Fast clustering of jets

Grégory Soyez

IPhT, CEA Saclay

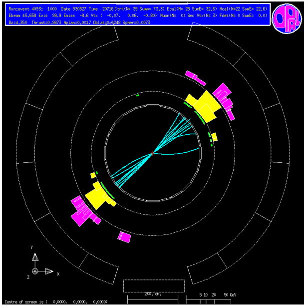
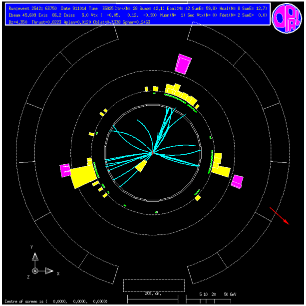
GDR QCD — progress in algorithms and numerical tools
May 16 2017

Brief plan

- Brief intro about the physics concepts: what are jets
- Clustering algorithms: Cambridge/Aachen, k_t, anti-k_t
- Main part: Nearest-neighbours and fast clustering
- If time left: Enumerating circles

Jets

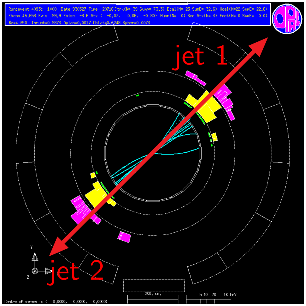
- Final-state events are pencil-like
  already observed in e^+e^- collisions:

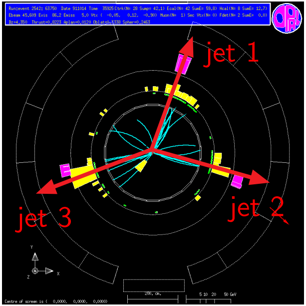



- Consequence of the collinear divergence
  QCD (quark & gluon) branching proba: $\frac{dP}{d\theta} \propto \frac{\alpha_s}{\theta}$

Jets

- Final-state events are pencil-like
  already observed in e^+e^- collisions:



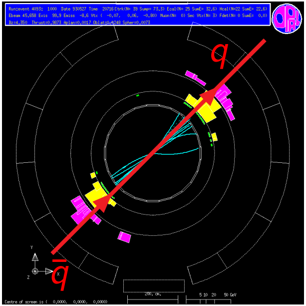


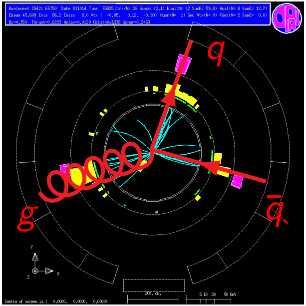
- Consequence of the collinear divergence
  QCD (quark & gluon) branching proba: $\frac{dP}{d\theta} \propto \frac{\alpha_s}{\theta}$

"Jets" $\equiv$ bunch of collimated particles

Jets

- Final-state events are pencil-like
  already observed in e^+e^- collisions:





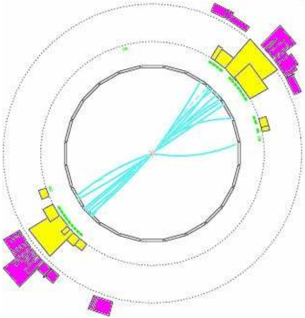
- Consequence of the collinear divergence
  QCD (quark & gluon) branching proba: $\frac{dP}{d\theta} \propto \frac{\alpha_s}{\theta}$

"Jets" $\equiv$ bunch of collimated particles $\cong$ hard partons

Jets and partons

"Jets" $\equiv$ bunch of collimated particles $\cong$ hard partons

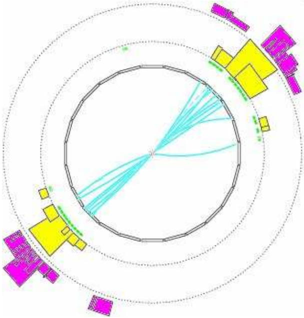
How many jets?



