Recent developments in jet clustering algorithms

Gavin Salam
work in progress with M. Cacciari

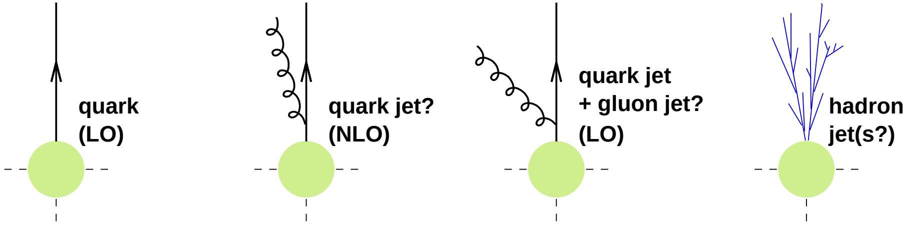
LPTHE, Universities of Paris VI and VII and CNRS

Università di Firenze
29 June 2006

Jets?

Electrons & muons are fundamental, weakly coupled particles — it makes sense physically and experimentally to think of them as concrete objects.

Partons (quarks, gluons) are not so simple...



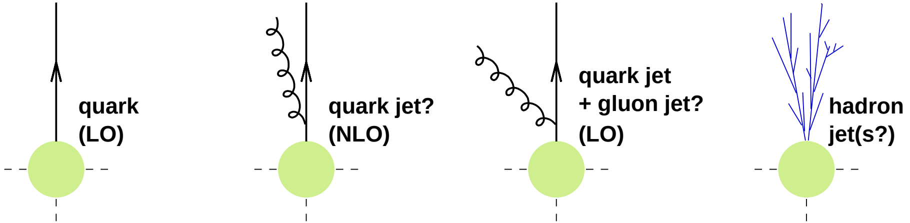
- Partons split into further partons
- Jets are a a way of thinking of the 'original parton'
- A 'jet' is a fundamentally ambiguous concept (e.g. requires a resolution)

Jets are only meaningful once you've defined a **jet algorithm**

Jets?

Electrons & muons are fundamental, weakly coupled particles — it makes sense physically and experimentally to think of them as concrete objects.

Partons (quarks, gluons) are not so simple...



- Partons split into further partons
- Jets are a a way of thinking of the 'original parton'
- A 'jet' is a fundamentally ambiguous concept (e.g. requires a resolution)

Jets are only meaningful once you've defined a jet algorithm

Jet algorithm requirements

What is **needed** of a jet algorithm

- Must be infrared and collinear (IRC) safe

  soft emissions shouldn't change jets

  collinear splitting shouldn't change jets
- Must be identical procedure at parton level, hadron-level

  So that theory calculations can be compared to experimental measurements

What is *nice* for a jet algorithm

- Shouldn't be too sensitive to hadronisation, underlying event, pileup

  Because we can only barely model them
- Should be realistically applicable at detector level

  Not too slow, not too complex to correct
- Should behave 'sensibly'

  e.g. don't want it to spuriously ignore large energy deposits

Jet algorithm types

Mainstream jet-algorithms

- Iterative cone algorithms (JetClu, ILCA/Midpoint, ...)

  Searches for cones centred on regions of energy flow

  Dominant at hadron colliders

- Sequential recombination algorithms (k_t, Cambridge/Aachen, Jade)

  Recombine closest pair of particles, next closest, etc.

  Dominant at e^+e^- and ep colliders

Other approaches

- 'Optimal Jet Finder', Deterministic Annealing

  Fit jet axes (and #) so as to minimise a weight function

  [forms of 'k-means' clustering]

- Jet energy flow project
- ...

Outline

As LHC startup approaches it's important for the choice of jet algorithm to be well-motivated.

This talk

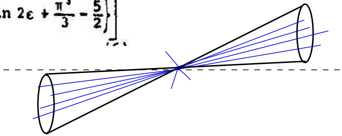
- Overview of iterative cone algorithms (& what's wrong with them)
- Clustering algorithms
  - How they work
  - Where they've been criticised (speed, underlying-event (UE) sensitivity)
  - How to solve the speed problem
  - Work in progress on understanding and reducing sensitivity to UE.

Cone Origins

First 'cone algorithm' dates back to Sterman and Weinberg (1977) — the original infrared-safe cross section:

To study jets, we consider the partial cross section $\sigma(E,\theta,\Omega,\epsilon,\delta)$ for e^+e^- hadron production events, in which all but a fraction $\epsilon \ll 1$ of the total e^+e^- energy E is emitted within some pair of oppositely directed cones of half-angle $\delta \ll 1$, lying within two fixed cones of solid angle Ω (with $\pi\delta^2 \ll \Omega \ll 1$) at an angle θ to the e^+e^- beam line. We expect this to be measur-

$$\sigma(E,\theta,\Omega,\epsilon,\delta) = (d\sigma/d\Omega)_0\,\Omega\left[1 - (g_E^2/3\pi^2)\left\{3\ln\delta + 4\ln\delta\,\ln 2\epsilon + \frac{\pi^2}{3} - \frac{5}{2}\right\}\right]$$



Modern cones address various issues

Where do you put the cones?

- Place a cone at some trial location
- Sum four-momenta of particles in cone – find corresponding axis
- Use that axis as a new trial location, and *iterate*
- Stop when you reach a stable axis [or when you get bored]

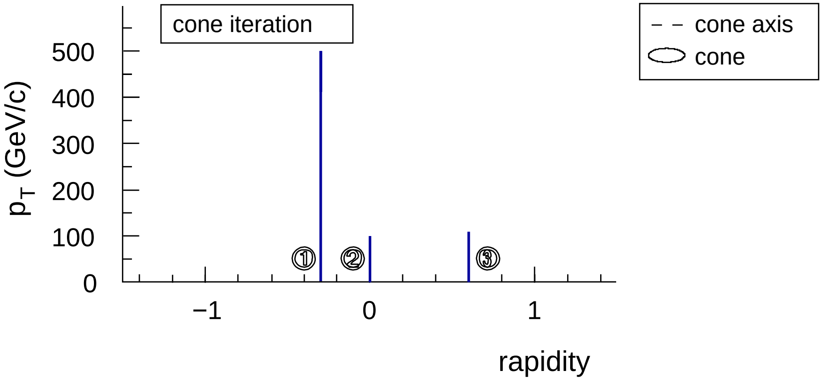
What are the initial trial locations?

- 'Seedless' — i.e. everywhere But too slow on computer
- Use locations with energy flow above some threshold as *seeds*

  Issue: is seed threshold = parton energy, hadron energy (collinear unsafe)?

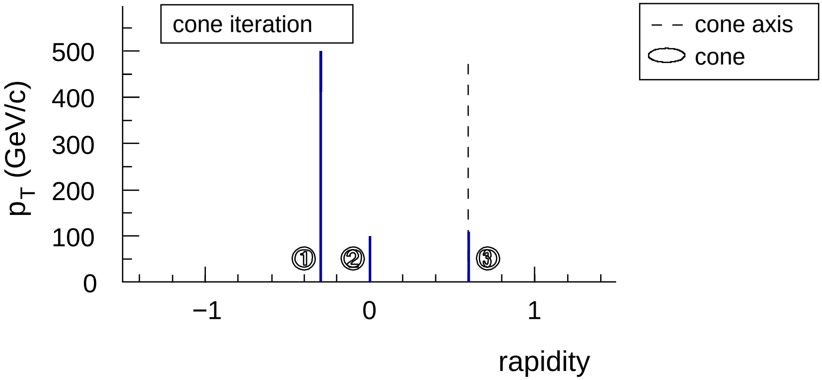
  Or calorimeter tower energy (experiment and η-dependent)?

