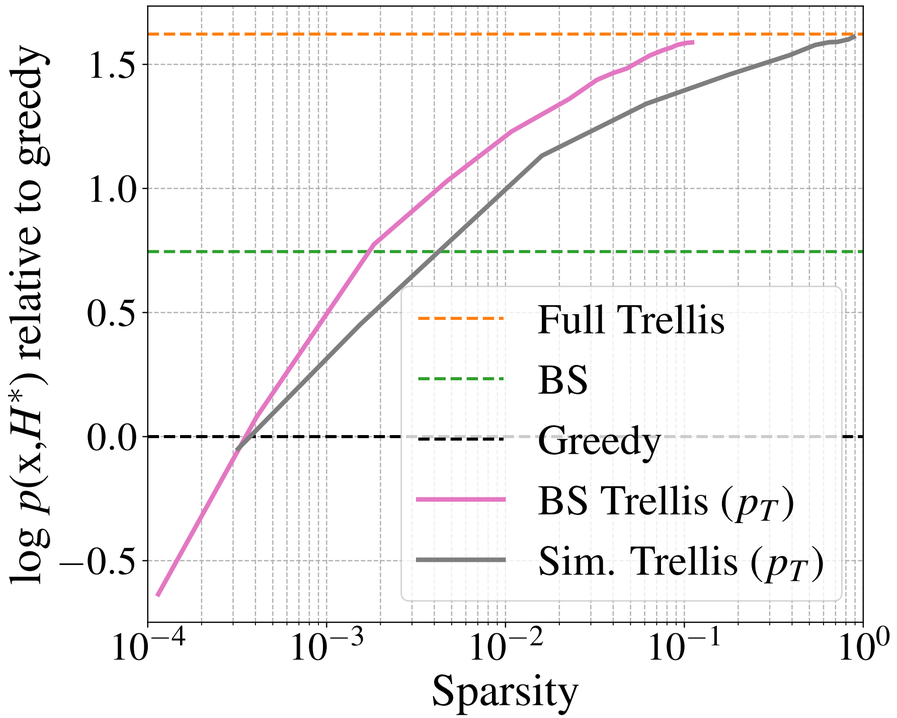


The sparse trellis

The problem with the full trellis is that it still grows exponentially

- But there is a natural **sparse trellis** that admits an **approximate** algorithm for the MAP clustering and partition function
- One simply removes nodes from the trellis (subject to a consistency condition)
- 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





