Jet-finding for LHC

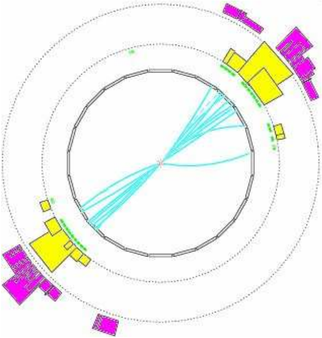
Gavin Salam
work with M. Cacciari and (in progress) G. Soyez

LPTHE, Universities of Paris VI and VII and CNRS

Robi is 60, SPhT Saclay
24 November 2006

Introduction

Jet finding at LEP/HERA:



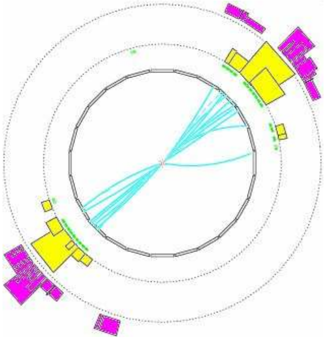
- Clean environment
- Moderate number of particles ($\lesssim 50$)

At LHC it will be more complex:

- Up to ~ 25 simultaneous pp collisions (10 in above event)
- Thousands of particles

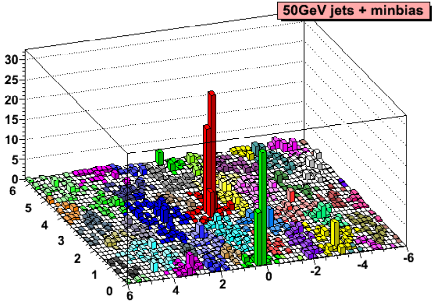
Introduction

Jet finding at LEP/HERA:



- Clean environment
- Moderate number of particles ($\lesssim 50$)

At LHC it will be more complex:



- Up to $\sim$ 25 simultaneous pp collisions (10 in above event)
- Thousands of particles

New issues at LHC

Two characteristically new issues appear for jet finding in the complex environment of of LHC:

- How, computationally, to deal with the clustering of thousands of particles

  Solutions exploit interesting links between jet-finding and computational geometry

- How to reconstruct correct jet kinematics despite the significant additional energy from the "pileup" events

  Useful to introduce the concept of a jet area

Jet finder defs.

Clustering jet finders

1. Calculate 'distances'
   - d_{ij} between all particles i and j
   - d_{iB} between i and beam
2. Find smallest of d_{ij} and d_{iB}
   - If d_{ij} is smallest, recombine i and j
   - if d_{iB} is smallest call i a jet
3. Goto step 1 if anything's left

Two variants (& one parameter, R)

- **k_t jet finder** [1991]

  $$d_{ij} = \min(k_{ti}^2, k_{tj}^2)\Delta R_{ij}^2, \quad d_{iB} = k_{ti}^2 R^2$$

- **Cambridge/Aachen** [1998]

  $$d_{ij} = \Delta R_{ij}^2, \quad d_{iB} = R^2 \qquad [\Delta R_{ij}^2 = \Delta y_{ij}^2 + \Delta\phi_{ij}^2]$$

Jet finder defs.

Clustering jet finders

1. Calculate 'distances'
   - d_{ij} between all particles i and j
   - d_{iB} between i and beam
2. Find smallest of d_{ij} and d_{iB}
   - If d_{ij} is smallest, recombine i and j
   - if d_{iB} is smallest call i a jet
3. Goto step 1 if anything's left

Two variants (& one parameter, R)

- **$\mathbf{k_t}$ jet finder** [1991]

$$d_{ij} = \min(k_{ti}^2, k_{tj}^2)\Delta R_{ij}^2, \quad d_{iB} = k_{ti}^2 R^2$$

- **Cambridge/Aachen** [1998]

$$d_{ij} = \Delta R_{ij}^2, \quad d_{iB} = R^2 \qquad [\Delta R_{ij}^2 = \Delta y_{ij}^2 + \Delta\phi_{ij}^2]$$

Jet finder defs.