Iteration example



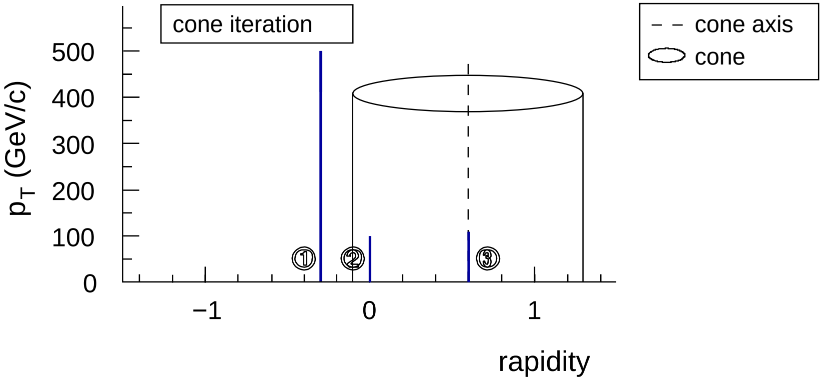
[These and related figures adapted/copied from Markus Wobisch]

Iteration example



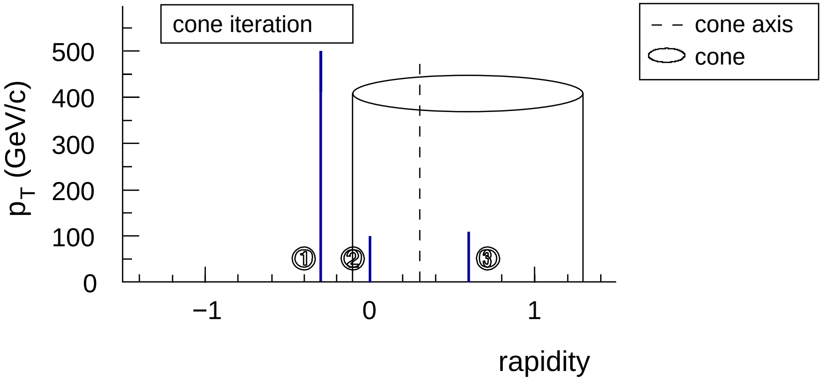
[These and related figures adapted/copied from Markus Wobisch]

Iteration example



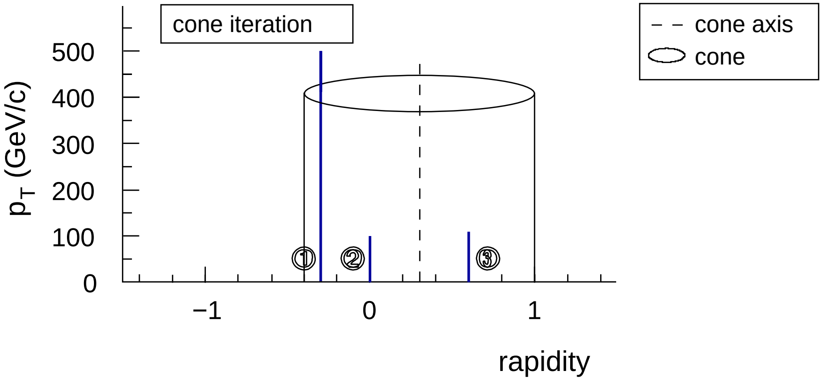
[These and related figures adapted/copied from Markus Wobisch]

Iteration example



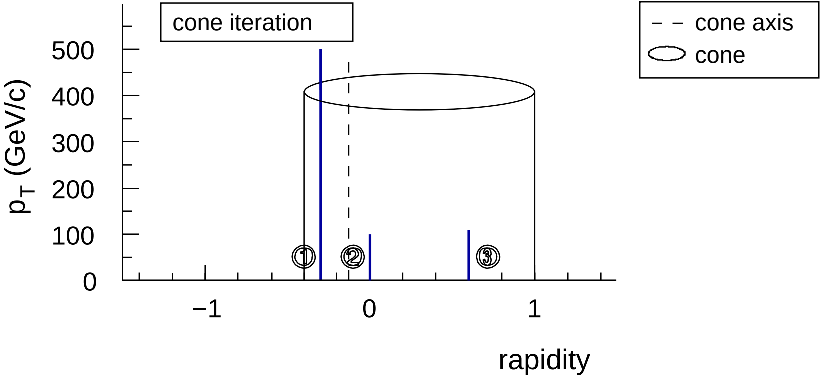
[These and related figures adapted/copied from Markus Wobisch]

Iteration example



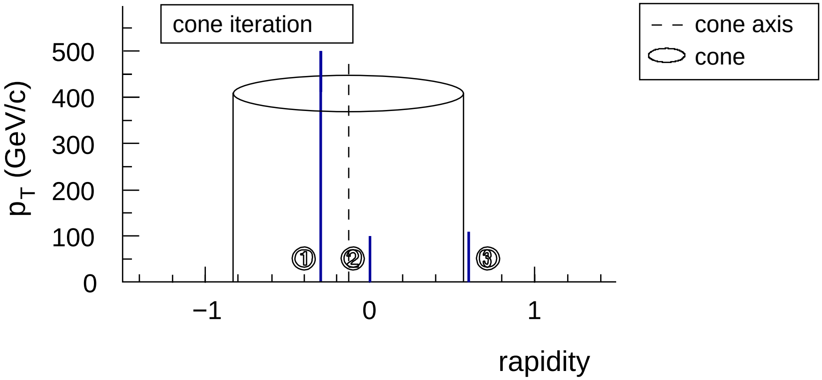
[These and related figures adapted/copied from Markus Wobisch]

Iteration example



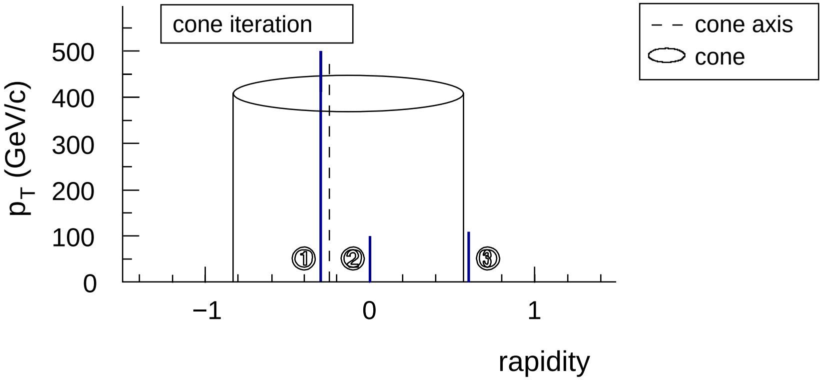
[These and related figures adapted/copied from Markus Wobisch]

Iteration example



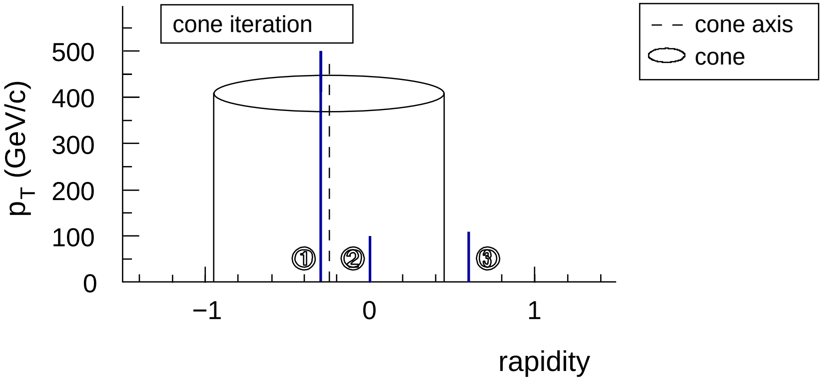
[These and related figures adapted/copied from Markus Wobisch]

Iteration example



[These and related figures adapted/copied from Markus Wobisch]

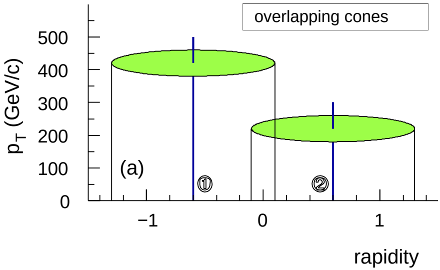
Iteration example



[These and related figures adapted/copied from Markus Wobisch]

Overlapping Jets

Jets can overlap



They are either *split* if the overlapping energy is

$$E_{\text{overlap}} < f_{\text{overlap}} \, E_{\text{softer-jet}}$$

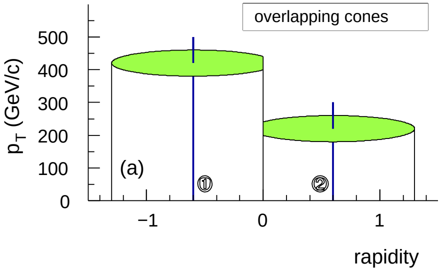
otherwise they are *merged*.

NB: f_{overlap} is parameter of cone-algo

NB: when many jets overlap, procedure for merging/splitting must be specified (e.g. wrt order in which jets are treated).

Overlapping Jets

Jets can overlap



They are either *split* if the overlapping energy is

$$E_{\rm overlap} < f_{\rm overlap}\, E_{\rm softer\text{-}jet}$$

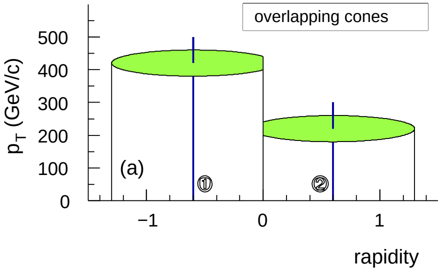
otherwise they are *merged*.