Efficient Sampling

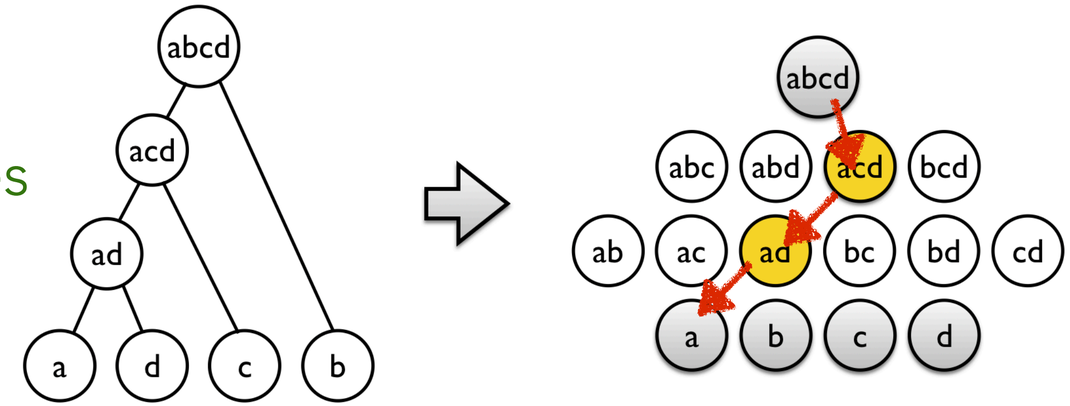
Combinatorial Marginalization

Variational Inference

Sampling

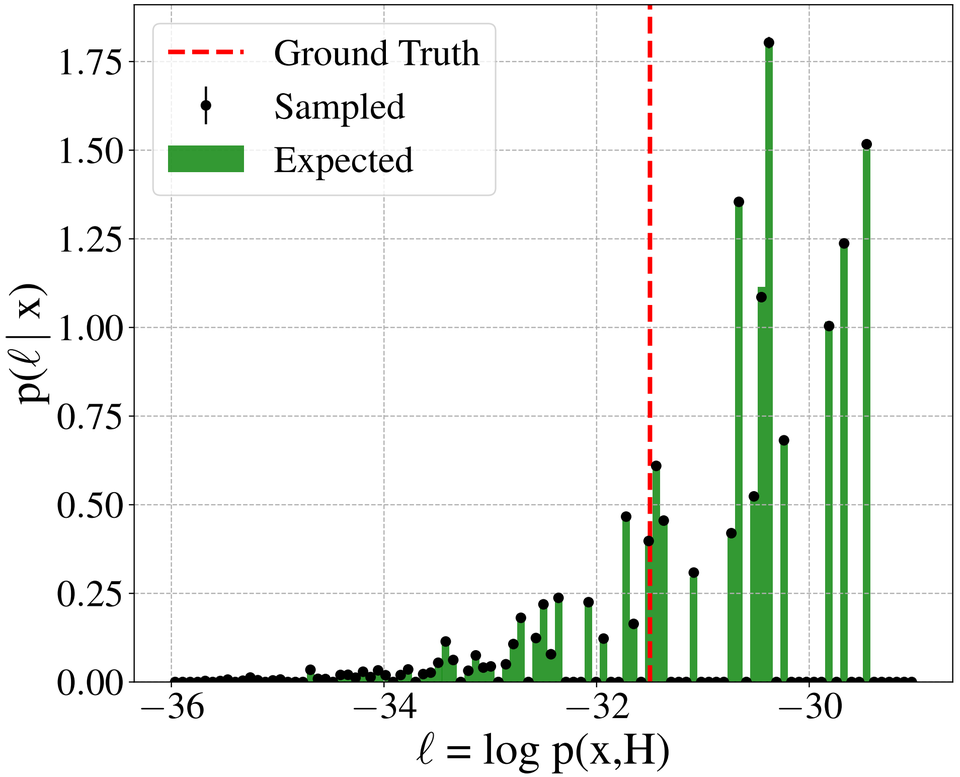
The partition function values $Z(V)$ for each node in the trellis encode a distribution over all possible trees

- One can populate a trellis with samples and think of the resulting trellis as a histogram over the discrete space of possible binary trees



Transversing the trellis from top to bottom mirrors the (Markov) generative model we have in physics

- A softmax over the partition functions for nodes at each level leads to categorical distribution for splitting at that level
- This leads to a natural, efficient, and exact sampling algorithm



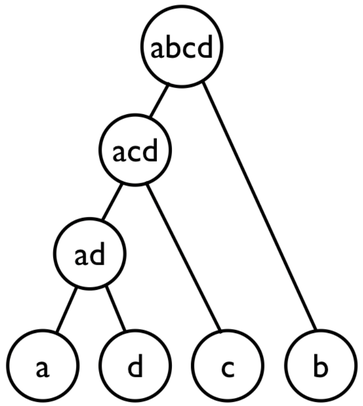
Maximum Likelihood & Variational Inference

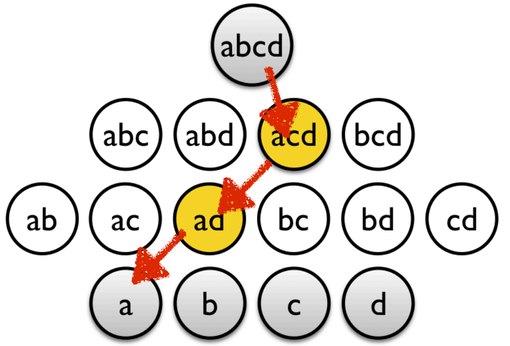
The partition function values $Z(V)$ for each node in the trellis encode a distribution over all possible trees

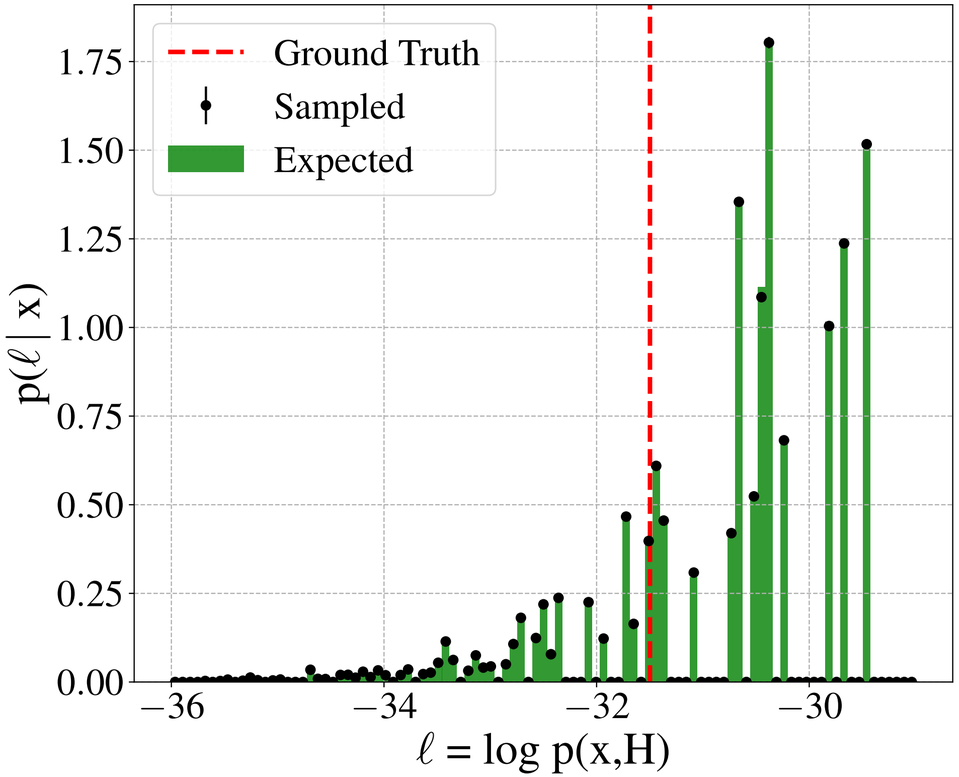
- This leads to a parametrization of distributions over the discrete space of possible binary trees
- We can also use a neural network or some other model to reparametrize $NN : \phi \rightarrow Z(V)$
- We can fit this parametrized distribution to samples of trees with maximum likelihood