Clustering jet finders

1. Calculate 'distances'
   - d_{ij} between all particles i and j
   - d_{iB} between i and beam
2. Find smallest of d_{ij} and d_{iB}
   - If d_{ij} is smallest, recombine i and j
   - if d_{iB} is smallest call i a jet
3. Goto step 1 if anything's left

Two variants (& one parameter, R)

- **$\mathbf{k_t}$ jet finder** [1991]

$$d_{ij} = \min(k_{ti}^2, k_{tj}^2)\Delta R_{ij}^2, \quad d_{iB} = k_{ti}^2 R^2$$

- **Cambridge/Aachen** [1998]

$$d_{ij} = \Delta R_{ij}^2, \quad d_{iB} = R^2 \qquad [\Delta R_{ij}^2 = \Delta y_{ij}^2 + \Delta\phi_{ij}^2]$$

Cone jet finders e.g.

1. Create a seed (3-vector) from the direction of each input particles (possibly implement a way to specify a smaller list of seeds to save processing time ie. calo clusters).
2. For each seed, s, create a cone in η-ϕ space of radius R (set by the parameter radius around the seed axis such that a particle, p, with

$$(\eta_s - \eta_p)^2 + (\phi_s - \phi_p)^2 < R^2 \qquad (1)$$

   is defined to be inside the cone.
3. Then combine every particle in this cone into a jet using a $p_\perp$ recombination scheme as described in section 2.5.2 of the KtJet paper.
4. Now create a new cone around this jet's axis and repeat step 3. If the new jet's axis is colinear with the previous axis then the jet is stable and is added to the list of meta-jets, otherwise the process is repeated until either a stable jet if found or a maximum number of iterations is reached.
5. The next stage is, to enforce infra-red safety, to repeat steps 2-4 with a new set of seeds in-between every pair of jets i, j, found above if i and j are between 1 and 2 cone radii apart ie.

   if:

$$R^2 < (\eta_i - \eta_j)^2 + (\phi_i - \phi_j)^2 < (2R)^2 \qquad (2)$$

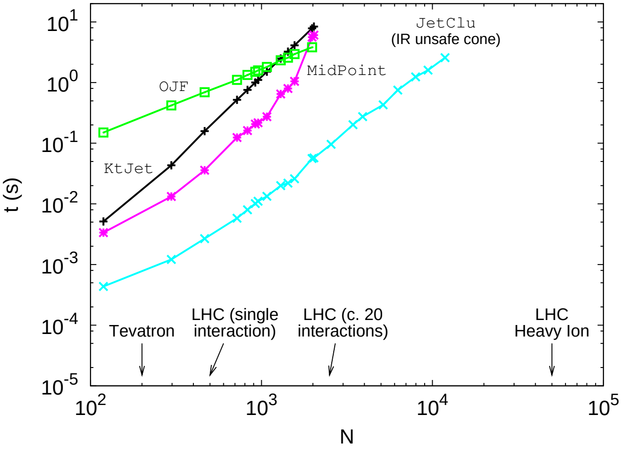
   then:

$$\eta_s = \frac{\eta_i + \eta_j}{2} \qquad \phi_s = \frac{\phi_i + \phi_j}{2} \qquad (3)$$

6. Next any jets with $p_\perp$ less than a pre-defined parameter epslon (typically of order 5 GeV) are removed from the list.
7. Then for each jet in the list, if the sum of the $p_\perp$s of any particles in the jet which are shared with a higher $p_\perp$ jet is greater than some fraction, ovlim, of this jet's $p_\perp$, then remove the jet from the list.
8. Next for each particle that is still in more than one jet, remove the particle from all but the closest jet to particle's direction, ie. the jet with the smallest $\Delta(\eta)^2 - \Delta(\phi)^2$.
9. Finally step 6 is repeated.

[from W. Plano]

Time to cluster N particles



Standard C++ (and fortran) k_t-clustering takes time $\sim N^3$.

a Pb-Pb event takes 1 day!

JetClu (cone) *is fast*, but IR unsafe at NLO.