NB: $f_{\rm overlap}$ is parameter of cone-algo

NB: when many jets overlap, procedure for merging/splitting must be specified (e.g. wrt order in which jets are treated).

Overlapping Jets

Jets can overlap



They are either *split* if the overlapping energy is

$$E_{\text{overlap}} < f_{\text{overlap}}\, E_{\text{softer-jet}}$$

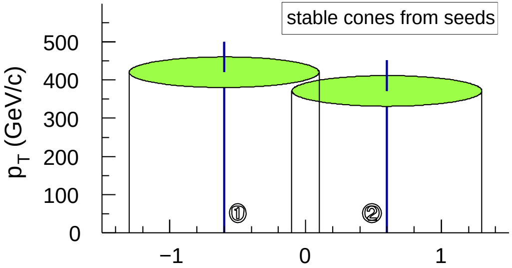
otherwise they are *merged*.

NB: f_{overlap} is parameter of cone-algo

NB: when many jets overlap, procedure for merging/splitting must be specified (e.g. wrt order in which jets are treated).

Midpoints

Use of seeds is *dangerous*



Extra soft particle adds new seed ⟶ changes final jet configuration.

This is **IR unsafe**.

Kilgore & Giele '97

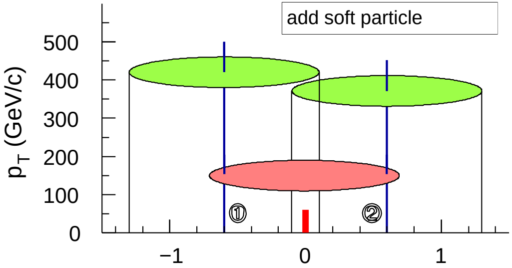
Solution: add extra seeds at midpoints of all pairs, triplets, . . . of stable cones.

Seymour '97 (?)

NB: only in past 1-2 years has this fix appeared in CDF and D0 analyses. . .

Midpoints

Use of seeds is *dangerous*



Extra soft particle adds new seed → changes final jet configuration.

This is **IR unsafe**.

Kilgore & Giele '97

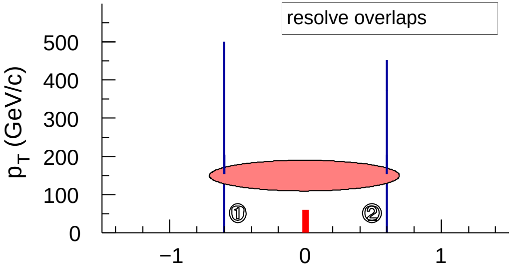
Solution: add extra seeds at midpoints of all pairs, triplets, . . . of stable cones.

Seymour '97 (?)

NB: only in past 1-2 years has this fix appeared in CDF and D0 analyses. . .

Midpoints

Use of seeds is *dangerous*



Extra soft particle adds new seed → changes final jet configuration.

This is **IR unsafe**.

Kilgore & Giele '97

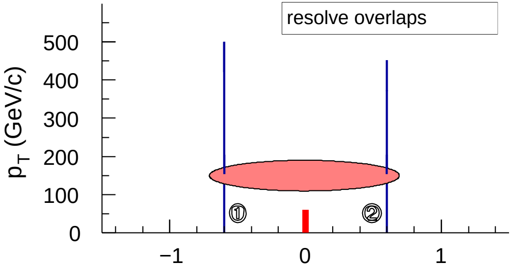
Solution: add extra seeds at midpoints of all pairs, triplets, . . . of stable cones.

Seymour '97 (?)

NB: only in past 1-2 years has this fix appeared in CDF and D0 analyses. . .

Midpoints

Use of seeds is *dangerous*



Extra soft particle adds new seed ⟶ changes final jet configuration.

This is **IR unsafe**.

Kilgore & Giele '97

Solution: add extra seeds at midpoints of all pairs, triplets, ...of stable cones.

Seymour '97 (?)

NB: only in past 1-2 years has this fix appeared in CDF and D0 analyses...

ILCA

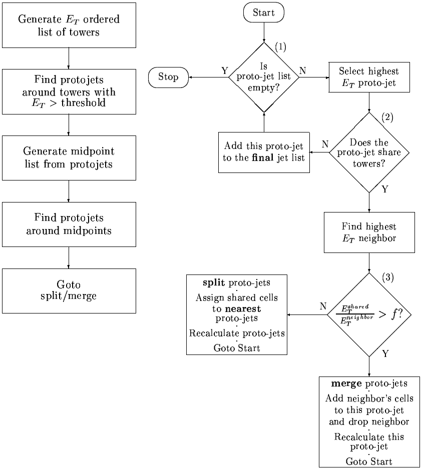
All of these considerations led recommendation of the *Improved Legacy Cone Algorithm* (ILCA), a.k.a. *Midpoint* algorithm.

hep-ex/0005012

Quite complex and has several parameters:

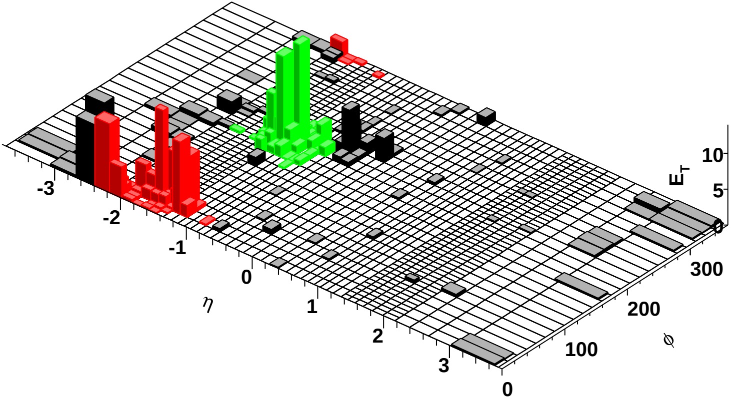
cone radius (R)
seed threshold (E_0)
f_{overlap}

Only one of these is remotely physical: R.



2/3 of ILCA flowchart

ILCA has "Dark Towers"

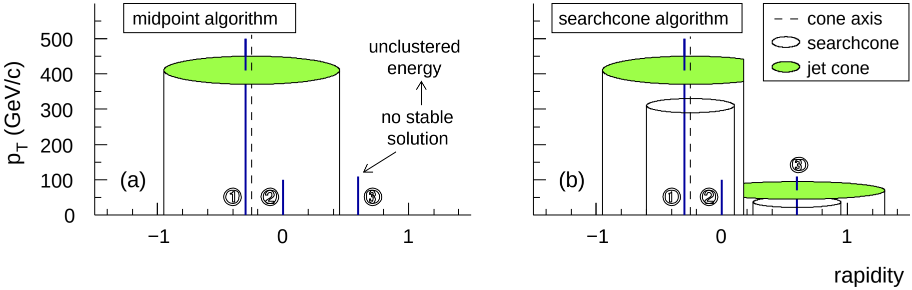


Considerable energy can be left out of jets ≡ **Dark Towers**

S. Ellis, Huston & Tönnesmann '01

Search Cone

Dark towers are consequence of particles that are never in stable cones:

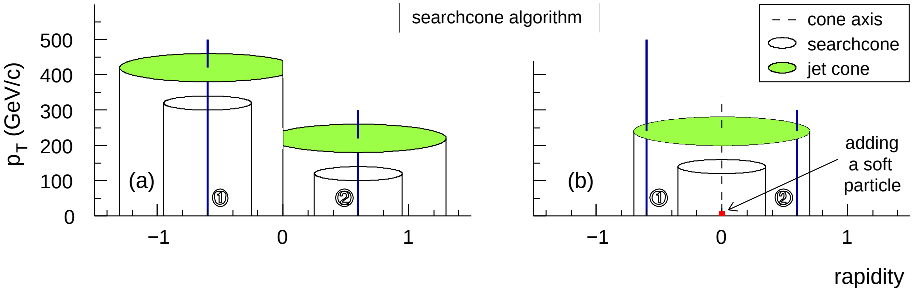


Ellis, Huston and Tönnesmann suggest *iterating a smaller 'search-cone'* and then drawing final cone around it.

Searchcone adopted by CDF (to confuse issue they call it 'midpoint'...).