We can also construct something like Variational Auto Encoder to approximate the posterior distribution $p(z \mid x)$ with a neural network parametrized trellis: $NN : x, \phi \rightarrow Z(V)$

- Together with efficient sampling could lead to efficient variational inference over the structured latent space







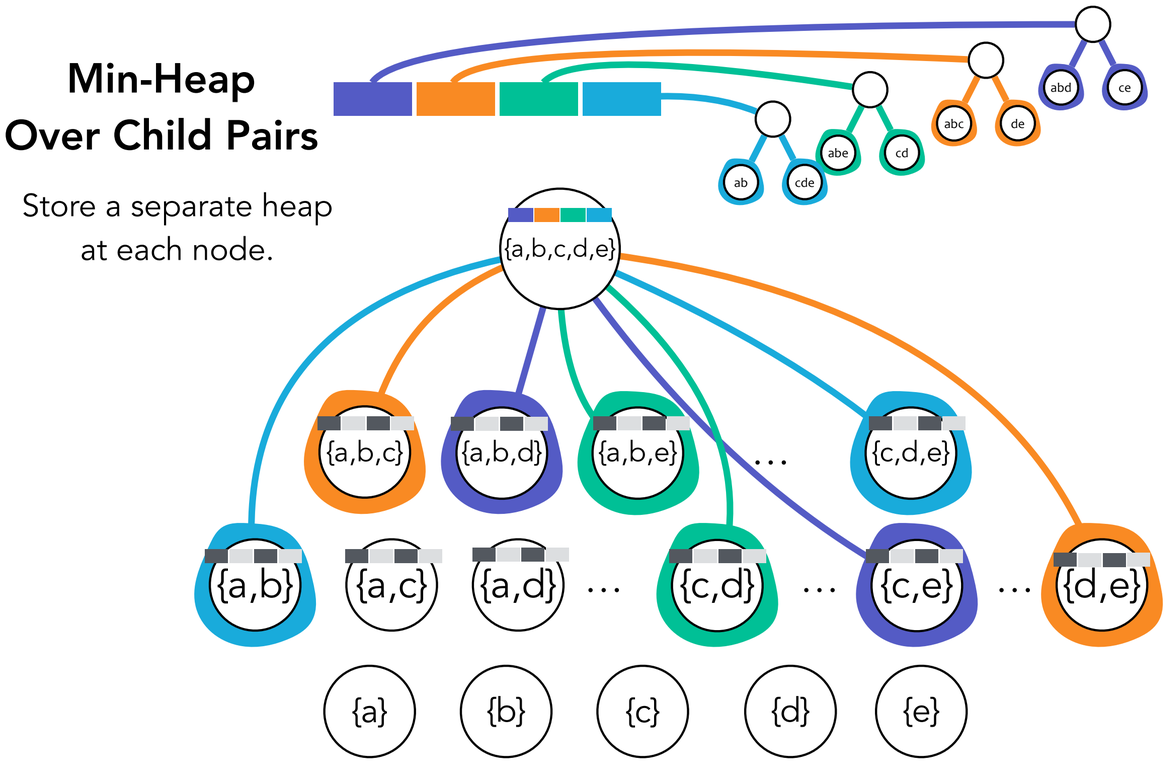
A* search + trellis

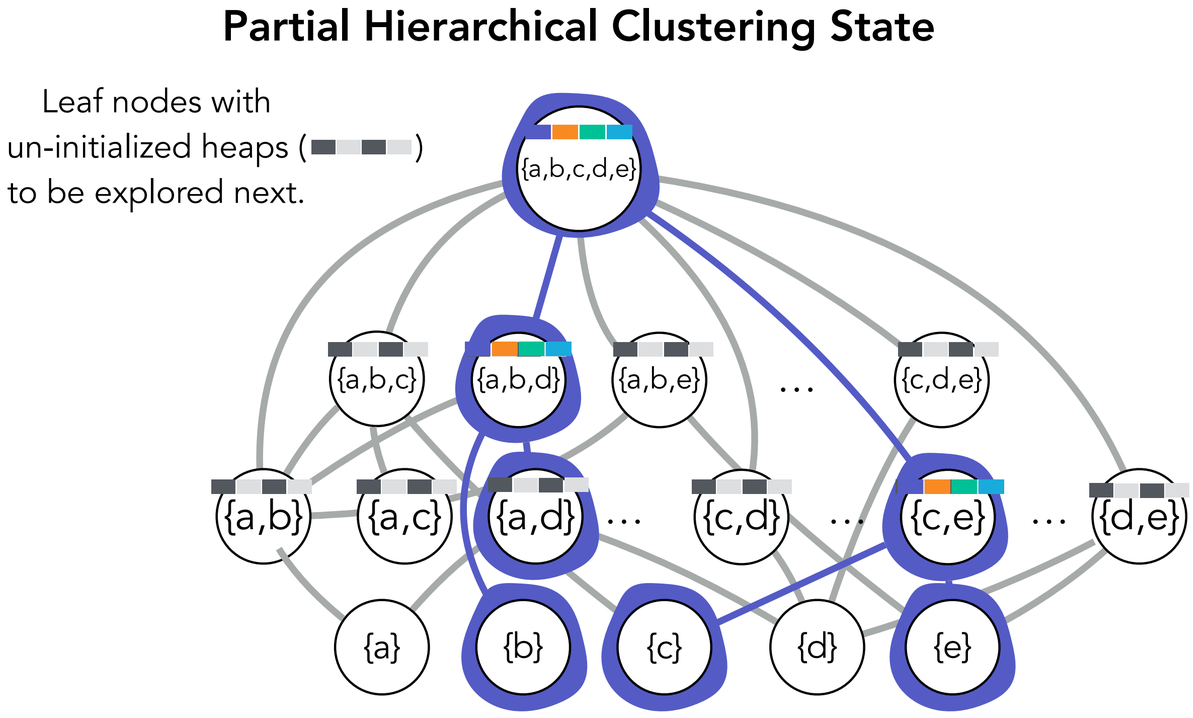
Preliminary work by:
Craig Greenberg*
Sebastian Macaluso*
Nicholas Monath*
Avinava Dubey
Patrick Flaherty
Manzil Zaheer
Amr Ahmed
Kyle Cranmer
Andrew McCallum

A* search + trellis