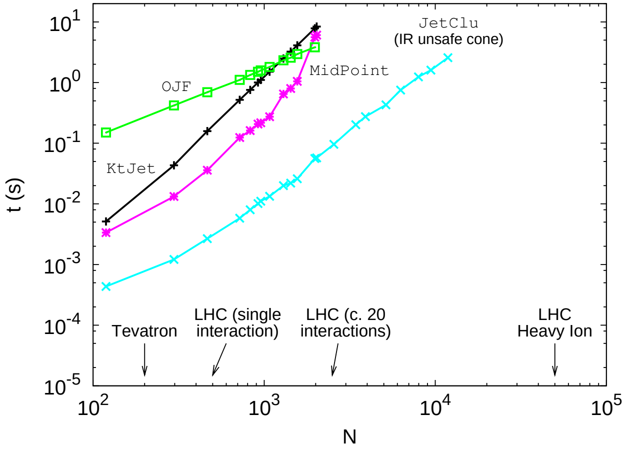
being phased out at Tevatron

IR-safer cone (Mid-point) is as slow as k_t

IR unsafe at NNLO

Jet-clustering speed is an issue for high-luminosity *pp* ($\sim 10^8$ events) and Pb-Pb ($\sim 10^7$ events) collisions at LHC.

Time to cluster N particles



Standard C++ (and fortran) k_t-clustering takes time $\sim N^3$.

a Pb-Pb event takes 1 day!

JetClu (cone) *is fast*, but IR unsafe at NLO.

being phased out at Tevatron

IR-safer cone (Mid-point) is as slow as k_t

IR unsafe at NNLO

Jet-clustering speed is an issue for high-luminosity pp ($\sim 10^8$ events) and Pb-Pb ($\sim 10^7$ events) collisions at LHC.

Why is k_t an N^3 algorithm?

1. Given the initial set of particles, construct a table of all the d_{ij}, d_{iB}.
   [$\mathcal{O}(N^2)$ *operations, done once*]
2. Scan the table to find the minimal value $d_{\min}$ of the d_{ij}, d_{iB}.
   [$\mathcal{O}(\mathbf{N^2})$ operations, done N times]
3. Merge or remove the particles corresponding to $d_{\min}$ as appropriate.
   [$\mathcal{O}(1)$ operations, done N times]
4. Update the table of d_{ij}, d_{iB} to take into account the merging or removal, and if any particles are left go to step 2.
   [$\mathcal{O}(N)$ operations, done N times]

This is the "brute-force" or "naive" method

Can one do better than N^2?

There are $N(N-1)/2$ distances d_{ij} — surely we have to calculate them all in order to find smallest?

k_t distance measure is partly *geometrical:*

- Consider smallest $d_{ij} = \min(k_{ti}^2, k_{tj}^2) R_{ij}^2$
- Suppose $k_{ti} < k_{tj}$
- Then: $R_{ij} <= R_{i\ell}$ for any $\ell \neq j$. [If $\exists\ \ell$ s.t. $R_{i\ell} < R_{ij}$ then $d_{i\ell} < d_{ij}$]

In words: if i, j form smallest d_{ij} then j is geometrical nearest neighbour (GNN) of i.

k_t distance need only be calculated between GNNs

Each point has 1 GNN $\rightarrow$ need only calculate N d_{ij}'s

Can one do better than N^2?

There are $N(N-1)/2$ distances d_{ij} — surely we have to calculate them all in order to find smallest?

k_t distance measure is partly *geometrical:*

- Consider smallest $d_{ij} = \min(k_{ti}^2, k_{tj}^2) R_{ij}^2$
- Suppose $k_{ti} < k_{tj}$
- Then: $R_{ij} <= R_{i\ell}$ for any $\ell \neq j$. [If $\exists\ \ell$ s.t. $R_{i\ell} < R_{ij}$ then $d_{i\ell} < d_{ij}$]

In words: if i, j form smallest d_{ij} then j is geometrical nearest neighbour (GNN) of i.

k_t distance need only be calculated between GNNs

Each point has 1 GNN → need only calculate N d_{ij}'s

Can one do better than N^2?

