Complexity of Jet Reconstruction Algorithms

Sean Ericson

UO
PHYS 662

March 18, 2024

- Jet Reconstruction algorithms are at the core of modern HEP data collection and analysis

Complexity of Jet Algorithms

- Jet Reconstruction algorithms are at the core of modern HEP data collection and analysis
- Efficient algorithms are necessary to make analysis of large datasets feasible.

Complexity of Jet Algorithms

- Jet Reconstruction algorithms are at the core of modern HEP data collection and analysis
- Efficient algorithms are necessary to make analysis of large datasets feasible.
- Outline

Complexity of Jet Algorithms

- Jet Reconstruction algorithms are at the core of modern HEP data collection and analysis
- Efficient algorithms are necessary to make analysis of large datasets feasible.
- Outline

Complexity of Jet Algorithms

- Jet Reconstruction algorithms are at the core of modern HEP data collection and analysis
- Efficient algorithms are necessary to make analysis of large datasets feasible.
- Outline
  - Review of jet algorithms

Complexity of Jet Algorithms

- Jet Reconstruction algorithms are at the core of modern HEP data collection and analysis
- Efficient algorithms are necessary to make analysis of large datasets feasible.
- Outline
  - Review of jet algorithms
  - Measuring Algorithmic Complexity

Complexity of Jet Algorithms

- Jet Reconstruction algorithms are at the core of modern HEP data collection and analysis
- Efficient algorithms are necessary to make analysis of large datasets feasible.
- Outline
  - Review of jet algorithms
  - Measuring Algorithmic Complexity
  - Analysis of FastJet, SISCone, and their predecessors

Review of Jet Algorithms

- Jet algorithm goals:

Review of Jet Algorithms

- Jet algorithm goals:
  - Combine data from calorimeters and trackers to define jets

Review of Jet Algorithms

- Jet algorithm goals:
  - Combine data from calorimeters and trackers to define jets
  - Identify subjets and jet substructure

Review of Jet Algorithms

- Jet algorithm goals:
  - Combine data from calorimeters and trackers to define jets
  - Identify subjets and jet substructure
  - Measure invariant masses of constituent particles

Review of Jet Algorithms

- Jet algorithm goals:
  - Combine data from calorimeters and trackers to define jets
  - Identify subjets and jet substructure
  - Measure invariant masses of constituent particles
  - Identify missing energy/momentum

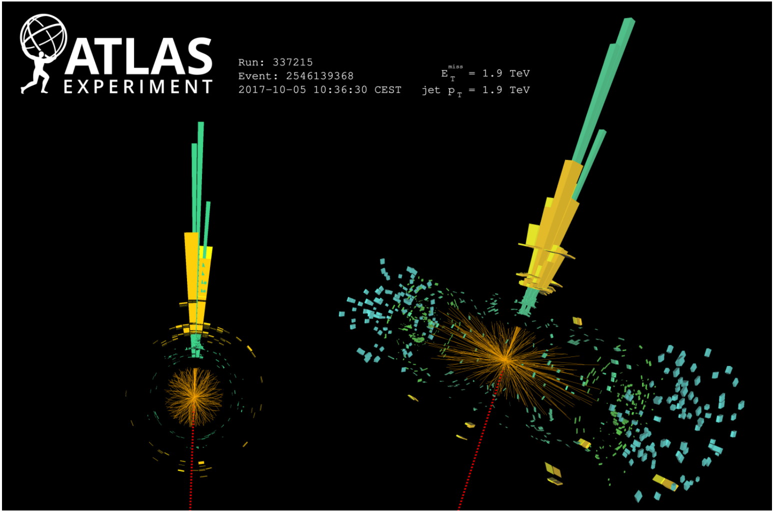
- Jet algorithm goals:
  - Combine data from calorimeters and trackers to define jets
  - Identify subjets and jet substructure
  - Measure invariant masses of constituent particles
  - Identify missing energy/momentum
- Jets:

Review of Jet Algorithms

- Jet algorithm goals:
  - Combine data from calorimeters and trackers to define jets
  - Identify subjets and jet substructure
  - Measure invariant masses of constituent particles
  - Identify missing energy/momentum
- Jets:
  - A collimated spray of stable particles arising from fragmentation and hadronization of a parton after a collision.

Review of Jet Algorithms (cont.)

Example Jet



Review of Jet Algorithms (cont.)

Considerations

Review of Jet Algorithms (cont.)

Considerations

- Jet size

Review of Jet Algorithms (cont.)

Considerations

- Jet size
  - Must be large enough to capture all particles for correct p_μ calculation.

Review of Jet Algorithms (cont.)

Considerations

- Jet size
  - Must be large enough to capture all particles for correct p_μ calculation.
  - However, larger jets are more susceptible to underlying events (UE) and pileup (PU)

Review of Jet Algorithms (cont.)

Considerations

- Jet size
  - Must be large enough to capture all particles for correct p_μ calculation.
  - However, larger jets are more susceptible to underlying events (UE) and pileup (PU)
- Jet shape

Review of Jet Algorithms (cont.)