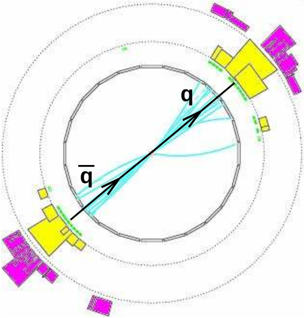
Jets and partons

"Jets" $\equiv$ bunch of collimated particles $\cong$ hard partons

obviously 2 jets

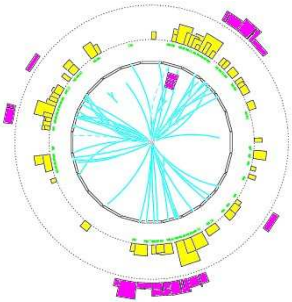


$\longrightarrow$



"Jets" $\equiv$ bunch of collimated particles $\cong$ hard partons

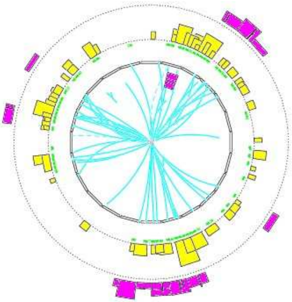
How many jets



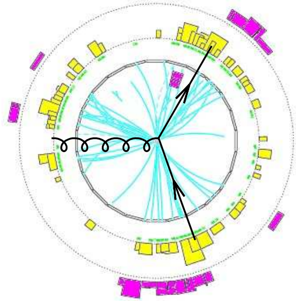
Jets and partons

"Jets" ≡ bunch of collimated particles ≅ hard partons

3 jets



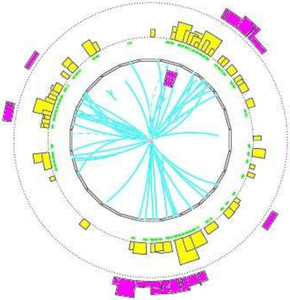
→



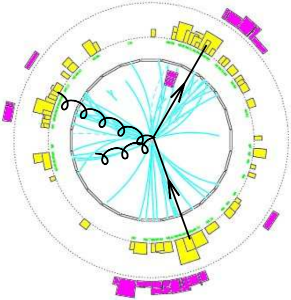
Jets and partons

"Jets" ≡ bunch of collimated particles ≅ hard partons

3 jets... or 4?



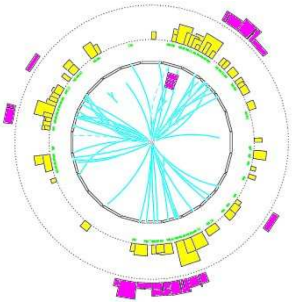
→



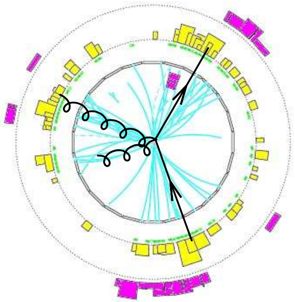
Jets and partons

"Jets" $\equiv$ bunch of collimated particles $\cong$ hard partons

3 jets... or 4?

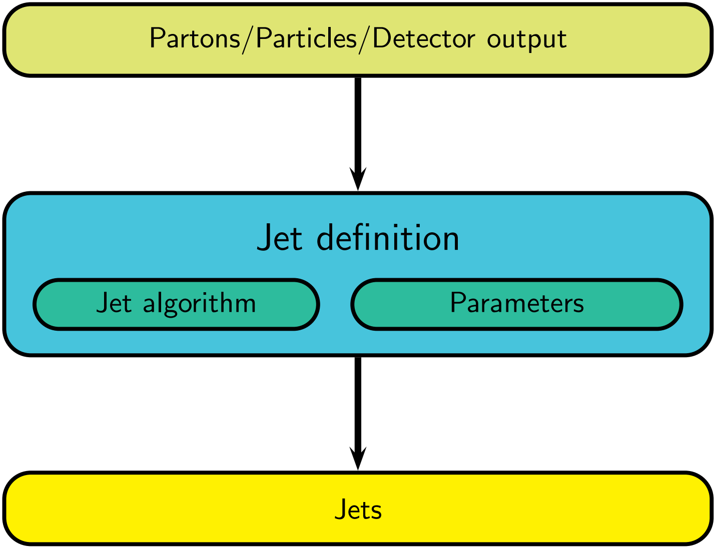


$\rightarrow$



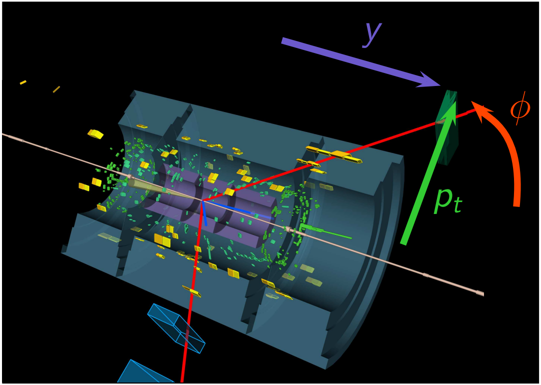
- "collinear" is arbitrary
- "parton" concept strictly valid only at LO