There are $N(N-1)/2$ distances d_{ij} — surely we have to calculate them all in order to find smallest?

k_t distance measure is partly *geometrical:*

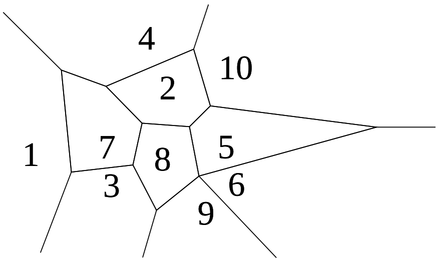
- Consider smallest $d_{ij} = \min(k_{ti}^2, k_{tj}^2) R_{ij}^2$
- Suppose $k_{ti} < k_{tj}$
- Then: $R_{ij} <= R_{i\ell}$ for any $\ell \neq j$. [If $\exists\ \ell$ s.t. $R_{i\ell} < R_{ij}$ then $d_{i\ell} < d_{ij}$]

In words: if i, j form smallest d_{ij} then j is geometrical nearest neighbour (GNN) of i.

k_t distance need only be calculated between GNNs

Each point has 1 GNN $\rightarrow$ need only calculate N d_{ij}'s

Finding Geom Nearest Neighbours



Given a set of vertices on plane (1...10) a *Voronoi diagram* partitions plane into cells containing all points closest to each vertex

Dirichlet '1850, Voronoi '1908

A vertex's nearest other vertex is always in an adjacent cell.

E.g. GNN of point 7 will be found among 1,4,2,8,3 (it turns out to be 3)

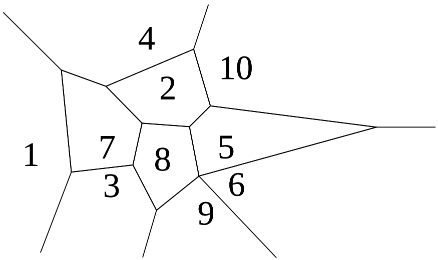
Construction of Voronoi diagram for N points: $N \ln N$ time   Fortune '88

Update of 1 point in Voronoi diagram: $\ln N$ time

Devillers '99 [+ related work by other authors]

Convenient C++ package available: **CGAL**   http://www.cgal.org

Finding Geom Nearest Neighbours



Given a set of vertices on plane (1...10) a *Voronoi diagram* partitions plane into cells containing all points closest to each vertex

Dirichlet '1850, Voronoi '1908

A vertex's nearest other vertex is always in an adjacent cell.

E.g. GNN of point 7 will be found among 1,4,2,8,3 (it turns out to be 3)

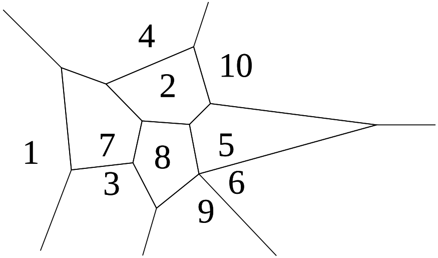
Construction of Voronoi diagram for N points: $N \ln N$ time Fortune '88

Update of 1 point in Voronoi diagram: $\ln N$ time

Devillers '99 [+ related work by other authors]

Convenient C++ package available: **CGAL** http://www.cgal.org

Finding Geom Nearest Neighbours



Given a set of vertices on plane (1...10) a *Voronoi diagram* partitions plane into cells containing all points closest to each vertex

Dirichlet '1850, Voronoi '1908

A vertex's nearest other vertex is always in an adjacent cell.