Considerations

- Jet size
  - Must be large enough to capture all particles for correct p_μ calculation.
  - However, larger jets are more susceptible to underlying events (UE) and pileup (PU)
- Jet shape
  - Algs with variable jet shape can allow for good resolution of subjets at the expense of greater susceptibility to UE and PU

Review of Jet Algorithms (cont.)

Considerations

- Jet size
  - Must be large enough to capture all particles for correct p_μ calculation.
  - However, larger jets are more susceptible to underlying events (UE) and pileup (PU)
- Jet shape
  - Algs with variable jet shape can allow for good resolution of subjets at the expense of greater susceptibility to UE and PU
- Infrared and Collinear safety (IRC safety)

Review of Jet Algorithms (cont.)

Considerations

- Jet size
  - Must be large enough to capture all particles for correct p_μ calculation.
  - However, larger jets are more susceptible to underlying events (UE) and pileup (PU)
- Jet shape
  - Algs with variable jet shape can allow for good resolution of subjets at the expense of greater susceptibility to UE and PU
- Infrared and Collinear safety (IRC safety)
  - Calculated jets should be insensitive to emission of soft radiation from, or collinear splitting of the constituent particles.

Review of Jet Algorithms (cont.)

Considerations

- Jet size
  - Must be large enough to capture all particles for correct p_μ calculation.
  - However, larger jets are more susceptible to underlying events (UE) and pileup (PU)
- Jet shape
  - Algs with variable jet shape can allow for good resolution of subjets at the expense of greater susceptibility to UE and PU
- Infrared and Collinear safety (IRC safety)
  - Calculated jets should be insensitive to emission of soft radiation from, or collinear splitting of the constituent particles.
  - Necessary for faithful comparison of data to experiment

Review of Jet Algorithms (cont.)

Class of Jet Algorithms

Review of Jet Algorithms (cont.)

Class of Jet Algorithms

- Cone Algorithms

Review of Jet Algorithms (cont.)

Class of Jet Algorithms

- Cone Algorithms
  - Assume particles in jets will appear in conical regions; cluster in (η, ϕ)-space.

Review of Jet Algorithms (cont.)

Class of Jet Algorithms

- Cone Algorithms
  - Assume particles in jets will appear in conical regions; cluster in (η, ϕ)-space.
  - Easy to implement, but typically contain unphysical parameters which hinder analysis

Review of Jet Algorithms (cont.)

Class of Jet Algorithms

- Cone Algorithms
  - Assume particles in jets will appear in conical regions; cluster in (η, ϕ)-space.
  - Easy to implement, but typically contain unphysical parameters which hinder analysis
  - Most are not IRC safe

Review of Jet Algorithms (cont.)

Class of Jet Algorithms

- Cone Algorithms
  - Assume particles in jets will appear in conical regions; cluster in (η, ϕ)-space.
  - Easy to implement, but typically contain unphysical parameters which hinder analysis
  - Most are not IRC safe
  - Examples include IC-PR, IC-SM, and SISCone.

Review of Jet Algorithms (cont.)

Class of Jet Algorithms

- Cone Algorithms
  - Assume particles in jets will appear in conical regions; cluster in (η, ϕ)-space.
  - Easy to implement, but typically contain unphysical parameters which hinder analysis
  - Most are not IRC safe
  - Examples include IC-PR, IC-SM, and SISCone.
- Sequential Clustering Algorithms

Review of Jet Algorithms (cont.)

Class of Jet Algorithms

- Cone Algorithms
  - Assume particles in jets will appear in conical regions; cluster in (η, ϕ)-space.
  - Easy to implement, but typically contain unphysical parameters which hinder analysis
  - Most are not IRC safe
  - Examples include IC-PR, IC-SM, and SISCone.
- Sequential Clustering Algorithms
  - Assume particles in jets have small differences in p_t; cluster in momentum space.

Review of Jet Algorithms (cont.)

Class of Jet Algorithms

- Cone Algorithms
  - Assume particles in jets will appear in conical regions; cluster in (η, ϕ)-space.
  - Easy to implement, but typically contain unphysical parameters which hinder analysis
  - Most are not IRC safe
  - Examples include IC-PR, IC-SM, and SISCone.
- Sequential Clustering Algorithms
  - Assume particles in jets have small differences in p_t; cluster in momentum space.
  - Automatically IRC safe

Review of Jet Algorithms (cont.)

Class of Jet Algorithms

- Cone Algorithms
  - Assume particles in jets will appear in conical regions; cluster in (η, ϕ)-space.
  - Easy to implement, but typically contain unphysical parameters which hinder analysis
  - Most are not IRC safe
  - Examples include IC-PR, IC-SM, and SISCone.
- Sequential Clustering Algorithms
  - Assume particles in jets have small differences in p_t; cluster in momentum space.
  - Automatically IRC safe
  - Examples include K_t, $\bar{K}_t$, and Cambridge/Aachen

Measuring Algorithmic Complexity

Assymptotics and Big-O Notation

Measuring Algorithmic Complexity

Assymptotics and Big-O Notation

- Algorithm runtime depends on the specific hardware its running on, and on the input to the particular problem instance

Measuring Algorithmic Complexity

Assymptotics and Big-O Notation

- Algorithm runtime depends on the specific hardware its running on, and on the input to the particular problem instance
- How can we compare algorithms in a context-independent way?

