Dispelling the $N^3$ Myth for the $k_t$ Jet-Finder

Matteo Cacciari
LPTHE - Paris 6

with Gavin Salam, hep-ph/0512210

Outline

- Why Jets
- Why $k_t$ clustering
- Why FastJet

'Suz it's better
'Suz it's way faster

Short version of this talk
Why Jets

Well, because.... jets happen!

A high energy event will in general show **collimated bunches of hadrons**

Starting from this observation and this very loose definition we must work to make jets **good proxies** of the underlying partons, **quarks and gluons**

NB. This is not a review talk in jet physics or even in jet-clustering algorithms

Rather, just a shameless **sales pitch** for our **FastJet** code
Why Jets

Jets are as old as the parton model (yes, even older than QCD...):

R.P. Feynman, Photon Hadron Interactions, p. 166 (1972)

The first rigorous definition of an **infrared and collinear safe** jet in QCD is due to Sterman and Weinberg, Phys. Rev. Lett. 39, 1436 (1977):

To study jets, we consider the partial cross section

\[ \sigma(E, \theta, \Omega, \epsilon, \delta) \] 

for e^+e^- hadron production events, in which a fraction \( \epsilon << 1 \) of the total e^+e^- energy E is emitted within some pair of oppositely directed cones of half-angle \( \delta << 1 \), lying within two fixed cones of solid angle \( \Omega \) (with \( \pi \delta^2 << \Omega << 1 \)) at an angle \( \theta \) to the e^+e^- beam line. We expect this to be mea-

\[ \sigma(E, \theta, \Omega, \epsilon, \delta) = \left( \frac{d\sigma}{d\Omega} \right) \Omega \left[ 1 - \left( g_0^2 / 3\pi^2 \right) \left( 3 \ln \delta + 4 \ln \delta \ln 2\epsilon + \frac{\pi^2}{3} - \frac{5}{2} \right) \right]. \]
Two main jet-finder classes: **cone algorithms** and **sequential clustering algorithms**

**Cone-type** algorithms are mainly used at the Tevatron. Extensions of original Sterman-Weinberg idea, i.e. identify energy flow into cones. Detailed definition can be messy. Infrared/collinear safety must be carefully studied.

**Sequential clustering** algorithms are based on pair-wise successive recombinations. Widely used at LEP and HERA. Simple definition, safely infrared and collinear safe.
At face value, both cone and $k_t$ allow for good data/theory comparisons. However, there are a number of reasons why $k_t$ should be preferred.
The definition of a cone algorithm can be extremely complicated.

For instance, take **MidPoint**:

- Begin with 1 GeV seed towers
- Cluster towers with $p_T > 100$ MeV into jet if $\Delta R = \sqrt{\Delta \eta^2 + \Delta \phi^2} < 0.7$
- Start new search cones at midpoint of stable cones
- Merge jets if overlapping energy is $> 0.75$ times the energy of the smaller jet
- Calculate jet quantities from stable cones

*At least four more or less arbitrary parameters*
More troubles:  

**Dark towers**

**Solution (?!)**

- Implement an initial search cone step with the search cone size $= R_{cone}/2$
- Less sensitive to effects of great attractors far away
- After stable cones are formed, expand jet cones to full size and decide whether to split/merge overlapping jets according to the standard criteria

(Cure worse than disease?)

[NB. Fifth parameter...]

Some of the energy is not collected in any jet

[A sixth parameter is also introduced (by CDF only!) to tweak the NLO calculation when running the algorithm on theoretical results]

Yet more troubles: at the end of the game, the modified Midpoint algorithm (the ‘search cone’) might not even be infrared safe
The definition of a sequential clustering algorithm, on the other hand, is extremely simple.

For instance, take the **longitudinally invariant** $k_t$:  

- Calculate the distances between the particles: $d_{ij} = \min(k_{ti}^2, k_{tj}^2)(\Delta \eta^2 + \Delta \phi^2)$
- Calculate the beam distances: $d_{iB} = k_{ti}^2$
- Combine particles with smallest distance or, if $d_{iB}$ is smallest, call it a jet
- Find again smallest distance and repeat procedure until no particles are left

This definition is infrared/collinear safe, has no artificial parameters, does not lead to dark towers or overlapping jets, can be applied equally well to data and theory.
Why FastJet

The $k_t$ jet-finder has, however, an apparent drawback: finding all the distances is an $N^2$ operation, to be repeated $N$ times.

$\Rightarrow \text{naively, the } k_t \text{ algorithm scales like } N^3$

**Time** taken to cluster $N$ particles:

Clustering quickly gets very slow: processing millions of events at LHC is simply not feasible with standard clustering algorithms.

- e.g. clustering a single heavy ion event at LHC would take 1 day of CPU!
Why FastJet

To improve the speed of the algorithm we must find more efficiently which particle is “close” to another and therefore gets combined with it.


If i and j form the smallest $d_{ij}$ and $k_{ti} < k_{tj}$

$\Rightarrow$ \hspace{1cm} $R_{ij} < R_{jk}$ \hspace{1cm} $\forall \ k \neq j$

(Approximate) translation from mathematics:

When a particle gets combined with another, its partner will be its geometrical nearest neighbour on the cylinder spanned by $\eta$ and $\phi$.

This means that we need to look for partners only among the $O(N)$ nearest neighbours of each particle.
Our problem has now become a **geometrical** one: how to find the (nearest) neighbour(s) of a point

Widely studied problem in computational geometry

**Tool:** Voronoi diagram

**Definition:** each cell contains the locations which have the given point as nearest neighbour

The **dual** of a Voronoi diagram is a Delaunay triangulation

Once the Voronoi diagram is constructed, the nearest neighbour of a point will be in one of the $O(1)$ cells sharing an edge with its own cell

Example: the G(eometrical) N(earest) N(eighbour) of point 7 will be found among 1, 4, 2, 8 and 3 (it turns out to be 3)
Why FastJet

The FastJet algorithm:

Construct the Voronoi diagram of the N particles using the CGAL library $O(N \ln N)$

Find the GNN of each of the N particles. Construct the $d_{ij}$ distances, store the results in a map $O(N \ln N)$

Merge/eliminate particles appropriately

Update Voronoi diagram and distances’ map $O(\ln N)$

repeat N times

Overall, an $O(N \ln N)$ algorithm

MC and G.P. Salam, hep-ph/0512210
Why FastJet

**Time** taken to cluster N particles:

Almost two orders of magnitude gain at small N ($O(N^2)$ implementation)

Large-N region now reachable
Why FastJet

‘Standard’ hard event.
Two well isolated jets

< 200 particles
Clustering easily doable even with standard algorithms
Why FastJet

Add minimum bias

~ 2000 particles

Clustering takes $O(20 \text{ s})$ with standard algorithms, but only $O(20 \text{ ms})$ with FastJet
Try to estimate \textbf{area} of each jet
Fill event with many very soft particles, count how many are clustered into given jet

\~ 10000 particles

Don’t even think about it with standard algorithms, \(O(1 \text{s})\) with \texttt{FastJet}
Why FastJet

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.
FastJet written in C++ and available at www.lpthe.jussieu.fr/~salam/fastjet

Extremely fast at small N, large N feasible (can cluster 50000 particles in O(3 s) )

Not a new clustering algorithm (results IDENTICAL to older and slower implementations of $k_t$)

However, the high speed allows one to do new things. Among others, cluster heavy ion events, and study the area of the jets

Full usefulness will only be clear with use. So, download it and run it!