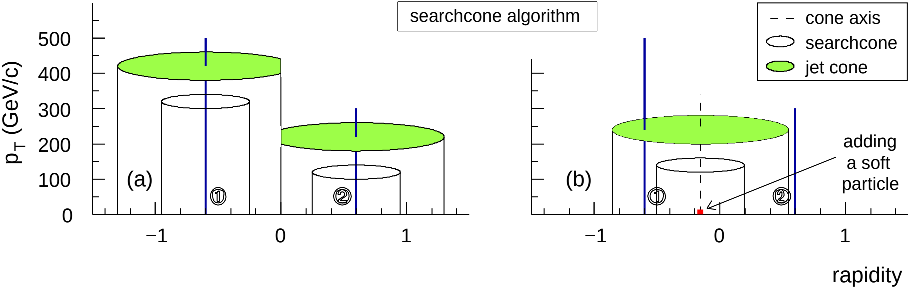
hep-ex/0505013, hep-ex/0512020

Search Cone is IR unsafe



Whether you see 1 or 2 jets depends on presence and position of a soft gluon — this is *IR unsafe (and unphysical)*. Wobisch, '06

Search Cone is IR unsafe



Whether you see 1 or 2 jets depends on presence and position of a soft gluon — this is *IR unsafe (and unphysical)*. Wobisch, '06

Cone summary

- Cone algorithms are complicated beasts.
- So much so, it's often not clear *which* cone algorithm is being used!
- They often behave in unforeseen ways.
- *Patching* them makes them more complex and error-prone.

Didn't even mention the hacks people put into cone theory calculations to 'tune' them to hadron level: (cf. R_{sep}, which breaks the NLO jet X-section).

LHC experiments should be wary of cone algorithms

Sequential Recombination Algorithms

Best known is **k_t algorithm:**

1. Calculate (or update) distances between all particles i and j, and between i and beam:

$$d_{ij} = \min(k_{ti}^2, k_{tj}^2)\frac{\Delta R_{ij}^2}{R^2}, \qquad d_{iB} = k_{ti}^2\,, \qquad \Delta R_{ij}^2 = \Delta y_{ij}^2 + \Delta\phi_{ij}^2$$

2. Find smallest of d_{ij} and d_{iB}
   - If d_{ij} is smallest, recombine i and j (add result to particle list, remove i, j)
   - if d_{iB} is smallest call i a jet (remove it from list of particles)

3. If any particles are left, repeat from step 1.

Catani, Dokshitzer, Olsson, Turnock, Seymour & Webber '91–93
S. Ellis & Soper, '93

One parameter: R (like cone radius), whose natural value is 1

Optional second parameter: stopping scale d_{cut} 'exclusive' k_t algorithm

Sequential Recombination Algorithms

Best known is **$\mathbf{k_t}$ algorithm:**

1. Calculate (or update) distances between all particles i and j, and between i and beam:

$$d_{ij} = \min(k_{ti}^2, k_{tj}^2)\frac{\Delta R_{ij}^2}{R^2}, \qquad d_{iB} = k_{ti}^2\,, \qquad \Delta R_{ij}^2 = \Delta y_{ij}^2 + \Delta\phi_{ij}^2$$

2. Find smallest of d_{ij} and d_{iB}
   - If d_{ij} is smallest, recombine i and j (add result to particle list, remove i, j)
   - if d_{iB} is smallest call i a jet (remove it from list of particles)

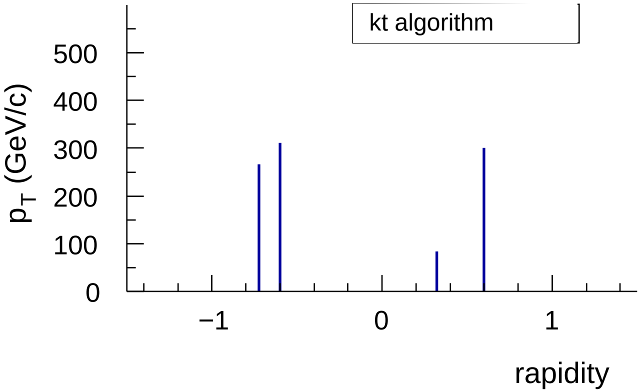
3. If any particles are left, repeat from step 1.

Catani, Dokshitzer, Olsson, Turnock, Seymour & Webber '91–93
S. Ellis & Soper, '93

One parameter: R (like cone radius), whose natural value is 1

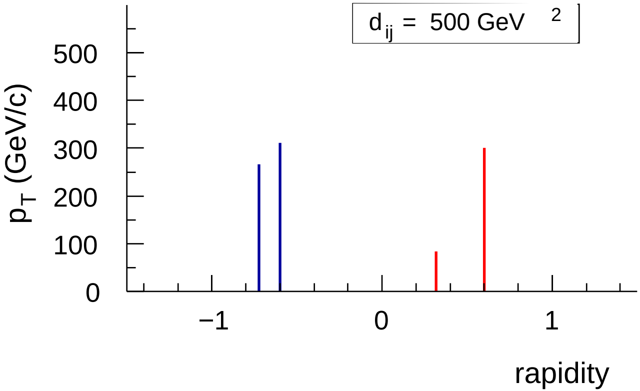
Optional second parameter: stopping scale d_{cut} 'exclusive' k_t algorithm

k_t algorithm in action ($R = 1$)



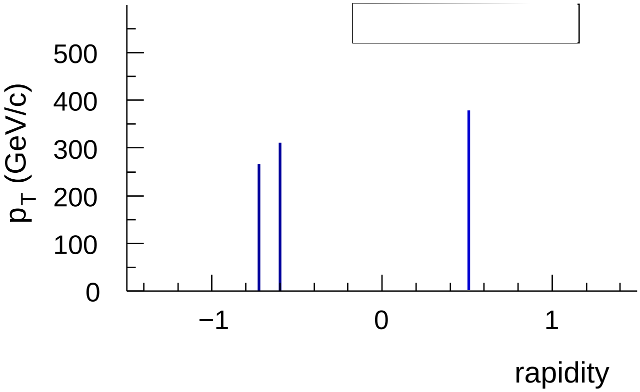
$$d_{ij} = \min(k_{ti}^2, k_{tj}^2)\Delta R_{ij}^2, \qquad d_{iB} = k_{ti}^2$$

k_t algorithm in action ($R = 1$)



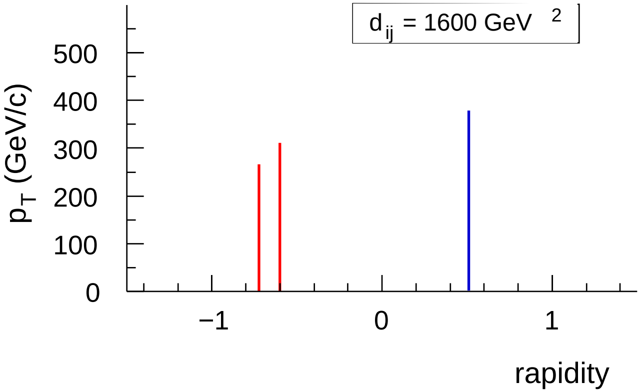
$$d_{ij} = \min(k_{ti}^2, k_{tj}^2)\Delta R_{ij}^2, \qquad d_{iB} = k_{ti}^2$$

k_t algorithm in action ($R = 1$)



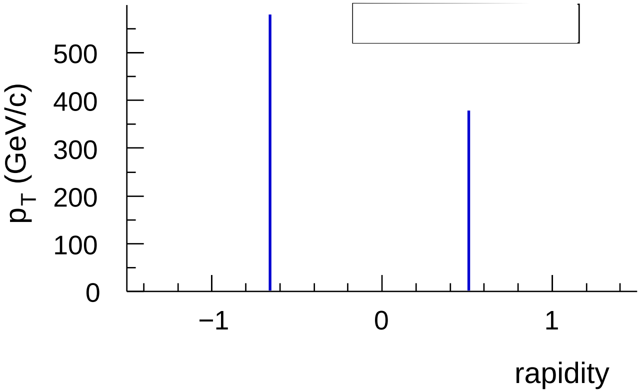
$$d_{ij} = \min(k_{ti}^2, k_{tj}^2)\Delta R_{ij}^2, \qquad d_{iB} = k_{ti}^2$$

k_t algorithm in action ($R = 1$)



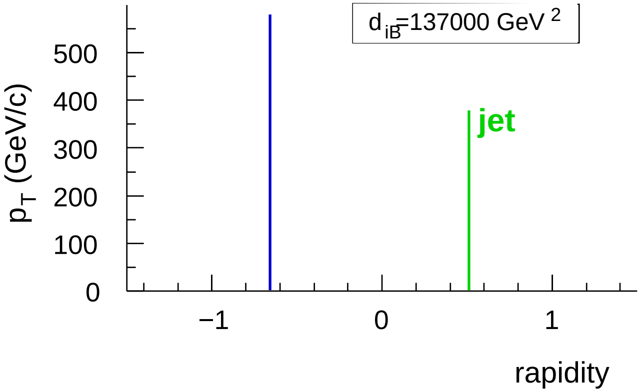
$$d_{ij} = \min(k_{ti}^2, k_{tj}^2)\Delta R_{ij}^2, \qquad d_{iB} = k_{ti}^2$$

k_t algorithm in action ($R = 1$)