Measuring Algorithmic Complexity

Assymptotics and Big-O Notation

- Algorithm runtime depends on the specific hardware its running on, and on the input to the particular problem instance
- How can we compare algorithms in a context-independent way?
  - Big-O Notation! (a.k.a Bachmann-Landau/asymptotic notation)

Measuring Algorithmic Complexity

Assymptotics and Big-O Notation

- Algorithm runtime depends on the specific hardware its running on, and on the input to the particular problem instance
- How can we compare algorithms in a context-independent way?
  - Big-O Notation! (a.k.a Bachmann-Landau/asymptotic notation)

$$f(x) \in \mathcal{O}(g(x))_{x \to \inf} \iff \exists c \; \exists x_0 \; \forall x > x_0 : \; |f(x)| \leq cg(x)$$

Measuring Algorithmic Complexity

Assymptotics and Big-O Notation

- Algorithm runtime depends on the specific hardware its running on, and on the input to the particular problem instance
- How can we compare algorithms in a context-independent way?
  - Big-O Notation! (a.k.a Bachmann-Landau/asymptotic notation)

    $$f(x) \in \mathcal{O}(g(x))_{x \to \inf} \iff \exists c \; \exists x_0 \; \forall x > x_0 : \; |f(x)| \leq cg(x)$$
  - E.g. $\mathcal{O}(\ln(x)) \subset \mathcal{O}(x) \subset \mathcal{O}(x^n) \subset \mathcal{O}(e^x) \subset \mathcal{O}(x!)$

Measuring Algorithmic Complexity

Assymptotics and Big-O Notation

- Algorithm runtime depends on the specific hardware its running on, and on the input to the particular problem instance
- How can we compare algorithms in a context-independent way?
  - Big-O Notation! (a.k.a Bachmann-Landau/asymptotic notation)

    $$f(x) \in \mathcal{O}(g(x))_{x \to \inf} \iff \exists c \; \exists x_0 \; \forall x > x_0 : \; |f(x)| \leq cg(x)$$
  - E.g. $\mathcal{O}(\ln(x)) \subset \mathcal{O}(x) \subset \mathcal{O}(x^n) \subset \mathcal{O}(e^x) \subset \mathcal{O}(x!)$
  - Also, $o()$, $\Theta(x)$, $\Omega()$, $\omega()$ with related definitions

Complexity of K_t (pre-FastJet)

The Naïve Algorithm

Algorithm Naïve K_t

repeat
    for particle pair (i, j) **do**
        $d_{Bi} \leftarrow p_{ti}^2$ ▷ Calculated once for each i
        $R_{ij}^2 \leftarrow (\eta_i - \eta_j)^2 + (\phi_i - \phi_j)^2$
        $d_{ij} \leftarrow \min(p_{ti}^2, p_{tj}^2) R_{ij}$
    end for
    $d_{\min} \leftarrow \min(\{d_{ij}\} \cup \{d_{Bi}\})$
    if $d_{\min}$ is d_{Bi} **then**
        Add particle i to list of jets, remove from list of particles
    else
        Merge particles i, j
    end if
until no particles remain

Pre-FastJet K_t Complexity (cont.)

Complexity Analysis

Pre-FastJet K_t Complexity (cont.)

Complexity Analysis

- Outermost loop executes $\mathcal{O}(n)$ times

Pre-FastJet K_t Complexity (cont.)

Complexity Analysis

- Outermost loop executes $\mathcal{O}(n)$ times
  - Calculation of d_{Bi} initially $\mathcal{O}(N)$, updates actually $\mathcal{O}(1)$

Pre-FastJet K_t Complexity (cont.)

Complexity Analysis

- Outermost loop executes $\mathcal{O}(n)$ times
  - Calculation of d_{Bi} initially $\mathcal{O}(N)$, updates actually $\mathcal{O}(1)$
  - Calculation of d_{ij} initially $\mathcal{O}(N^2)$, updates actually $\mathcal{O}(N)$

Pre-FastJet K_t Complexity (cont.)

Complexity Analysis

- Outermost loop executes $\mathcal{O}(n)$ times
  - Calculation of d_{Bi} initially $\mathcal{O}(N)$, updates actually $\mathcal{O}(1)$
  - Calculation of d_{ij} initially $\mathcal{O}(N^2)$, updates actually $\mathcal{O}(N)$
  - Determining $d_{\min}$ is $\mathcal{O}(N^2)$

Pre-FastJet K_t Complexity (cont.)

Complexity Analysis

- Outermost loop executes $\mathcal{O}(n)$ times
  - Calculation of d_{Bi} initially $\mathcal{O}(N)$, updates actually $\mathcal{O}(1)$
  - Calculation of d_{ij} initially $\mathcal{O}(N^2)$, updates actually $\mathcal{O}(N)$
  - Determining $d_{\min}$ is $\mathcal{O}(N^2)$
  - Merging operation is $\mathcal{O}(1)$

Pre-FastJet K_t Complexity (cont.)