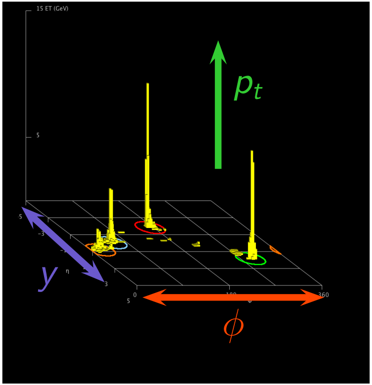
Jet definition



A bit of useful kinematics

[Both: ATLAS public events ($H \to 2\mu 2e$ & 4 jets)]





- Rapidity y: longitudinal component (along the beam axis)
- Azimuthal angle ϕ: around the beam axis
- Transverse momentum p_t: "energy" transverse to the beam

2 big approaches to jet clustering

- 30 years of history and debates
- All introduce a parameter R
  - "Jet radius"
  - distance of "collinearity" in $y - \phi$
- 2 big categories:
  - find circles (cones) containing flows of energy
  - undo a branching process by successive pairwise recombinations
- See Gavin's review from 2009 for details

Generalised-k_t algorithm

- From all the objects to cluster, define the distances

$$d_{ij} = \min(p_{t,i}^{2p}, p_{t,j}^{2p})(\Delta y_{ij}^2 + \Delta \phi_{ij}^2), \qquad d_{iB} = p_{t,i}^{2p} R^2$$

- repeatedly find the minimal distance
  if d_{ij}: recombine i and j into $k = i + j$
  if d_{iB}: call i a jet

Most common approach today: recombination algorithms

Generalised-k_t algorithm

- From all the objects to cluster, define the distances

$$d_{ij} = \min(p_{t,i}^{2p}, p_{t,j}^{2p})(\Delta y_{ij}^2 + \Delta\phi_{ij}^2), \qquad d_{iB} = p_{t,i}^{2p} R^2$$

- repeatedly find the minimal distance
  if d_{ij}: recombine i and j into $k = i + j$
  if d_{iB}: call i a jet

- Parameter p is (typically) one of
  - $p = 1$: k_t algorithm (closest to QCD)
    [Catani,Dokshitzer,Seymour,Weber,Ellis,Soper,1993]
  - $p = 0$: Cambridge/Aachen (geometrical distance)
    [Dokshitzer,Leder,Moretti,Webber,1997]
  - $p = -1$: anti-k_t (the LHC choice) [M.Cacciari,G.Salam,GS,2008]

Main question for today

1. Cambridge/Aachen

Given a set of N points (with weights p_t) in a plane ($y - \phi$) repeatedly find the closest pair

2. (anti-)k_t

Same, but use (anti)-k_t distance for measuring closeness

Things to keep in mind

- N is in the 1000-50000 range $\Rightarrow$ look at large N
- stopping distance R

Geometrical (Camb./Aachen) case: Naive approach

compute all d_{ij}	N^2
find minimum	N^2
recombine $i + j$	1
iterate	$\times N$
total	$\mathcal{O}(N^3)$

- works for all algs
- prohibitively slow

Geometrical (Camb./Aachen) case: Nearest neighbours

Observations:

- No need to keep track of all the distances:

$$\min_{i,j}\{d_{ij}\} = \min_i\{d_{i,NN(i)}\} \qquad \text{with} \quad NN(i) = \min_j\{d_{ij}\}$$

  only keep track of the nearest neighbour (NN) of each particle
- Do not recalculate all NNs at each step; if $i + j \to k$, we need $NN(k)$ and $NN(\ell)$ when $NN(\ell) = i$ or j

Geometrical (Camb./Aachen) case: Nearest neighbours

Observations:

- No need to keep track of all the distances:

$$\min_{i,j}\{d_{ij}\} = \min_i\{d_{i,NN(i)}\} \qquad \text{with} \quad NN(i) = \min_j\{d_{ij}\}$$

  only keep track of the nearest neighbour (NN) of each particle

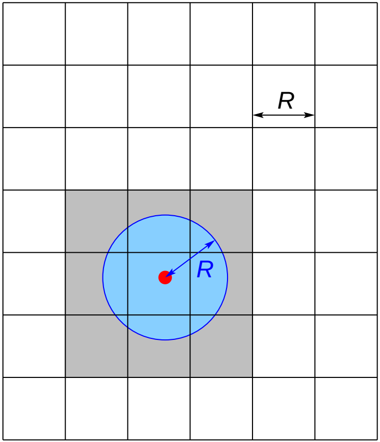
- Do not recalculate all NNs at each step; if $i + j \to k$, we need $NN(k)$ and $NN(\ell)$ when $NN(\ell) = i$ or j

New implementation:

Init: compute all $NN(i)$	N^2
find smallest $d_{i,NN(i)}$	N
recombine $i + j$	1
compute $NN(k)$ and $NN(\ell)$'s	N
iterate	$\times N$
total	$\mathcal{O}(N^2)$

- works for all algs
- efficient for N not too large

Geometrical (Camb./Aachen) case: Tiling



- NN only in current or neighbouring tile
- $\Rightarrow$ NN search is $\mathcal{O}\left(n = N/N_{\text{tiles}}\right)$

Geometrical (Camb./Aachen) case: Tiling

Init: create tiling	N
Init: compute all $NN(i)$	Nn
Init: sort the $d_{i,NN(i)}$	$N\log(N)$
find smallest $d_{i,NN(i)}$	1
recombine $i+j$	1
compute $NN(k)$ and $NN(\ell)$'s	n
iterate	$\times N$
total	$\mathcal{O}(Nn)$

- NN only in current or neighbouring tile
- $\Rightarrow$ NN search is $\mathcal{O}(n = N/N_{\text{tiles}})$

Geometrical (Camb./Aachen) case: Tiling

Init: create tiling	N
Init: compute all $NN(i)$	Nn
Init: sort the $d_{i,NN(i)}$	$N\log(N)$
find smallest $d_{i,NN(i)}$	1
recombine $i+j$	1
compute $NN(k)$ and $NN(\ell)$'s	n
iterate	$\times N$
total	$\mathcal{O}(Nn)$

- NN only in current or neighbouring tile
- $\Rightarrow$ NN search is $\mathcal{O}(n = N/N_{\text{tiles}})$
- Valid for Cambr./Aachen ($d_{ij} = \Delta R_{ij}^2$)
- Variants for finding $\min\{d_{i,NN(i)}\}$.
- Tricks to avoid neighbour tiles when possible

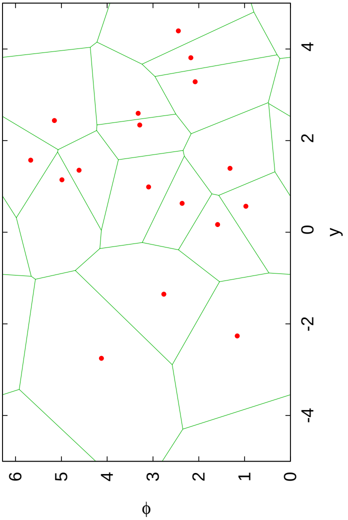
Geometrical (Camb./Aachen) case: Tiling

Init: create tiling	N
Init: compute all $NN(i)$	Nn
Init: sort the $d_{i,NN(i)}$	$N\log(N)$
find smallest $d_{i,NN(i)}$	1
recombine $i+j$	1
compute $NN(k)$ and $NN(\ell)$'s	n
iterate	$\times N$
total	$\mathcal{O}(Nn)$

- NN only in current or neighbouring tile
- $\Rightarrow$ NN search is $\mathcal{O}(n = N/N_{\text{tiles}})$
- Valid for Cambr./Aachen $(d_{ij} = \Delta R_{ij}^2)$
- Variants for finding $\min\{d_{i,NN(i)}\}$.
- Tricks to avoid neighbour tiles when possible
- **Optimal for** $30 \lesssim N \lesssim 5\,10^5$

Still can do better

[M.Cacciari,G.Salam,2005]



- Voronoi graph: bisectors between pairs of points
- $NN(i)$ is one of the $\mathcal{O}(\log N)$ adjacent cells
- Construct: $\mathcal{O}(N \log N)$
- Add/remove: $\mathcal{O}(\log N)$

Still can do better

[M.Cacciari,G.Salam,2005]