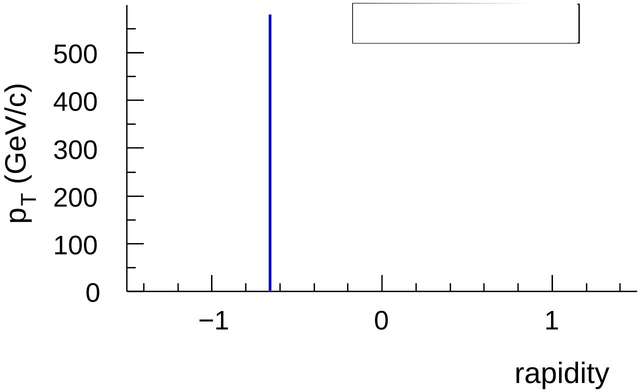
$$d_{ij} = \min(k_{ti}^2, k_{tj}^2)\Delta R_{ij}^2, \qquad d_{iB} = k_{ti}^2$$

k_t algorithm in action ($R = 1$)



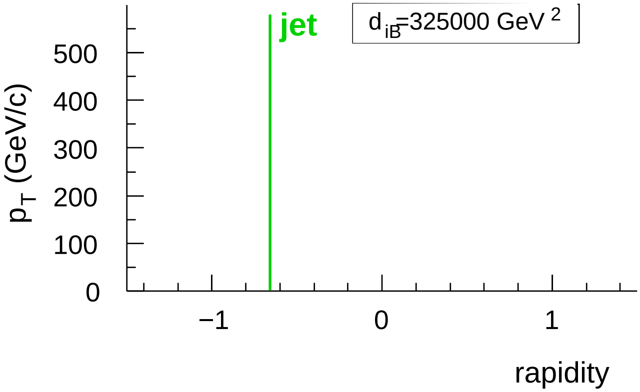
$$d_{ij} = \min(k_{ti}^2, k_{tj}^2)\Delta R_{ij}^2, \qquad d_{iB} = k_{ti}^2$$

k_t algorithm in action ($R = 1$)



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

k_t algorithm in action ($R = 1$)



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

Why k_t?

k_t distance measures

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

are closely related to structure of divergences for QCD emissions

$$[dk_j]|M^2_{g\to g_i g_j}(k_j)| \sim \frac{\alpha_s C_A}{2\pi}\frac{dk_{tj}}{\min(k_{ti}, k_{tj})}\frac{d\Delta R_{ij}}{\Delta R_{ij}}\,, \qquad (k_{tj} \ll k_{ti}\,,\ \Delta R_{ij} \ll 1)$$

and

$$[dk_i]|M^2_{Beam\to Beam+g_i}(k_i)| \sim \frac{\alpha_s C_A}{\pi}\frac{dk_{ti}}{k_{ti}}\,d\eta_i\,, \qquad (k_{ti}^2 \ll \{\hat{s}, \hat{t}, \hat{u}\})$$

k_t algorithm attempts approximate inversion of branching process

Why k_t?

k_t distance measures

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

are closely related to structure of divergences for QCD emissions

$$[dk_j]|M^2_{g\to g_i g_j}(k_j)| \sim \frac{\alpha_s C_A}{2\pi}\frac{dk_{tj}}{\min(k_{ti}, k_{tj})}\frac{d\Delta R_{ij}}{\Delta R_{ij}}\,, \qquad (k_{tj} \ll k_{ti}\,,\;\; \Delta R_{ij} \ll 1)$$

and

$$[dk_i]|M^2_{Beam\to Beam+g_i}(k_i)| \sim \frac{\alpha_s C_A}{\pi}\frac{dk_{ti}}{k_{ti}}\,d\eta_i\,, \qquad (k_{ti}^2 \ll \{\hat{s}, \hat{t}, \hat{u}\})$$

k_t algorithm attempts approximate inversion of branching process

k_t v. cone

k_t algorithm seems better than cone

- it's simpler, safer and better-defined
- exclusive variant is more flexible (allows cuts on momentum scales)
- less sensitive to hadronization
- In MC studies k_t alg. is systematically as good as, or better than cone algorithms for typical reconstruction tasks

Seymour '94

Butterworth, Cox & Forshaw '02

Benedetti et al (Les Houches) '06

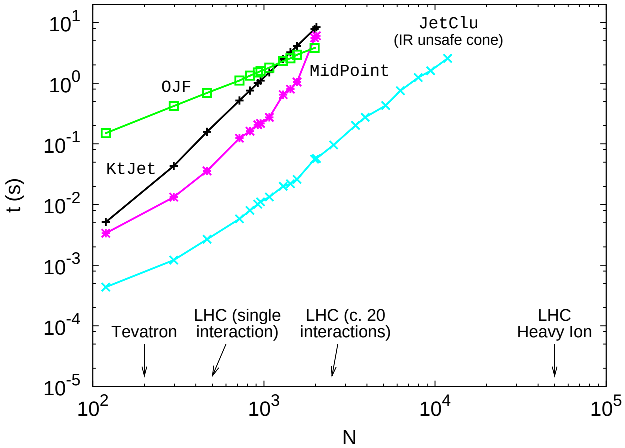
But seldom used at Tevatron. **Why?**

1. Because it's slow?
2. Because it includes more underlying event?
3. Because it's harder to understand detector effects?

But all LEP and HERA experiments managed fine

And as of '05, CDF too

Time to cluster N particles



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

a Pb-Pb event takes 1 day!

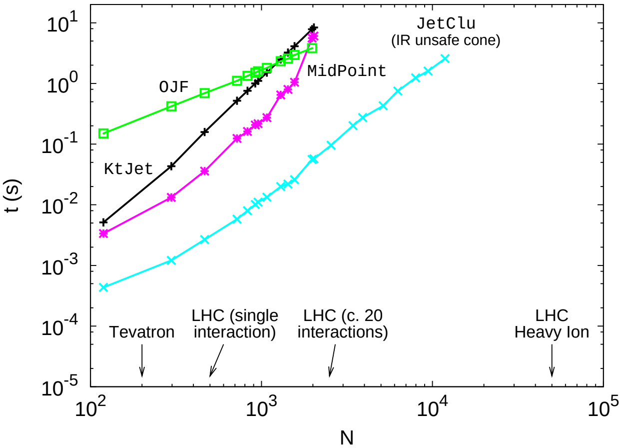
IR-unsafe cone (JetClu) is *much faster*.

IR-safe cone (Midpoint) is as bad as k_t

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

NB: want to rerun jet-alg. with a range of parameter choices
+ want to run on multiple MC samples of similar size

Time to cluster N particles



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

a Pb-Pb event takes 1 day!

IR-unsafe cone (JetClu) is *much faster*.

IR-safe cone (Midpoint) is as bad as k_t

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

NB: want to rerun jet-alg. with a range of parameter choices

+ want to run on multiple MC samples of similar size

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

k_t is a form of Hierarchical Clustering

Fast Hierarchical Clustering and Other Applications of Dynamic Closest Pairs

David Eppstein
UC Irvine

We develop data structures for dynamic closest pair problems with arbitrary distance functions, that do not necessarily come from any geometric structure on the objects. Based on a technique previously used by the author for Euclidean closest pairs, we show how to insert and delete objects from an n-object set, maintaining the closest pair, in $O(n \log^2 n)$ time per update and $O(n)$ space. With quadratic space, we can instead use a quadtree-like structure to achieve an optimal time bound, $O(n)$ per update. We apply these data structures to hierarchical clustering, greedy matching, and TSP heuristics, and discuss other potential applications in machine learning, Gröbner bases, and local improvement algorithms for partition and placement problems. Experiments show our new methods to be faster in practice than previously used heuristics.

Categories and Subject Descriptors: F.2.2 [**Analysis of Algorithms**]: Nonnumeric Algorithms

General Terms: Closest Pair, Agglomerative Clustering

Additional Key Words and Phrases: TSP, matching, conga line data structure, quadtree, nearest neighbor heuristic

1. INTRODUCTION

Hierarchical clustering has long been a mainstay of statistical analysis, and clustering based methods have attracted attention in other fields: computational biology (reconstruction of evolutionary trees; tree-based multiple sequence alignment), scientific simulation (n-body problems), theoretical computer science (network design and nearest neighbor searching) and of course the web (hierarchical indices such as Yahoo). Many clustering methods have been devised and used in these applications, but less effort has gone into algorithmic speedups of these methods.