Complexity Analysis

- Outermost loop executes $\mathcal{O}(n)$ times
  - Calculation of d_{Bi} initially $\mathcal{O}(N)$, updates actually $\mathcal{O}(1)$
  - Calculation of d_{ij} initially $\mathcal{O}(N^2)$, updates actually $\mathcal{O}(N)$
  - Determining d_{min} is $\mathcal{O}(N^2)$
  - Merging operation is $\mathcal{O}(1)$
- Full time complexity is therefore

$$\mathcal{O}(N + N^2 + N(1 + N + N^2 + 1)) = \mathcal{O}(N^3)$$

Pre-FastJet K_t Complexity (cont.)

Complexity Analysis

- Outermost loop executes $\mathcal{O}(n)$ times
  - Calculation of d_{Bi} initially $\mathcal{O}(N)$, updates actually $\mathcal{O}(1)$
  - Calculation of d_{ij} initially $\mathcal{O}(N^2)$, updates actually $\mathcal{O}(N)$
  - Determining $d_{\min}$ is $\mathcal{O}(N^2)$
  - Merging operation is $\mathcal{O}(1)$
- Full time complexity is therefore

$$\mathcal{O}(N + N^2 + N(1 + N + N^2 + 1)) = \mathcal{O}(N^3)$$

- Also easy to see that the space complexity is $\mathcal{O}(N^2)$.

Geometric Insight

Geometric Insight

- The key to improving K_t is the (rather obvious) geometric insight:

Enter: FastJet

Geometric Insight

- The key to improving K_t is the (rather obvious) geometric insight:
  - Let $d_{\min} = \min(\{d_{ij}\})$, and let $p_{ti} < p_{tj}$.

Enter: FastJet

Geometric Insight

- The key to improving K_t is the (rather obvious) geometric insight:
  - Let $d_{\min} = \min(\{d_{ij}\})$, and let $p_{ti} < p_{tj}$.
  - Then, $\forall l: \ R_{ij} \leq R_{il}$

Enter: FastJet

Geometric Insight

- The key to improving K_t is the (rather obvious) geometric insight:
  - Let $d_{\min} = \min(\{d_{ij}\})$, and let $p_{ti} < p_{tj}$.
  - Then, $\forall l: \; R_{ij} \leq R_{il}$
- That is, particles i and j are geometric *nearest neighbors*

Enter: FastJet

Geometric Insight

- The key to improving K_t is the (rather obvious) geometric insight:
  - Let $d_{\min} = \min(\{d_{ij}\})$, and let $p_{ti} < p_{tj}$.
  - Then, $\forall l: \; R_{ij} \leq R_{il}$
- That is, particles i and j are geometric *nearest neighbors*
- Therefore, we don't need the $\mathcal{O}(N^2)$ table d_{ij}!

Enter: FastJet

Geometric Insight

- The key to improving K_t is the (rather obvious) geometric insight:
  - Let $d_{\min} = \min(\{d_{ij}\})$, and let $p_{ti} < p_{tj}$.
  - Then, $\forall l: \; R_{ij} \leq R_{il}$
- That is, particles i and j are geometric *nearest neighbors*
- Therefore, we don't need the $\mathcal{O}(N^2)$ table d_{ij}!
- We only need an n-element list of nearest neighbors $d_{i\mathcal{G}_i}$

Complexity of FastJet

The Smart Algorithm

Algorithm `FastJet` K_t

for particle i **do**
  $\mathcal{G}_i \leftarrow$ nearest neighbor of particle i
  $d_{i\mathcal{G}_i}, d_{Bi}$ calculated as previously
end for
repeat
  $d_{\min} \leftarrow \min(\{\mathcal{G}_i\} \cup \{d_{Bi}\})$
  Merge or Remove according to $d_{\min}$ as previously
  Update $\mathcal{G}_i$, $d_{i\mathcal{G}_i}$, d_{Bi}.
until no particles remain

Complexity of FastJet (cont.)

Semi-Naïve Complexity Analysis

Complexity of FastJet (cont.)

Semi-Naïve Complexity Analysis

- This appears to have a time complexity of $\mathcal{O}(N^2)$:

Semi-Naïve Complexity Analysis

- This appears to have a time complexity of $\mathcal{O}(N^2)$:
  - Initial construction of $\mathcal{G}_i$ is $\mathcal{O}(N)$.

Complexity of FastJet (cont.)

Semi-Naïve Complexity Analysis

- This appears to have a time complexity of $\mathcal{O}(N^2)$:
  - Initial construction of $\mathcal{G}_i$ is $\mathcal{O}(N)$.
  - Initial calculation of $d_{i\mathcal{G}_i}$, d_{Bi} is $\mathcal{O}(N)$

Semi-Naïve Complexity Analysis

- This appears to have a time complexity of $\mathcal{O}(N^2)$:
  - Initial construction of $\mathcal{G}_i$ is $\mathcal{O}(N)$.
  - Initial calculation of $d_{i\mathcal{G}_i}$, d_{Bi} is $\mathcal{O}(N)$
  - Finding $d_{\min}$ takes $\mathcal{O}(N)$ times, repeated $\mathcal{O}(N)$ times.