Most recently, we combined the trellis with A* search to find an efficient algorithm to find the global MAP hierarchical clustering that can leverage a heuristic

- Compactly encode states in the space of hierarchical clusterings (as paths from the root to the leaves of the trellis), and
- Compactly represent the search frontier (as nested priority queues).





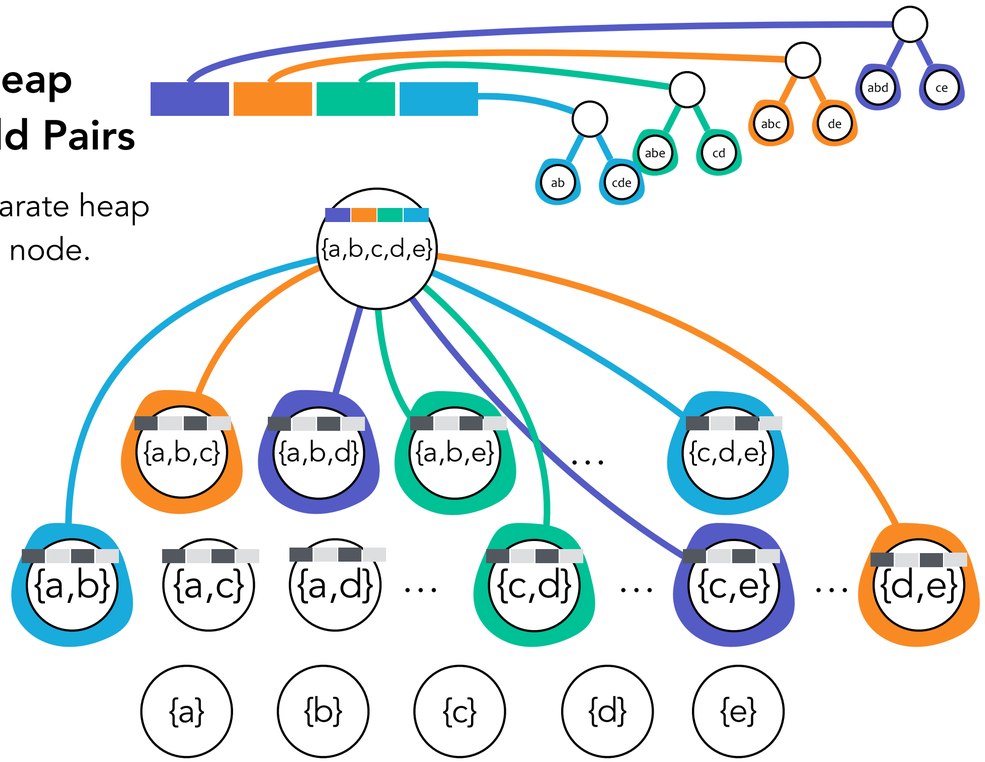
Approximate A* search + trellis

We can limit the size of the priorities queues and/or number of trees explored

- Yields an approximate A* search
- We can control the computational complexity of the search
- This allows us to scale the algorithms to many more elements