E.g. GNN of point 7 will be found among 1,4,2,8,3 (it turns out to be 3)

Construction of Voronoi diagram for N points: $N \ln N$ time Fortune '88

Update of 1 point in Voronoi diagram: $\ln N$ time

Devillers '99 [+ related work by other authors]

Convenient C++ package available: **CGAL** `http://www.cgal.org`

Assembling fast k_t clustering

The FastJet algorithm:

Construct the Voronoi diagram of the N particles with CGAL **$\mathcal{O}$ (N ln N)**

Find the GNN of each of the N particles, calculate d_{ij} store result in a *priority queue* (`C++ map`) **$\mathcal{O}$ (N ln N)**

Repeat following steps **N** times:

- Find smallest d_{ij}, merge/eliminate i, j **N** $\times$ **$\mathcal{O}$ (1)**
- Update Voronoi diagram and distance map **N** $\times$ **$\mathcal{O}$ (ln N)**

Overall an $\mathcal{O}$ (N ln N) algorithm

Cacciari & GPS, hep-ph/0512210
http://www.lpthe.jussieu.fr/~salam/fastjet/
Results identical to standard N^3 implementations

Assembling fast k_t clustering

The FastJet algorithm:

Construct the Voronoi diagram of the N particles with CGAL $\mathcal{O}\left(\mathbf{N} \ln \mathbf{N}\right)$

Find the GNN of each of the N particles, calculate d_{ij} store result in a *priority queue* (`C++ map`) $\mathcal{O}\left(\mathbf{N} \ln \mathbf{N}\right)$

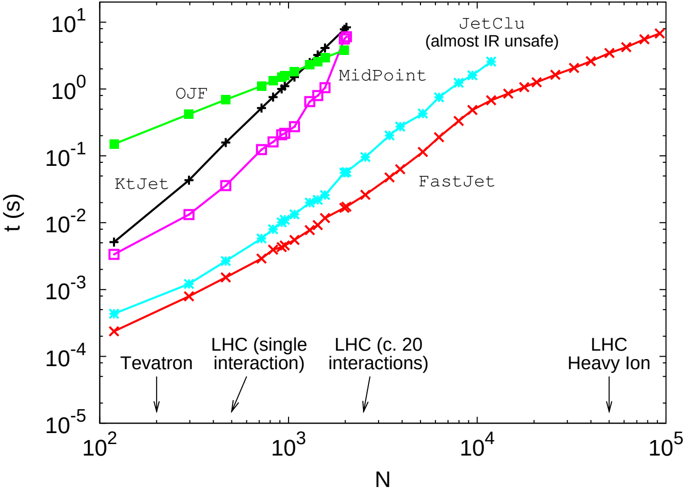
Repeat following steps **N** times:

- Find smallest d_{ij}, merge/eliminate i, j $\mathbf{N} \times \mathcal{O}\left(\mathbf{1}\right)$
- Update Voronoi diagram and distance map $\mathbf{N} \times \mathcal{O}\left(\ln \mathbf{N}\right)$

Overall an $\mathcal{O}\left(\mathbf{N} \ln \mathbf{N}\right)$ algorithm

Cacciari & GPS, hep-ph/0512210
`http://www.lpthe.jussieu.fr/~salam/fastjet/`
Results identical to standard N^3 implementations

FastJet performance (kt)



For $N \gtrsim 10^4$, FastJet algorithm scales as $N \ln N$

For $N \lesssim 10^4$, FastJet switches to a related geometrical N^2 alg.

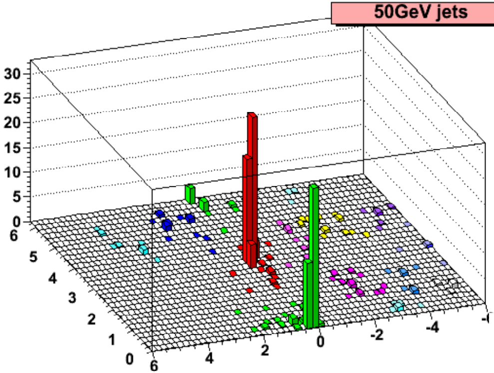
Jets ↔ Comp. Geom.

jet.alg.	Brute force scaling	Geometrical concepts	Geom. scaling
k_t	N^3	dynamic nearest-neighbour graph Dynamic voronoi diagram Devillers '99 (and others)	$N \ln N$
Cam / Aachen	N^3	dynamic closest pairs Shuffles, quad-trees T. Chan '02	$N \ln N$
Seedless Cone	$N\,2^N$	All circular partitions of a 2D set of points (+ range-searching) not clear if studied	$N^{5/2}$

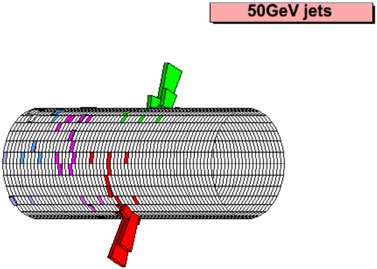
Cam/Aachen: Cacciari & GPS '06

Seedless cone (replaces IR unsafe midpoint cone): GPS & Soyez, in progress

What is speed good for?