Complexity of FastJet (cont.)

Semi-Naïve Complexity Analysis

- This appears to have a time complexity of $\mathcal{O}(N^2)$:
  - Initial construction of $\mathcal{G}_i$ is $\mathcal{O}(N)$.
  - Initial calculation of $d_{i\mathcal{G}_i}$, d_{Bi} is $\mathcal{O}(N)$
  - Finding $d_{\min}$ takes $\mathcal{O}(N)$ times, repeated $\mathcal{O}(N)$ times.
  - Updating $\mathcal{G}_i$, d_{Bi} after a merge or removal takes $\mathcal{O}(N)$ (repeated $\mathcal{O}(N)$ times)

Complexity of FastJet (cont.)

Semi-Naïve Complexity Analysis

- This appears to have a time complexity of $\mathcal{O}(N^2)$:
  - Initial construction of $\mathcal{G}_i$ is $\mathcal{O}(N)$.
  - Initial calculation of $d_{i\mathcal{G}_i}$, d_{Bi} is $\mathcal{O}(N)$
  - Finding $d_{\min}$ takes $\mathcal{O}(N)$ times, repeated $\mathcal{O}(N)$ times.
  - Updating $\mathcal{G}_i$, d_{Bi} after a merge or removal takes $\mathcal{O}(N)$ (repeated $\mathcal{O}(N)$ times)
    - Updating $\mathcal{G}_i$ in $\mathcal{O}(N)$ is a nontrivial geometric affect

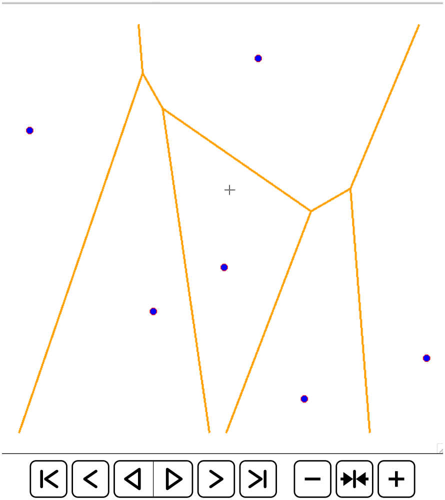
Complexity of FastJet (cont.)

Semi-Naïve Complexity Analysis

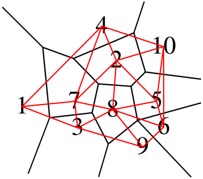
- This appears to have a time complexity of $\mathcal{O}(N^2)$:
  - Initial construction of $\mathcal{G}_i$ is $\mathcal{O}(N)$.
  - Initial calculation of $d_{i\mathcal{G}_i}$, d_{Bi} is $\mathcal{O}(N)$
  - Finding $d_{\min}$ takes $\mathcal{O}(N)$ times, repeated $\mathcal{O}(N)$ times.
  - Updating $\mathcal{G}_i$, d_{Bi} after a merge or removal takes $\mathcal{O}(N)$ (repeated $\mathcal{O}(N)$ times)
    - Updating $\mathcal{G}_i$ in $\mathcal{O}(N)$ is a nontrivial geometric affect
- However, we can do better!

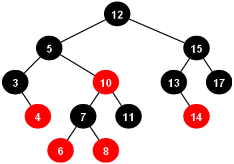
Complexity of FastJet (cont.)

Dynamic Voronoi Diagram



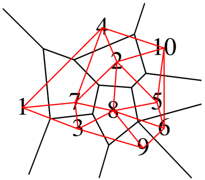
Better Complexity Analysis

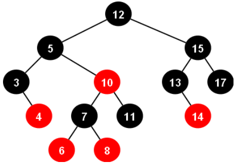




Better Complexity Analysis

- A Voronoi diagram for NN calculations

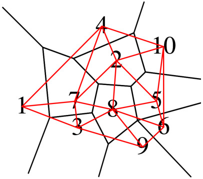


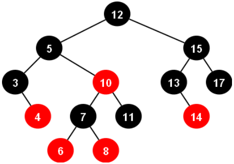


Complexity of FastJet (cont.)

Better Complexity Analysis

- A Voronoi diagram for NN calculations
  - Constructed in $\mathcal{O}(n \ln(N))$ time

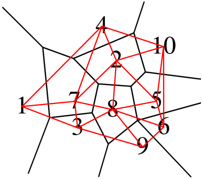


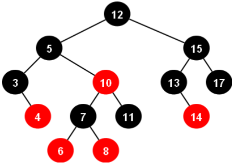


Complexity of FastJet (cont.)