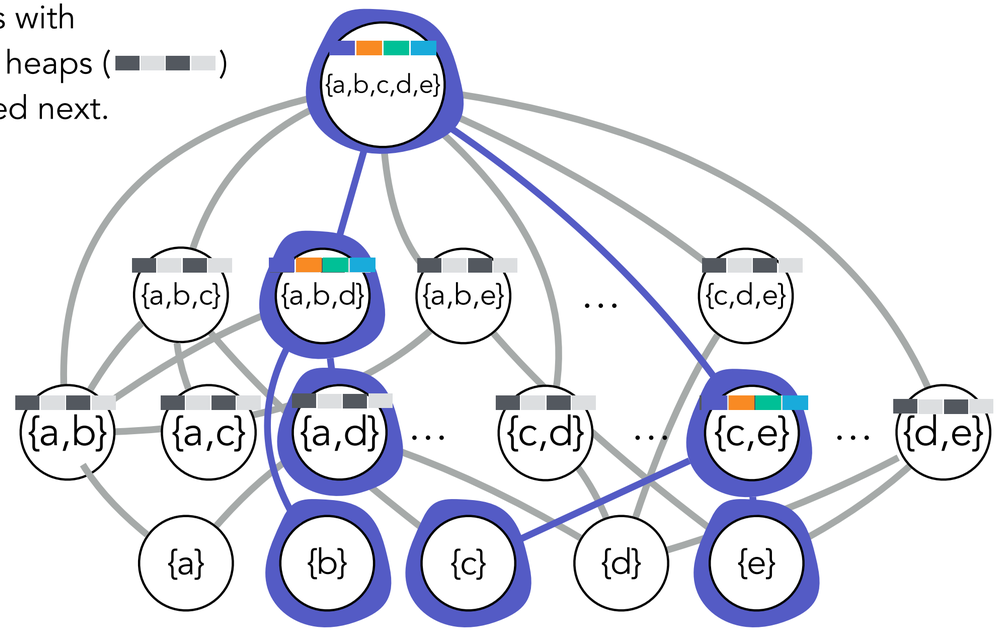
Min-Heap Over Child Pairs

Store a separate heap at each node.