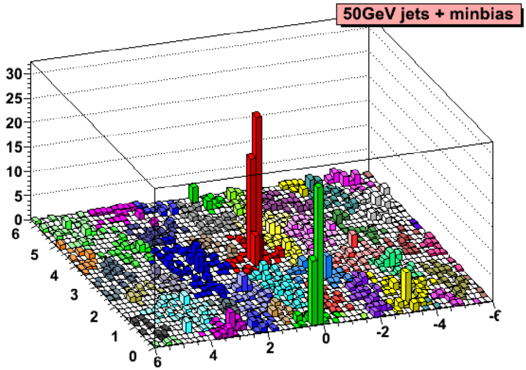
'Standard hard' event
Two well isolated jets



∼ 200 particles

Easy even with old methods

What is speed good for?

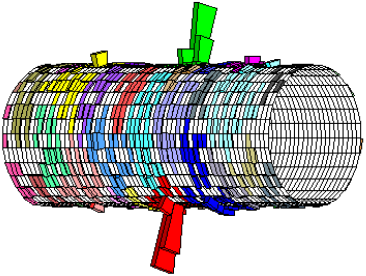


Add 10 min-bias events (moderately high lumi)

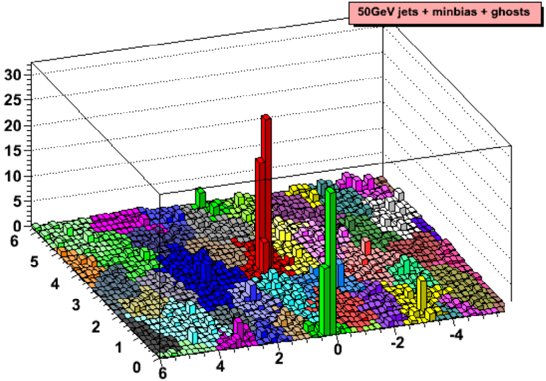
$\sim$ 2000 particles

Clustering takes $\mathcal{O}(10s)$ with old methods.

20ms with `FastJet`.



What is speed good for?



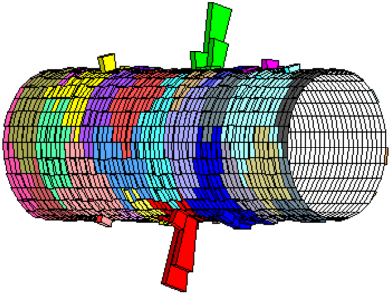
Add dense coverage of infinitely soft *"ghosts"*

See how many end up in jet to measure jet area

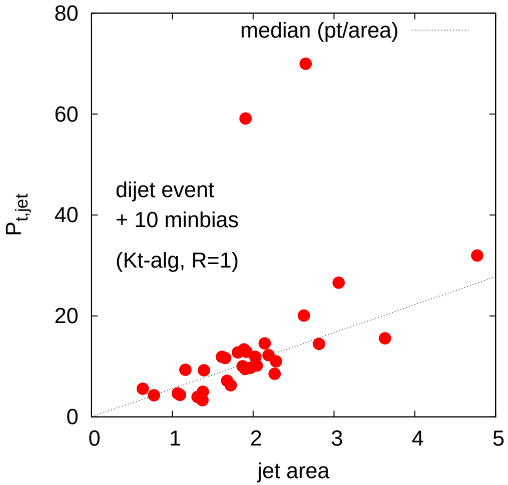
$\sim$ 10000 particles

Clustering takes $\sim$ 20 minutes with old methods.

0.6s with `FastJet`.