Init: create Voronoi graph	$N\log(N)$
compute&sort $NN(i)$	$N\log(N)$
find smallest $d_{i,NN(i)}$	1
recombine $i+j$	1
update Voronoi	$\log N$
iterate	$\times N$
total	$\mathcal{O}(N\log N)$

- Voronoi graph: bisectors between pairs of points
- $NN(i)$ is one of the $\mathcal{O}(\log N)$ adjacent cells
- Construct: $\mathcal{O}(N\log N)$
- Add/remove: $\mathcal{O}(\log N)$

Uses http://www.ccgal.org (divide-and-conquer) [alternative: sweep line]

Still can do better

[M.Cacciari,G.Salam,2005]

Init: create Voronoi graph	$N\log(N)$
compute&sort $NN(i)$	$N\log(N)$
find smallest $d_{i,NN(i)}$	1
recombine $i+j$	1
update Voronoi	$\log N$
iterate	$\times N$
total	$\mathcal{O}(N\log N)$

- Voronoi graph: bisectors between pairs of points
- $NN(i)$ is one of the $\mathcal{O}(\log N)$ adjacent cells
- Construct: $\mathcal{O}(N\log N)$
- Add/remove: $\mathcal{O}(\log N)$
- Theoretical bound
- Variants according to treatment of periodicity
- Alternative: T. Chan's closest pair algorithm

Uses http://www.ccgal.org (divide-and-conquer) [alternative: sweep line]

Still can do better

[M.Cacciari,G.Salam,2005]

Init: create Voronoi graph	$N \log(N)$
compute&sort $NN(i)$	$N \log(N)$
find smallest $d_{i,NN(i)}$	1
recombine $i + j$	1
update Voronoi	$\log N$
iterate	$\times N$
total	$\mathcal{O}(N \log N)$

- Voronoi graph: bisectors between pairs of points
- $NN(i)$ is one of the $\mathcal{O}(\log N)$ adjacent cells
- Construct: $\mathcal{O}(N \log N)$
- Add/remove: $\mathcal{O}(\log N)$
- Theoretical bound
- Variants according to treatment of periodicity
- Alternative: T. Chan's closest pair algorithm
- **Optimal for large N**

Uses http://www.ccgal.org (divide-and-conquer) [alternative: sweep line]

Other algorithms

What about $d_{ij} = \min(p_{ti}^{2p}, p_{tj}^{2p})\,\Delta R_{ij}^2$?

Other algorithms

What about $d_{ij} = \min(p_{ti}^{2p}, p_{tj}^{2p})\,\Delta R_{ij}^2$?

FastJet lemma

If the pair (i, j) minimises d_{ij} and $p_{ti}^{2p} < p_{tj}^{2p}$,
then j is the geometrical NN of i.

Proof: Assume there is k s.t. $\Delta R_{ik} < \Delta R_{ij}$. We would have

$$\begin{aligned} d_{ik} &= \min(p_{ti}^{2p}, p_{tk}^{2p})\,\Delta R_{ik}^2 \\ &< p_{ti}^{2p}\,\Delta R_{ij}^2 = d_{ij}, \end{aligned}$$

a contradiction.

Other algorithms

What about $d_{ij} = \min(p_{ti}^{2p}, p_{tj}^{2p})\,\Delta R_{ij}^2$?

FastJet lemma

If the pair (i,j) minimises d_{ij} and $p_{ti}^{2p} < p_{tj}^{2p}$,
then j is the geometrical NN of i.

Proof: Assume there is k s.t. $\Delta R_{ik} < \Delta R_{ij}$. We would have

$$
\begin{aligned}
d_{ik} &= \min(p_{ti}^{2p}, p_{tk}^{2p})\,\Delta R_{ik}^2 \\
&< p_{ti}^{2p}\,\Delta R_{ij}^2 = d_{ij},
\end{aligned}
$$

a contradiction.