Partial Hierarchical Clustering State

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

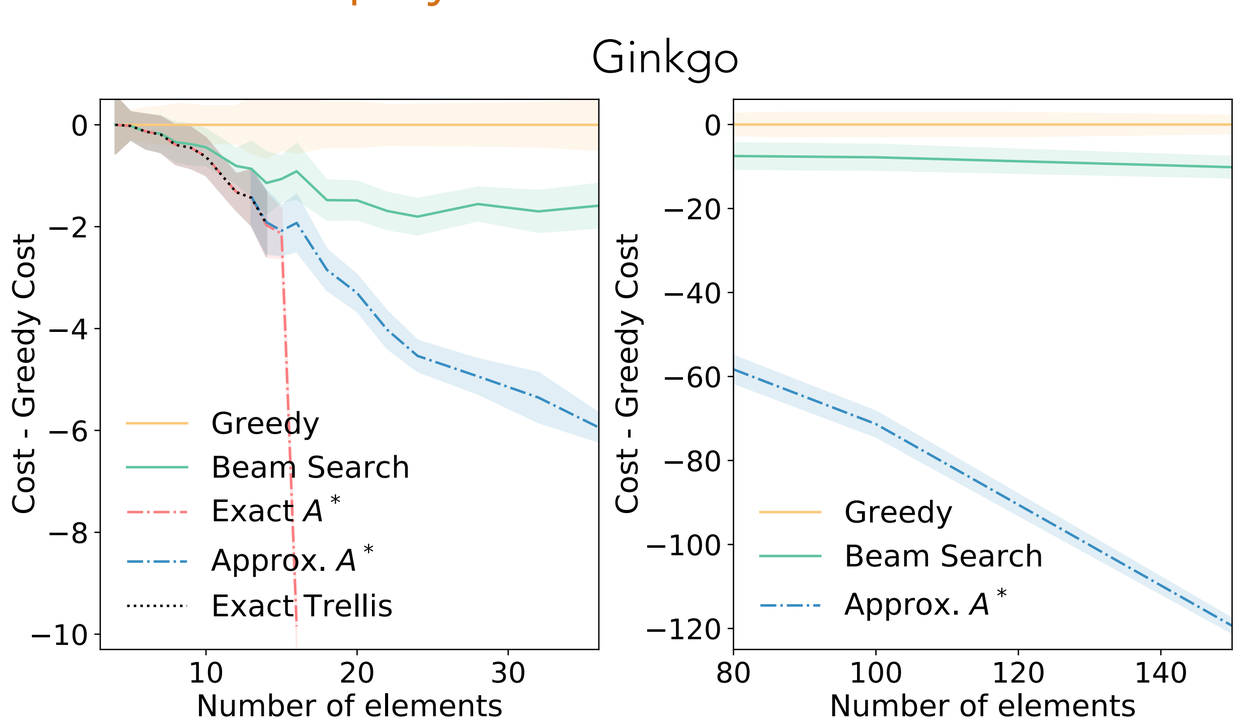


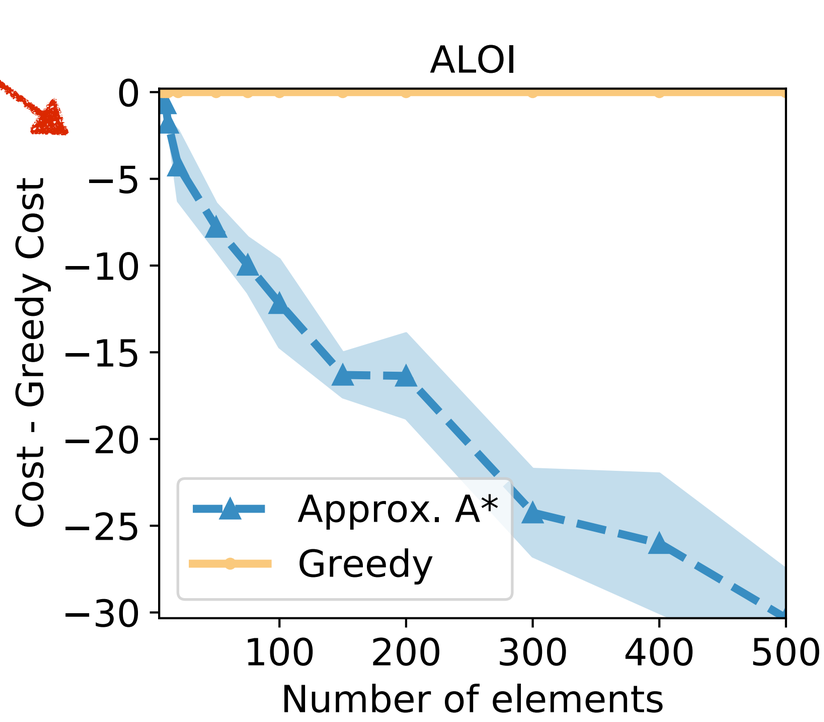
Results

The resulting algorithm allows us to extend the search for the exact MAP / lowest-cost hierarchical clustering to more elements (from 10^{12} to 10^{15} trees)

The approximate algorithm improves over baselines, even in enormous search spaces with 150 elements (10^{300} trees)

- Also non-physics benchmark with 500 elements (more than 10^{1000} trees)