Better Complexity Analysis

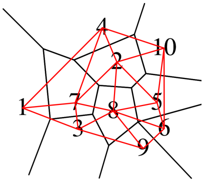
- A Voronoi diagram for NN calculations
  - Constructed in $\mathcal{O}(n \ln(N))$ time
  - Allows for calculation of $\mathcal{G}_i$ in $\mathcal{O}(N)$

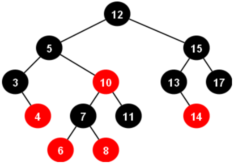




Better Complexity Analysis

- A Voronoi diagram for NN calculations
  - Constructed in $\mathcal{O}(n \ln(N))$ time
  - Allows for calculation of $\mathcal{G}_i$ in $\mathcal{O}(N)$
  - Dynamic updates in $\mathcal{O}(\ln(n))$

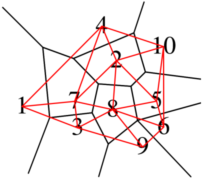


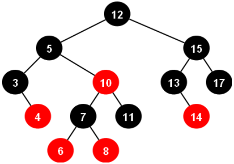


Complexity of FastJet (cont.)

Better Complexity Analysis

- A Voronoi diagram for NN calculations
  - Constructed in $\mathcal{O}(n \ln(N))$ time
  - Allows for calculation of $\mathcal{G}_i$ in $\mathcal{O}(N)$
  - Dynamic updates in $\mathcal{O}(\ln(n))$
- Represent $d_{i\mathcal{G}_i}$ as a Red-Black tree

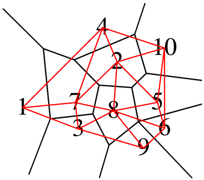


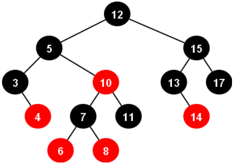


Complexity of FastJet (cont.)

Better Complexity Analysis

- A Voronoi diagram for NN calculations
  - Constructed in $\mathcal{O}(n \ln(N))$ time
  - Allows for calculation of $\mathcal{G}_i$ in $\mathcal{O}(N)$
  - Dynamic updates in $\mathcal{O}(\ln(n))$
- Represent $d_{i\mathcal{G}_i}$ as a Red-Black tree
  - Constructed in $\mathcal{O}(N \ln(N))$ time

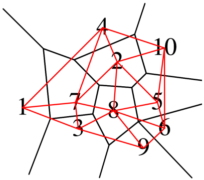


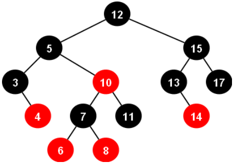


Complexity of FastJet (cont.)

Better Complexity Analysis

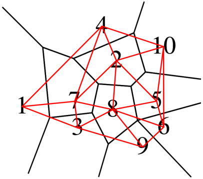
- A Voronoi diagram for NN calculations
  - Constructed in $\mathcal{O}(n \ln(N))$ time
  - Allows for calculation of $\mathcal{G}_i$ in $\mathcal{O}(N)$
  - Dynamic updates in $\mathcal{O}(\ln(n))$
- Represent $d_{i\mathcal{G}_i}$ as a Red-Black tree
  - Constructed in $\mathcal{O}(N \ln(N))$ time
  - Dynamic updates and searching in $\mathcal{O}(\ln(N))$

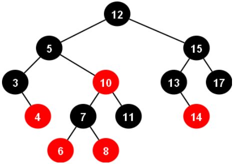




Complexity of FastJet (cont.)

Better Complexity Analysis

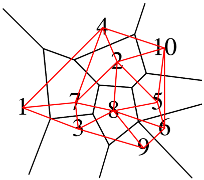




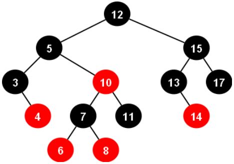
- A Voronoi diagram for NN calculations
  - Constructed in $\mathcal{O}(n \ln(N))$ time
  - Allows for calculation of $\mathcal{G}_i$ in $\mathcal{O}(N)$
  - Dynamic updates in $\mathcal{O}(\ln(n))$
- Represent $d_{i\mathcal{G}_i}$ as a Red-Black tree
  - Constructed in $\mathcal{O}(N \ln(N))$ time
  - Dynamic updates and searching in $\mathcal{O}(\ln(N))$
  - Searches/updates occur $\mathcal{O}(N)$ times; $\mathcal{O}(N \ln(N))$ operations total

Complexity of FastJet (cont.)

Better Complexity Analysis



- A Voronoi diagram for NN calculations
  - Constructed in $\mathcal{O}(n\ln(N))$ time
  - Allows for calculation of $\mathcal{G}_i$ in $\mathcal{O}(N)$
  - Dynamic updates in $\mathcal{O}(\ln(n))$
- Represent $d_{i\mathcal{G}_i}$ as a Red-Black tree
  - Constructed in $\mathcal{O}(N\ln(N))$ time
  - Dynamic updates and searching in $\mathcal{O}(\ln(N))$
  - Searches/updates occur $\mathcal{O}(N)$ times; $\mathcal{O}(N\ln(N))$ operations total
- Total time complexity: $\mathcal{O}(N\ln(N))$.