In this paper we identify and demonstrate speedups for a key subroutine used in several clustering algorithms, that of maintaining closest pairs in a dynamic set of objects. We also describe several other applications or potential applications of the

k_t alg. is so good it's used throughout science!

NB HEP is not only field to use brute-force. . .

For general distance measures problem reduces to $\sim N^2$ (factor ~ 20 for $N = 1000$).

Eppstein '99
+ Cardinal '03

k_t is a form of Hierarchical Clustering

Fast Hierarchical Clustering and Other Applications of Dynamic Closest Pairs

David Eppstein
UC Irvine

We develop data structures for dynamic closest pair problems with arbitrary distance functions, that do not necessarily come from any geometric structure on the objects. Based on a technique previously used by the author for Euclidean closest pairs, we show how to insert and delete objects from an n-object set, maintaining the closest pair, in $O(n \log^2 n)$ time per update and $O(n)$ space. With quadratic space, we can instead use a quadtree-like structure to achieve an optimal time bound, $O(n)$ per update. We apply these data structures to hierarchical clustering, greedy matching, and TSP heuristics, and discuss other potential applications in machine learning, Gröbner bases, and local improvement algorithms for partition and placement problems. Experiments show our new methods to be faster in practice than previously used heuristics.

Categories and Subject Descriptors: F.2.2 [**Analysis of Algorithms**]: Nonnumeric Algorithms

General Terms: Closest Pair, Agglomerative Clustering

Additional Key Words and Phrases: TSP, matching, conga line data structure, quadtree, nearest neighbor heuristic

Of these naive methods, brute force recomputation may be most commonly used, due to its low space requirements and ease of implementation. Three hierarchical clustering codes we examined, Zupan's [Zupan 1982], CLUSTAL W [Thompson et al. 1994], and PHYLIP [Felsenstein 1995] use brute force. (Indeed, they do not even save space by doing so, since they all store the distance matrix.) Pazzani's learning code [Pazzani 1997] also uses brute force (M. Pazzani, personal communication), as does *Mathematica*'s Gröbner basis code (D. Lichtblau, personal communication).

k_t alg. is so good it's used throughout science!

NB HEP is not only field to use brute-force. . .

For general distance measures problem reduces to $\sim N^2$ (factor ~ 20 for $N = 1000$).

Eppstein '99
+ Cardinal '03

Can we 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 we 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 we 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:*

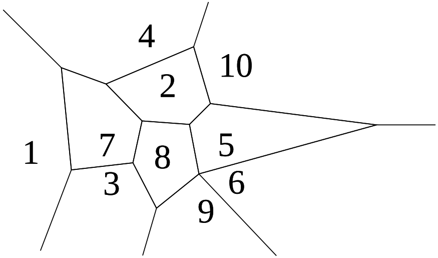
- Consider smallest $d_{ij} = \min(k_{ti}^2, k_{tj}^2) R_{ij}^2$
- Suppose $k_{ti} < k_{tj}$
- Then: $R_{ij} <= R_{i\ell}$ for any $\ell \neq j$. [If $\exists\ \ell$ s.t. $R_{i\ell} < R_{ij}$ then $d_{i\ell} < d_{ij}$]

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 1,4,2,3 (it turns out to be 3)

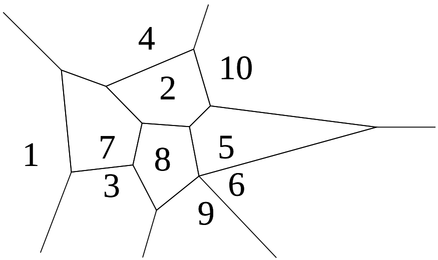
Construction of Voronoi diagram for N points: $N \ln N$ time   Fortune '88

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 1,4,2,3 (it turns out to be 3)

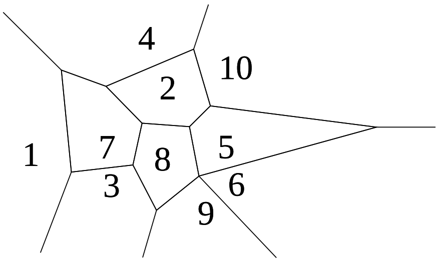
Construction of Voronoi diagram for N points: $N \ln N$ time — Fortune '88

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 1,4,2,3 (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 $\mathcal{O}(\mathbf{N} \ln \mathbf{N})$

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

Repeat following steps **N** times:

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

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

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

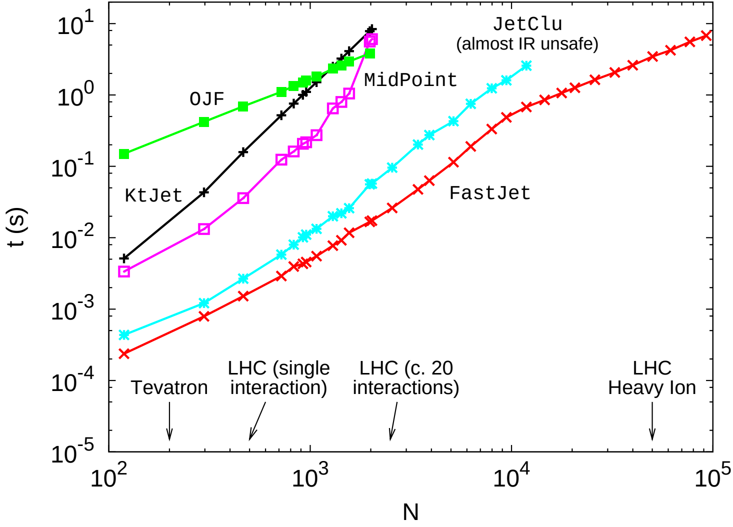
Repeat following steps **N** times:

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

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

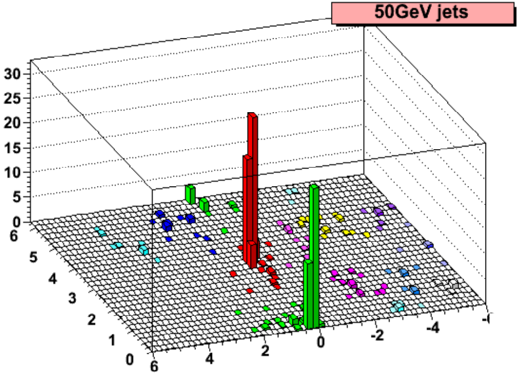
Cacciari & GPS, hep-ph/0512210
http://www.lpthe.jussieu.fr/~salam/fastjet/
Results identical to standard N^3 implementations

FastJet performance



NB: for $N < 10^4$, FastJet switches to a related geometrical N^2 alg.

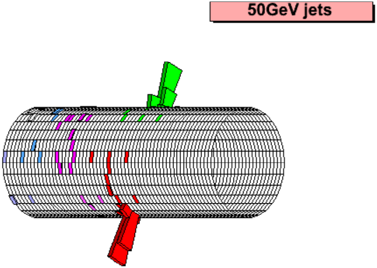
What is speed good for?



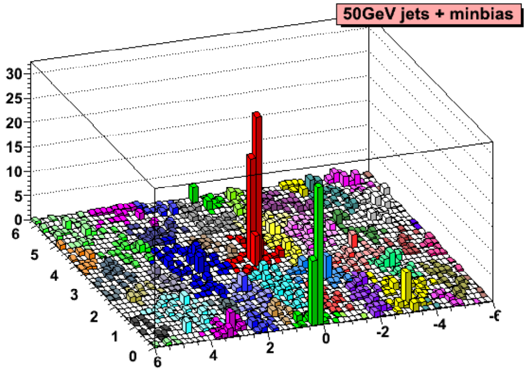
'Standard hard' event
Two well isolated jets

$\sim$ 200 particles

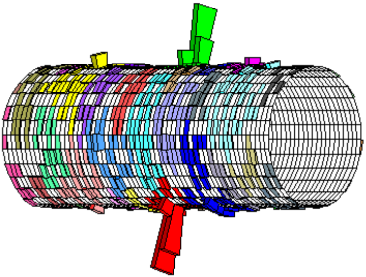
Easy even with old methods



What is speed good for?



Add 10 min-bias events (moderately high lumi)

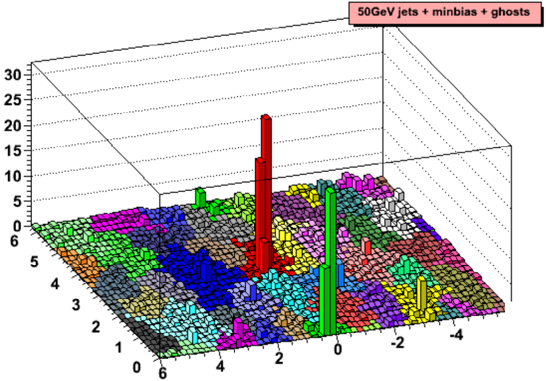


$\sim$ 2000 particles

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

20ms with `FastJet`.