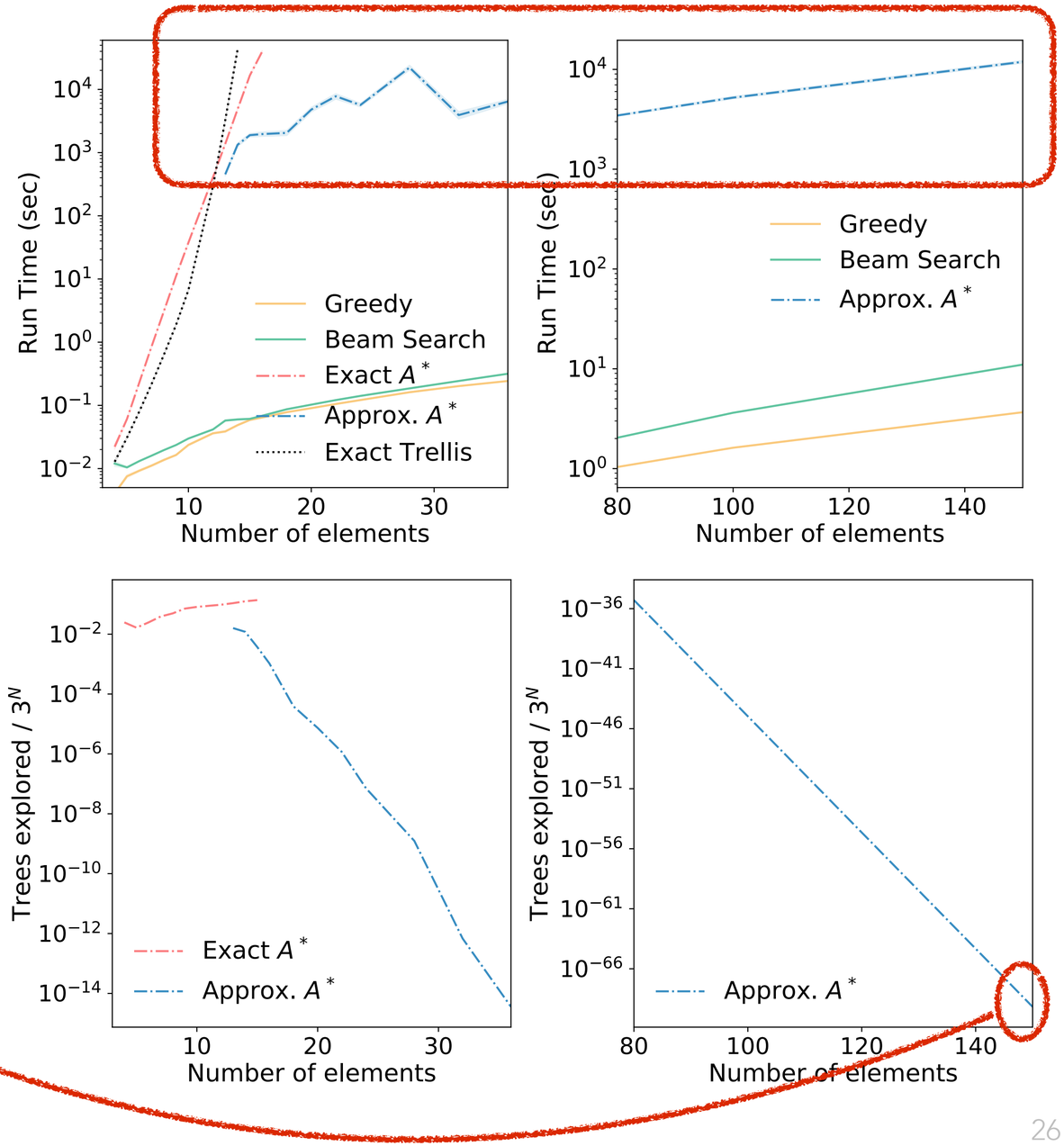


Scaling

The approximate A* algorithm has controlled run time even for ~150 elements

The number of trees explored by the approximate A* algorithm is tiny compared to the 3^N nodes in the trellis, which is super-exponentially smaller than the number of trees.

Only about 10^{-180} of the total number of trees are explored in this case



Hierarchical Clustering as a Markov Decision Process + Reinforcement Learning

Hierarchical Clustering as an MDP

Johann Brehmer, Sebastian Macaluso, Duccio Pappadopulo, KC [arXiv:2011.08191]
NeurIPS 2020 ML4Physical Sciences workshop

In a separate, but related thread of research, we framed the hierarchical clustering as a Markov Decision Process

- Interfaced to Open AI Gym
- Used various Reinforcement Learning algorithms

Markov Decision Process (MDP). We treat the problem of clustering as an MDP $(\mathcal{S}, \mathcal{A}, P, R)$:

- The state space $\mathcal{S}$ is given by all possible particle sets at any given point during the clustering process, $s = z_t$.
- The actions $\mathcal{A}$ are the choice of two particles $a = (i, j)$ with $1 \leq i < j \leq n_t$ to be merged.
- The state transitions P are deterministic and update z_t to z_{t-1} by replacing the particles $p_{t,i}$ and $p_{t,j}$ with a parent $p_{t-1,i} = p_{t,i} + p_{t,j}$. All other particles are left unchanged, each state transition thus reduces the number of particles by one.
- The rewards R are the splitting probabilities, $R(s = z_t, a = (i, j)) = \log p_s(z_t | z_{t-1}(i, j))$.
- The MDP is episodic and terminates when only a single particle is left.