Complexity of IC-SM

A Naïve Cone-Finding Algorithm

Algorithm Iterative Cone with Split-Merge

repeat
    $p_t^* \leftarrow \max(\{p_{ti}\})$ ▷ seed axis
    repeat
        $p_j \leftarrow$ sum of momenta of particles within R of seed axis
        if $p_t^* == p_j$ **then**
            Label p_t^* a protojet, remove all particles
        else
            $p_t^* \leftarrow p_j$
        end if
    until The jet axis and seed axis coincide
until No seeds remain
Run Split-Merge on the protojets

Complexity of IC-SM

The Split-Merge Procedure

Algorithm Split-Merge

Remove all protojets with $p_t < p_{t,\text{cut}}$
repeat
    Determine hardest protojet i
    Determine hardest protojet j such that i and j share particles
    if no such j exists **then**
        i is a final jet; remove particles
    else
        if $p_{t,\text{shared}} < f \times p_{tj}$ **then**
            Split particles between protojets
        else
            Replace i and j with their merger
        end if
    end if
until no protojets left

Complexity of IC-SM

Complexity Analysis

Complexity of IC-SM

Complexity Analysis

- Stable cone production

Complexity of IC-SM

Complexity Analysis

- Stable cone production
  - Process of finding a stable cone from a seed roughly $\mathcal{O}(Nn)$ where n is the average particle count in a circle of radius R

Complexity of IC-SM

Complexity Analysis

- Stable cone production
  - Process of finding a stable cone from a seed roughly $\mathcal{O}(Nn)$ where n is the average particle count in a circle of radius R
  - This is repeated $\mathcal{O}(N)$ times.

Complexity of IC-SM

Complexity Analysis

- Stable cone production
  - Process of finding a stable cone from a seed roughly $\mathcal{O}(Nn)$ where n is the average particle count in a circle of radius R
  - This is repeated $\mathcal{O}(N)$ times.
  - Codes MCFM and NLOJet implement this in $\mathcal{O}(N2^N)$ by iterating through all possible cones!!

Complexity of IC-SM

Complexity Analysis

- Stable cone production
  - Process of finding a stable cone from a seed roughly $\mathcal{O}(Nn)$ where n is the average particle count in a circle of radius R
  - This is repeated $\mathcal{O}(N)$ times.
  - Codes MCFM and NLOJet implement this in $\mathcal{O}(N2^N)$ by iterating through all possible cones!!
- Split-Merge

Complexity of IC-SM

Complexity Analysis

- Stable cone production
  - Process of finding a stable cone from a seed roughly $\mathcal{O}(Nn)$ where n is the average particle count in a circle of radius R
  - This is repeated $\mathcal{O}(N)$ times.
  - Codes MCFM and NLOJet implement this in $\mathcal{O}(N2^N)$ by iterating through all possible cones!!
- Split-Merge
  - Determining hardest protojet takes $\mathcal{O}(N)$

Complexity of IC-SM

Complexity Analysis

- Stable cone production
  - Process of finding a stable cone from a seed roughly $\mathcal{O}(Nn)$ where n is the average particle count in a circle of radius R
  - This is repeated $\mathcal{O}(N)$ times.
  - Codes MCFM and NLOJet implement this in $\mathcal{O}(N2^N)$ by iterating through all possible cones!!
- Split-Merge
  - Determining hardest protojet takes $\mathcal{O}(N)$
  - Finding hardest overlapping jet takes $\mathcal{O}(N/n)$, repeated $\mathcal{O}(N)$ times.

Complexity of IC-SM

Complexity Analysis

- Stable cone production
  - Process of finding a stable cone from a seed roughly $\mathcal{O}(Nn)$ where n is the average particle count in a circle of radius R
  - This is repeated $\mathcal{O}(N)$ times.
  - Codes MCFM and NLOJet implement this in $\mathcal{O}(N2^N)$ by iterating through all possible cones!!
- Split-Merge
  - Determining hardest protojet takes $\mathcal{O}(N)$
  - Finding hardest overlapping jet takes $\mathcal{O}(N/n)$, repeated $\mathcal{O}(N)$ times.
  - Merging takes $\mathcal{O}(Nn)$

Complexity of IC-SM

Complexity Analysis

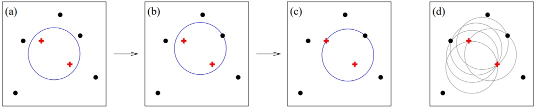
- Stable cone production
  - Process of finding a stable cone from a seed roughly $\mathcal{O}(Nn)$ where n is the average particle count in a circle of radius R
  - This is repeated $\mathcal{O}(N)$ times.
  - Codes MCFM and NLOJet implement this in $\mathcal{O}(N2^N)$ by iterating through all possible cones!!
- Split-Merge
  - Determining hardest protojet takes $\mathcal{O}(N)$
  - Finding hardest overlapping jet takes $\mathcal{O}(N/n)$, repeated $\mathcal{O}(N)$ times.
  - Merging takes $\mathcal{O}(Nn)$
- Total complexity: $\mathcal{O}(N^2 n)$ (can be optimized to $\mathcal{O}(N^2)$ with 2D tree data structures)

Complexity of SISCone

Algorithm SISCone

```
for particle i do
    Find all particles j within 2R of i
    if no such j exists then
        i is made a protojet
    else
        Find circles of radius R with i j on their circumference
        for each circle do
            for each permutations of edge point containment do
                Label circle as cone
                Check cone stability w.r.t the edge points
            end for
        end for
    end if
end for
Explicitly check all cones not labeled unstable for stability
```

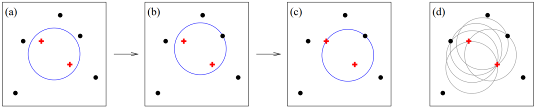
Improved Cone-Finding



SISCone's geometric approach to seedless cone-finding.