What is speed good for?



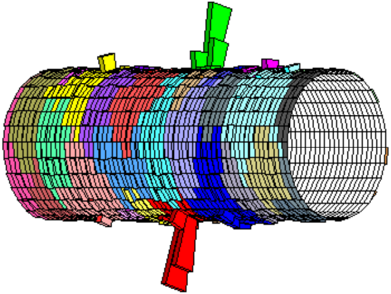
Add dense coverage of infinitely soft *"ghosts"*

See how many end up in jet to measure jet area

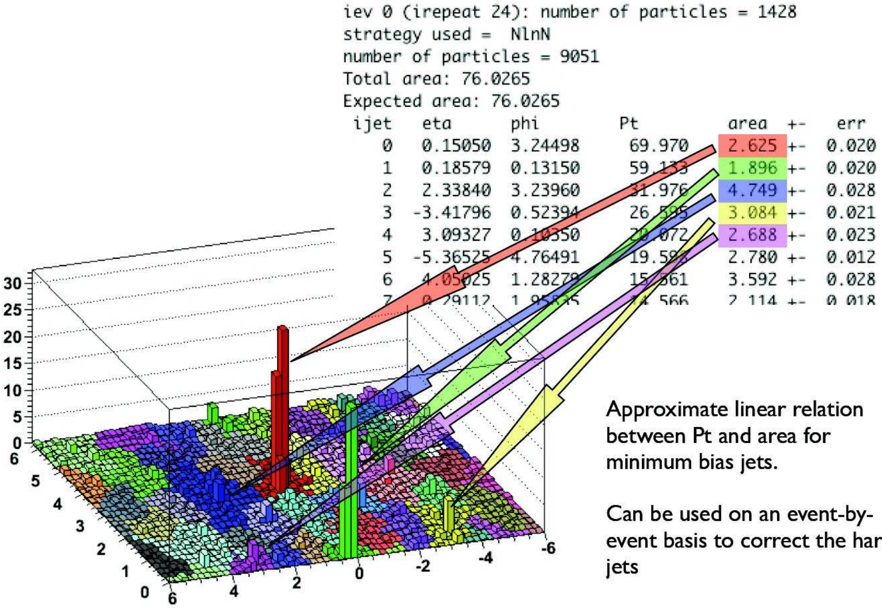
~ 10000 particles

Clustering takes ~ 20 minutes with old methods.

0.6s with `FastJet`.



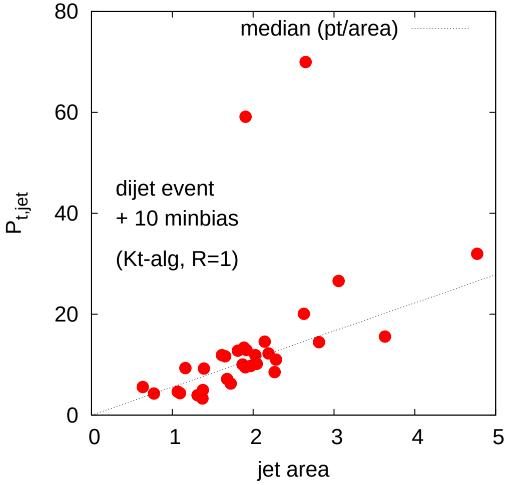
Jet areas



Approximate linear relation between Pt and area for minimum bias jets.

Can be used on an event-by-event basis to correct the hard jets

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

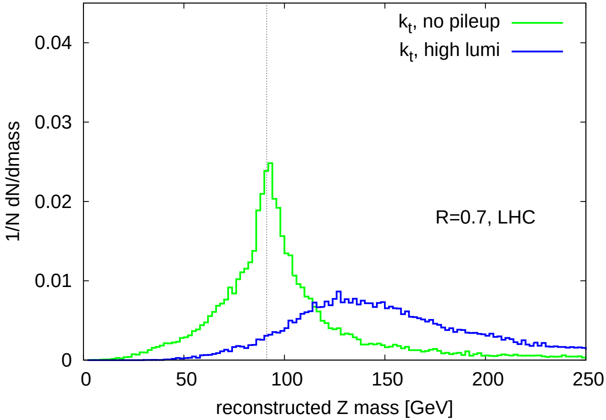
Complicates corrections: min-bias subtraction is different for each jet.

Cone supposedly simpler
Area $= \pi R^2$?

Z mass: k_t v. cone (uncorrected)

Try reconstructing M_Z from $Z \to 2$ jets [Use inv. mass of two hardest jets]

On same events, compare uncorrected k_t v. ILCA (midpoint) cone



k_t allegedly more sensitive to min-bias.
Is this true?

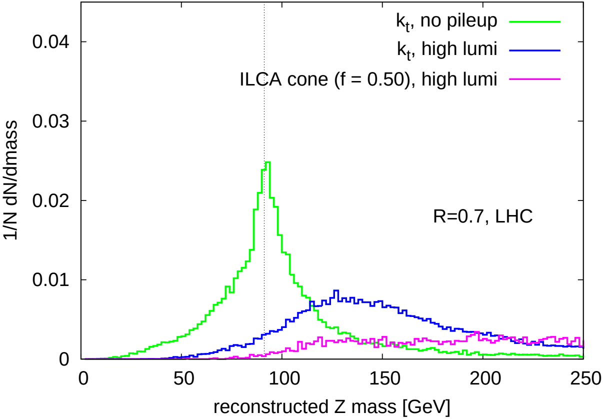
ILCA with standard parameters ($f_{overlap} = 0.5$) fares *very poorly*

ILCA with modified params. is no better than k_t.

Z mass: k_t v. cone (uncorrected)

Try reconstructing M_Z from $Z \to 2$ jets [Use inv. mass of two hardest jets]

On same events, compare uncorrected k_t v. ILCA (midpoint) cone



k_t allegedly more sensitive to min-bias.
Is this true?

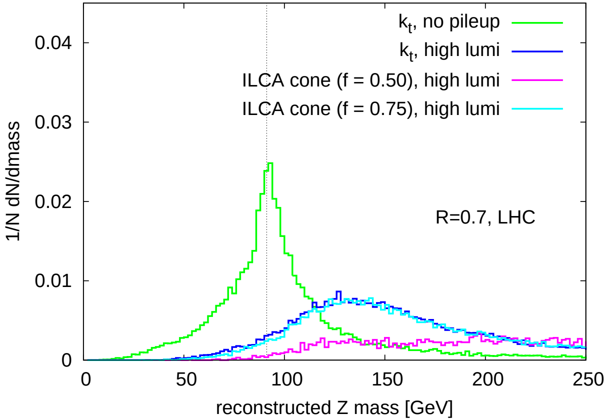
ILCA with standard parameters ($f_{overlap} = 0.5$) fares *very poorly*

ILCA with modified params. is no better than k_t.

Z mass: k_t v. cone (uncorrected)

Try reconstructing M_Z from $Z \to 2$ jets [Use inv. mass of two hardest jets]

On same events, compare uncorrected k_t v. ILCA (midpoint) cone

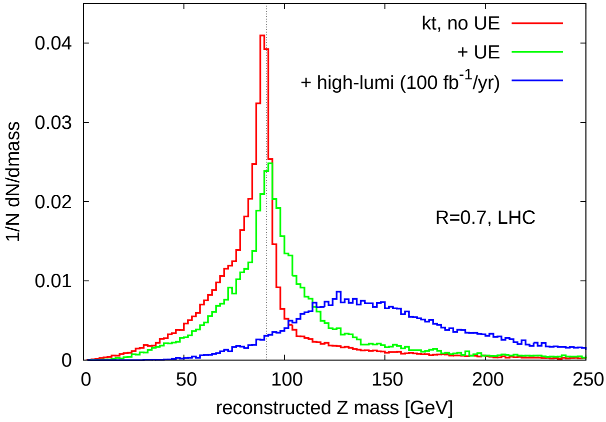


k_t allegedly more sensitive to min-bias.
Is this true?

ILCA with standard parameters ($f_{overlap} = 0.5$) fares *very poorly*

ILCA with modified params. is no better than k_t.