Jet areas



Jet areas in k_t algorithm are quite varied

Because k_t-alg adapts to the jet structure

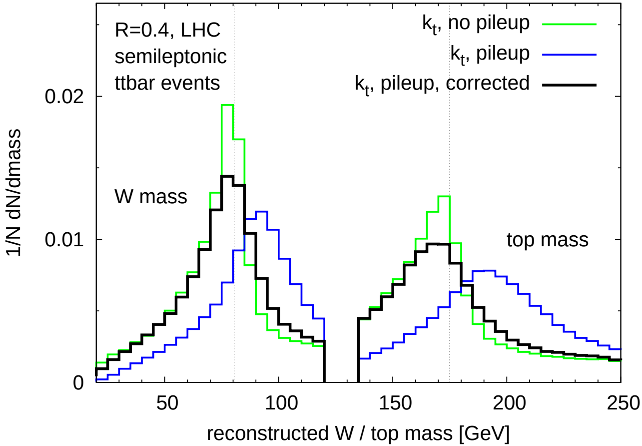
- Contamination from min-bias $\sim$ area

Complicates corrections: min-bias subtraction is different for each jet.

Cone supposedly simpler
Area $= \pi R^2$? (Not quite...)

But: area can be measured for each jet, as can typical median p_t/area.

Application: semi-leptonic $t\bar{t}$ @ LHC



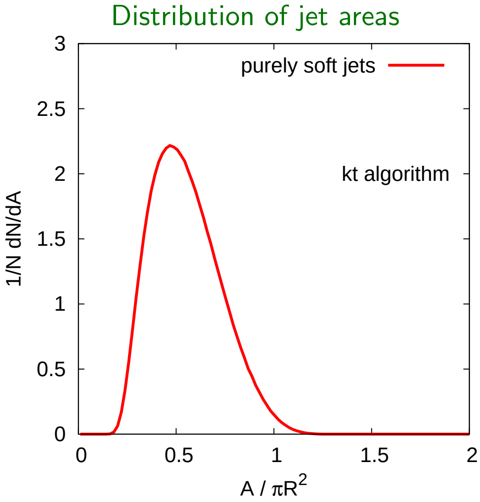
Each jet corrected by area $\times$ median (P_t/area)

Naive analysis: no cuts; assume both b's tagged

Take two hardest non-b jets — call them a W

Take correct sign b, combine with $W \rightarrow$ top

Distribution of jet areas



Since *areas are a crucial quantity* in reconstructing jet kinematics, study them further...

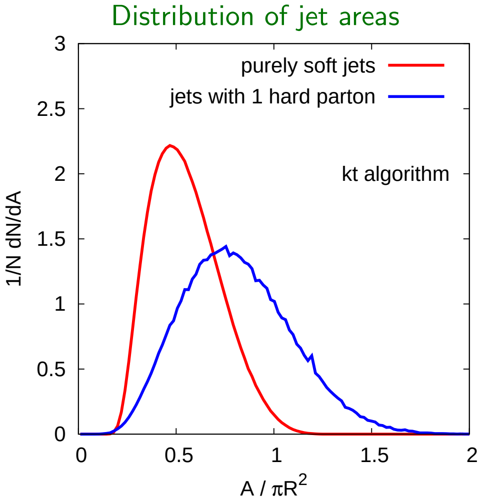
Two simple cases:

1. run clustering on *many soft particles* & look at areas of jets that come out
2. Add *one hard particle,* and examine area of its jet

Conclusion: jet area expands when it is anchored by structure.

- Can one obtain analytical insight into this? To some extent in 1D
- 'Hierarchical clustering' is used in many fields (bio, computing, ...) — are similar features of relevance there?

Distribution of jet areas



Since *areas are a crucial quantity* in reconstructing jet kinematics, study them further...