Complexity of SISCone

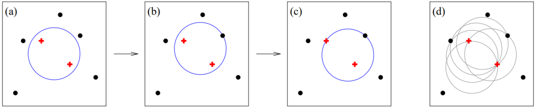
Improved Cone-Finding



SISCone's geometric approach to seedless cone-finding.

- SISCone again uses geometric insight to optimize cone finding.

Improved Cone-Finding

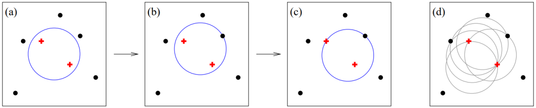


SISCone's geometric approach to seedless cone-finding.

- SISCone again uses geometric insight to optimize cone finding.
- Able to provably find all stable cones in $\mathcal{O}(Nn\ln(n))$ time!

Complexity of SISCone

Improved Cone-Finding



SISCone's geometric approach to seedless cone-finding.

- SISCone again uses geometric insight to optimize cone finding.
- Able to provably find all stable cones in $\mathcal{O}(Nn\ln(n))$ time!
- Seedlessness $\implies$ inherent IR safety

Conclusion

- Jet Reconstruction algorithms are central to modern HEP experiment and theory

Conclusion

- Jet Reconstruction algorithms are central to modern HEP experiment and theory
- Algorithms need to be fast; algorithmic complexity quantifies this.

Conclusion

- Jet Reconstruction algorithms are central to modern HEP experiment and theory
- Algorithms need to be fast; algorithmic complexity quantifies this.
- Seemingly the same algorithm can be implement in ways with different complexities

Conclusion

- Jet Reconstruction algorithms are central to modern HEP experiment and theory
- Algorithms need to be fast; algorithmic complexity quantifies this.
- Seemingly the same algorithm can be implement in ways with different complexities
  - K_t: $\mathcal{O}(N^3) \to \mathcal{O}(N \ln(N))$

Conclusion

- Jet Reconstruction algorithms are central to modern HEP experiment and theory
- Algorithms need to be fast; algorithmic complexity quantifies this.
- Seemingly the same algorithm can be implement in ways with different complexities
  - K_t: $\mathcal{O}(N^3) \to \mathcal{O}(N \ln(N))$
  - Cone Algs: $\mathcal{O}(N2^N) \to \mathcal{O}(N^3) \to \mathcal{O}(N^2 \ln(N))$

Conclusion

- Jet Reconstruction algorithms are central to modern HEP experiment and theory
- Algorithms need to be fast; algorithmic complexity quantifies this.
- Seemingly the same algorithm can be implement in ways with different complexities
  - K_t: $\mathcal{O}(N^3) \rightarrow \mathcal{O}(N \ln(N))$
  - Cone Algs: $\mathcal{O}(N2^N) \rightarrow \mathcal{O}(N^3) \rightarrow \mathcal{O}(N^2 \ln(N))$
- We, as physicists, should be embarrassed!

Conclusion

- Jet Reconstruction algorithms are central to modern HEP experiment and theory
- Algorithms need to be fast; algorithmic complexity quantifies this.
- Seemingly the same algorithm can be implement in ways with different complexities
  - K_t: $\mathcal{O}(N^3) \to \mathcal{O}(N \ln(N))$
  - Cone Algs: $\mathcal{O}(N2^N) \to \mathcal{O}(N^3) \to \mathcal{O}(N^2 \ln(N))$
- We, as physicists, should be embarrassed!
  - (particularly by those $N2^N$ codes...)

Conclusion

- Jet Reconstruction algorithms are central to modern HEP experiment and theory
- Algorithms need to be fast; algorithmic complexity quantifies this.
- Seemingly the same algorithm can be implement in ways with different complexities
  - K_t: $\mathcal{O}(N^3) \rightarrow \mathcal{O}(N \ln(N))$
  - Cone Algs: $\mathcal{O}(N2^N) \rightarrow \mathcal{O}(N^3) \rightarrow \mathcal{O}(N^2 \ln(N))$
- We, as physicists, should be embarrassed!
  - (particularly by those $N2^N$ codes...)
- Have a good spring break!

References

- Ryan Atkin 2015 J. Phys.: Conf. Ser. 645 012008
- G. P. Salam and G. Soyez, JHEP 0705 (2007) 086 [arXiv:0704.0292 [hep-ph]].
- M. Cacciari and G. P. Salam, arXiv:hep-ph/0512210.
- S. Fortune, in Proceedings of the second annual symposium on Computational geometry, p. 312 (1986).