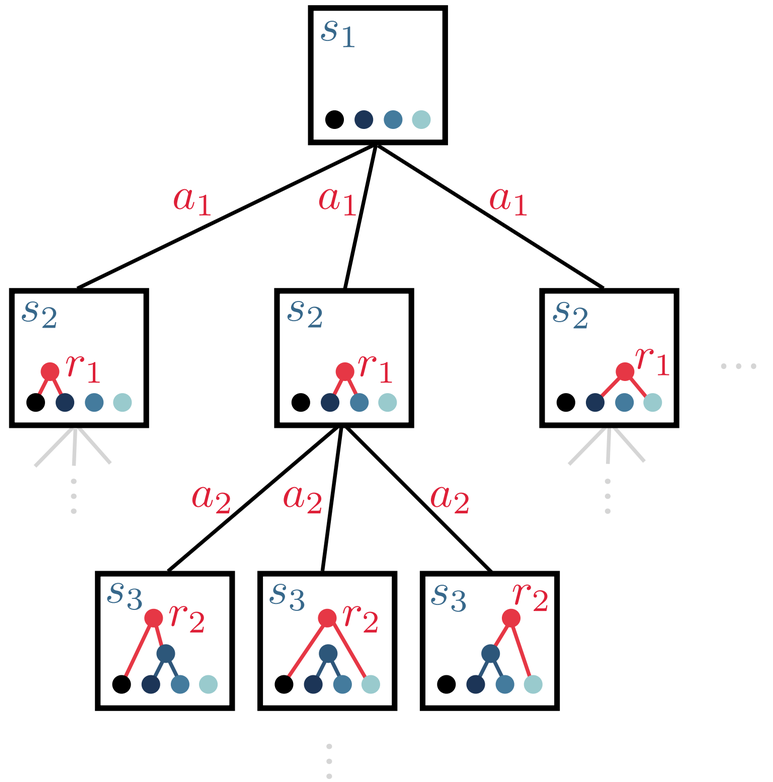


Figure 1: Jet clustering as a Markov Decision Process. States s (squares) represent (partial) clusterings of the original particles (small circles), the agent begins in the unclustered state (top square). Each action a chooses a pair of particles in the current state (which may be either part of the original particle set or the result of a partial clustering) to be merged next. The reward r is the log likelihood of the corresponding $1 \to 2$ splitting.

Reinforcement Learning

Johann Brehmer, Sebastian Macaluso, Duccio Pappadopulo, KC [arXiv:2011.08191]
NeurIPS 2020 ML4Physical Sciences workshop

In particular, we compared the performance of

- Monte Carlo Tree Search (MCTS)
- Imitation learning / Behavioral Cloning (BC).

to traditional baselines:

- Greedy
- Beam Search

And the exact MLE/MAP found by using the trellis algorithm

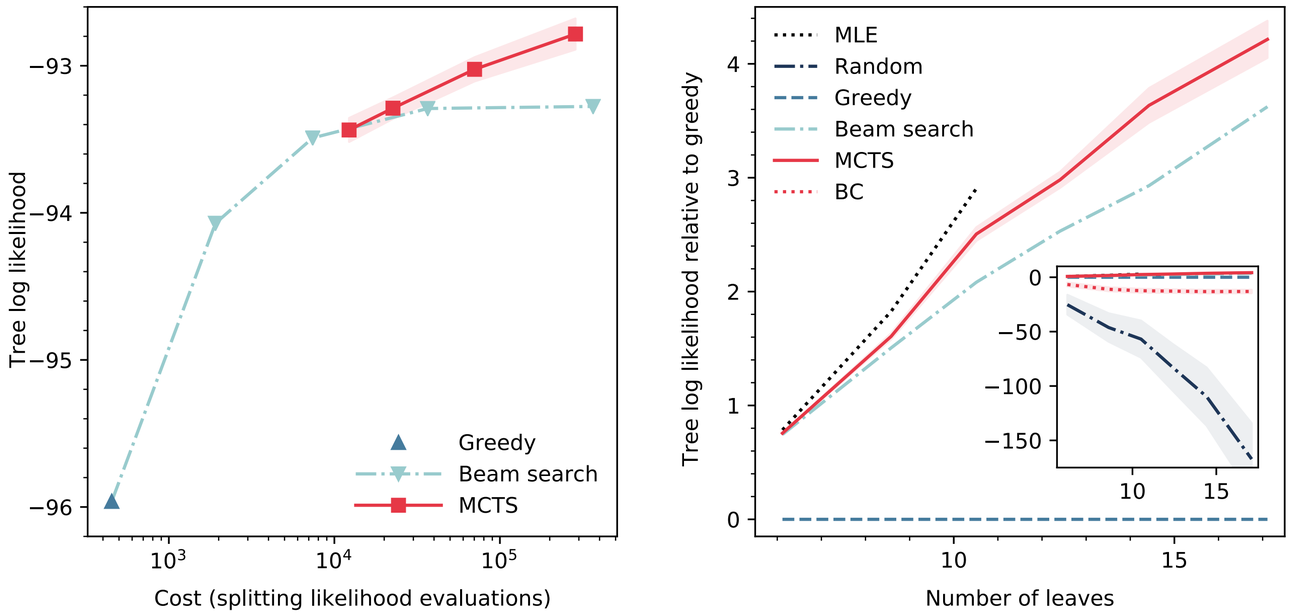


Figure 2: Mean log likelihood of clustered trees (larger is better). We show the mean and its standard error between five models trained with different random seeds. **Left**: against the computational cost, measured as the number of evaluations of the splitting likelihood p_s required by the different algorithms. For beam search and MCTS we show four different hyperparameter settings. **Right**: as a function of the number of final-state particles (leaves of the tree), using the best-performing (and most computationally expensive) hyperparameter setup for each algorithm. MCTS (solid, red) gives the highest-quality tree clusterings.

Conclusion

Hierarchical clustering is a common task in particle physics and many other areas

- The probabilistic view of hierarchical clustering as inverting generative model to infer a structured latent space is interesting and connects mechanistic models, variational inference, deep learning, and combinatorial optimization

The trellis data structure + dynamic programs for MAP and partition function look like a more classical algorithm, but

- The trellis represents a parametrized distribution over hierarchies
- We can differentiate the MAP and partition function w.r.t. these parameters
- We can parametrize this distribution with some model, like a neural network, and have amortized posteriors with efficient sampling (great for a VAE over discrete structures)
- All of these could be useful for reinforcement learning or model-based planning