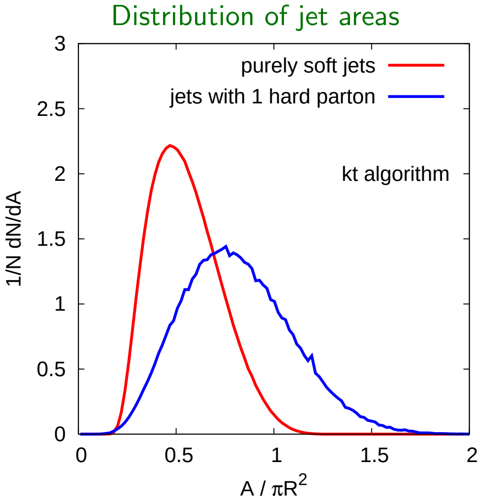
Two simple cases:

1. run clustering on *many soft particles* & look at areas of jets that come out
2. Add *one hard particle,* and examine area of its jet

Conclusion: jet area expands when it is anchored by structure.

- Can one obtain analytical insight into this? To some extent in 1D
- 'Hierarchical clustering' is used in many fields (bio, computing, ...) — are similar features of relevance there?

Distribution of jet areas



Since *areas are a crucial quantity* in reconstructing jet kinematics, study them further. . .

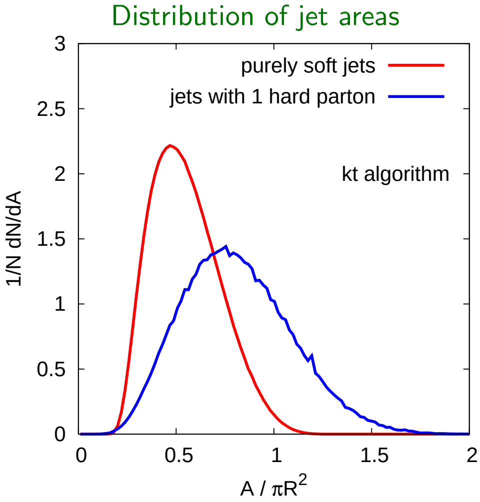
Two simple cases:

1. run clustering on *many soft particles* & look at areas of jets that come out
2. Add *one hard particle,* and examine area of its jet

Conclusion: jet area expands when it is anchored by structure.

- Can one obtain analytical insight into this? To some extent in 1D
- 'Hierarchical clustering' is used in many fields (bio, computing, . . .) — are similar features of relevance there?

Distribution of jet areas



Since *areas are a crucial quantity* in reconstructing jet kinematics, study them further. . .

Two simple cases:

1. run clustering on *many soft particles* & look at areas of jets that come out
2. Add *one hard particle,* and examine area of its jet

Conclusion: jet area expands when it is anchored by structure.

- Can one obtain analytical insight into this? To some extent in 1D
- 'Hierarchical clustering' is used in many fields (bio, computing, . . .) — are similar features of relevance there?

Conclusions

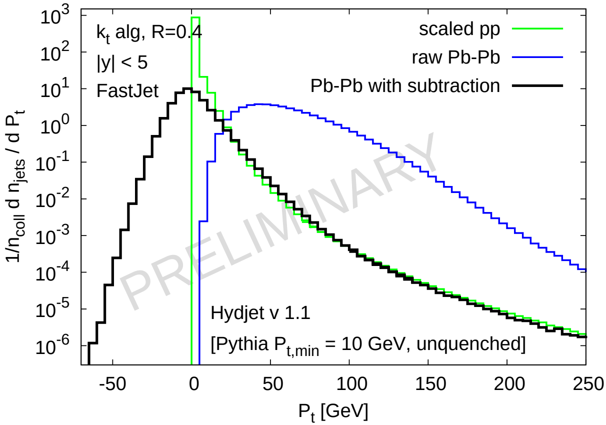
Jet finding in the context of LHC is a rich subject!

- Speed (a major issue) improves enormously when one exploits the *geometrical structures* that underly jet algorithms — especially together with recent developments by computational geometers.
- *Jet areas* are a new concept of particular relevance in *high-noise environments* — much is still to be understood about them.

NB: more is known than could be shown here!

EXTRA MATERIAL

Inclusive jets in Pb-Pb @ LHC



Most HI studies use just particles with $p_t >$ a few GeV IR unsafe

affected by quenching

We use *all* particles and area-based subtraction.

Good results despite the huge subtraction being performed.