Use jet areas to correct jet kinematics



Correction procedure:

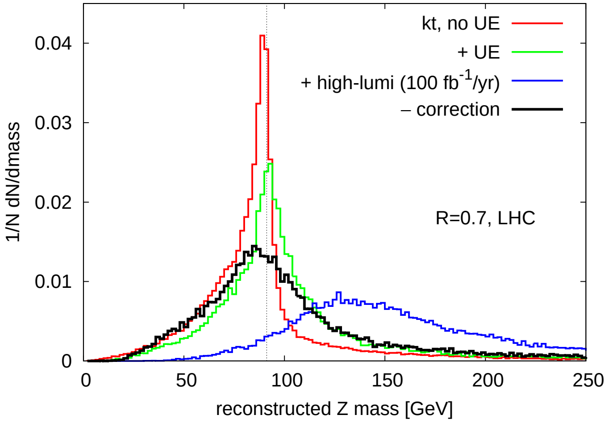
Measure area A of each jet

Find median $p_t/A = Q_0$

Subtract $\Delta p_t = A \times Q_0$ from each jet.

NB: cone much harder to correct this way — too slow to add 10^4 ghosts

Use jet areas to correct jet kinematics



Correction procedure:

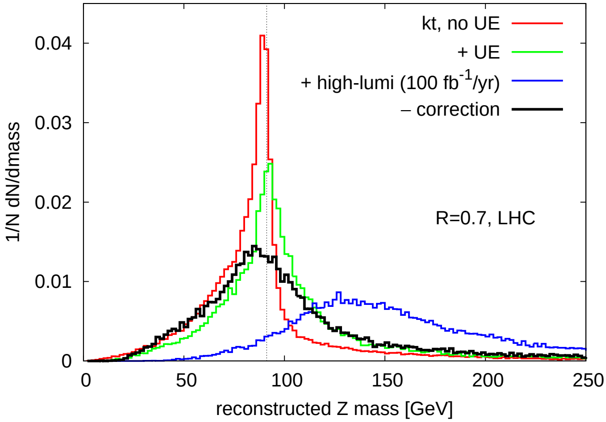
Measure area A of each jet

Find median $p_t/A = Q_0$

Subtract $\Delta p_t = A \times Q_0$ from each jet.

NB: cone much harder to correct this way — too slow to add 10^4 ghosts

Use jet areas to correct jet kinematics



Correction procedure:

Measure area A of each jet

Find median $p_t/A = Q_0$

Subtract $\Delta p_t = A \times Q_0$ from each jet.

NB: cone much harder to correct this way — too slow to add 10^4 ghosts

Analytical results for mean areas

Suppose incoming partons (colour charge C_i) and outgoing jets (col. charge $= C_o$) are not colour connected.

Mean outgoing jet area $\langle A \rangle$ depends on jet P_t as follows:

$$\langle A \rangle = R^2 \left(\pi + (a_0 C_o + a_2 C_i R^2) \frac{\alpha_s}{\pi} \ln \frac{P_t^2}{Q_0^2} + \mathcal{O}\left(\alpha_s, \alpha_s^2 L^2\right) \right)$$

GPS & Cacciari, prelim.

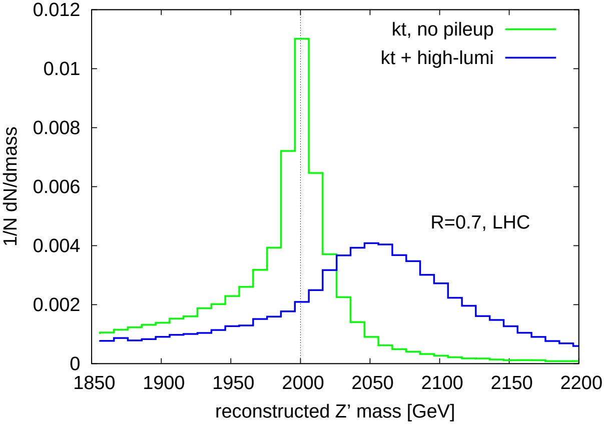
	a_0	a_2	comment
k_t	$+1.771$	$+0.325$	significant, positive
ILCA (cone)	-0.200	-0.325	small, negative
Cam / Aachen	$+0.249$	0	small, positive

For $Q_0 \sim 10$ GeV, $P_t \sim 100 - 1000$ GeV, $\frac{\alpha_s}{\pi} \ln P_t^2/Q_0^2 \sim 0.2 - 0.4$

Cambridge / Aachen algorithm? Like k_t with but $d_{ij} = R_{ij}^2/R^2$ and $d_{iB} = 1$.

Dokshitzer, Leder, Moretti & Webber '97; Wobisch '00

Reconstruct Z' mass [2 TeV]

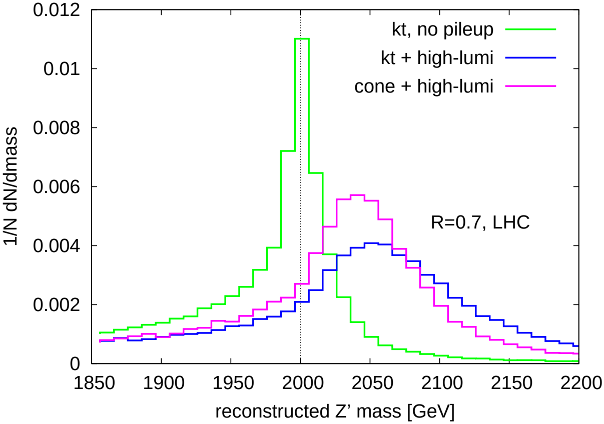


Uncorrected cone better than k_t.

Cam is intermediate ($\langle A_{cam} \rangle \simeq \langle A_{cone} \rangle$, but fluctuations larger)

Corrected Cam (and k_t) is best.

Reconstruct Z' mass [2 TeV]

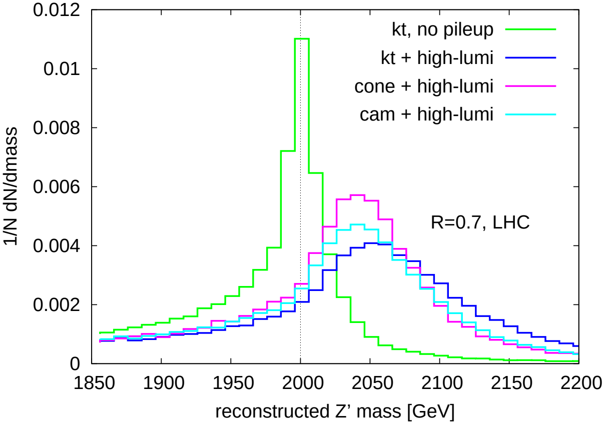


Uncorrected cone better than k_t.

Cam is intermediate ($\langle A_{cam} \rangle \simeq \langle A_{cone} \rangle$, but fluctuations larger)

Corrected Cam (and k_t) is best.

Reconstruct Z' mass [2 TeV]

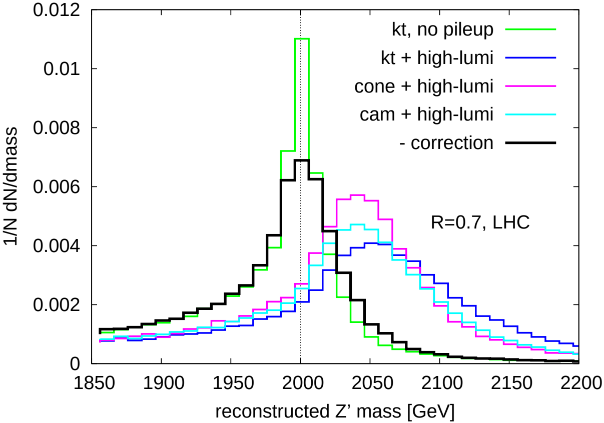


Uncorrected cone better than k_t.

Cam is intermediate ($\langle A_{cam} \rangle \simeq \langle A_{cone} \rangle$, but fluctuations larger)

Corrected Cam (and k_t) is best.

Reconstruct Z' mass [2 TeV]



Uncorrected cone better than k_t.

Cam is intermediate ($\langle A_{cam} \rangle \simeq \langle A_{cone} \rangle$, but fluctuations larger)

Corrected Cam (and k_t) is best.

k_t summary

- k_t alg. is **fast** (faster than IR unsafe JetClu) — key observation is geometrical reformulation

  Get code from `http://www.lpthe.jussieu.fr/~salam/fastjet`

- Jet areas ($\rightarrow$ min. bias. contributions) do fluctuate

  Some aspects of areas amenable to analytical calculations

- But areas can (should) be **measured** and **used for correction** on jet-by-jet basis.

  Preliminary studies very promising

- k_t is part of a class of algorithms — other example deserving more attention is *Cambridge/Aachen* alg.

  It too can be made fast