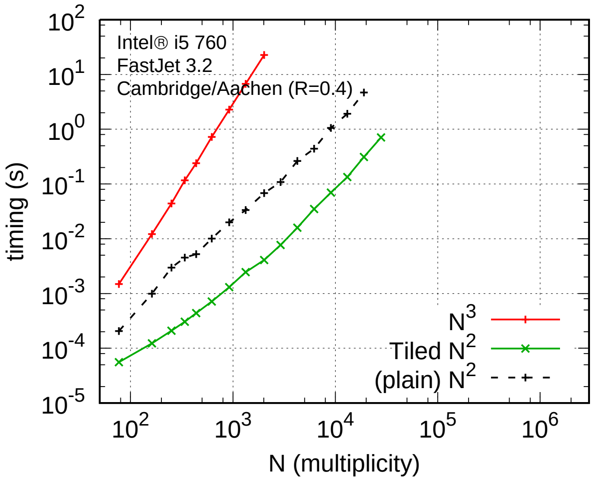
$\Rightarrow$ all the above strategy (working with geometrical NN) work

http://www.fastjet.fr (1/2)

[M.Cacciari, G.Salam, 2005]
[M.Cacciari, G.Salam, GS, 2011]

- Matteo Cacciari and Gavin Salam in 2005; I joined in 2008.
- `http://www.fastjet.fr`
- Software for fast jet clustering
- Now extended to reference software for jet clustering + manipulations
- Used by the whole LHC community

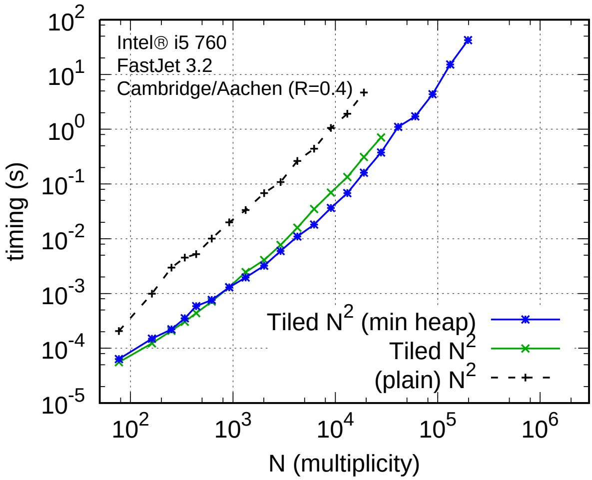
[M.Cacciari, G.Salam, 2005]
[M.Cacciari, G.Salam, GS, 2011]



- N^2 and tiling helps

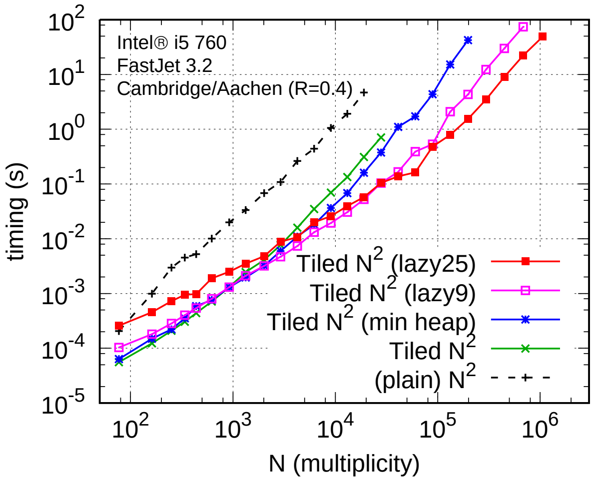
http://www.fastjet.fr (2/2)

[M.Cacciari, G.Salam, 2005]
[M.Cacciari, G.Salam, GS, 2011]



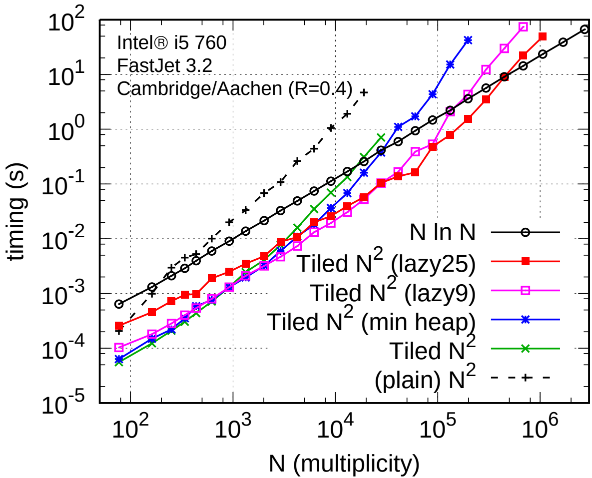
- N^2 and tiling helps
- tiled variant

[M.Cacciari, G.Salam, 2005]
[M.Cacciari, G.Salam, GS, 2011]



- N^2 and tiling helps
- tiled variant
- "lazy" version (9 or 25 tiles) New in FJ-3.1

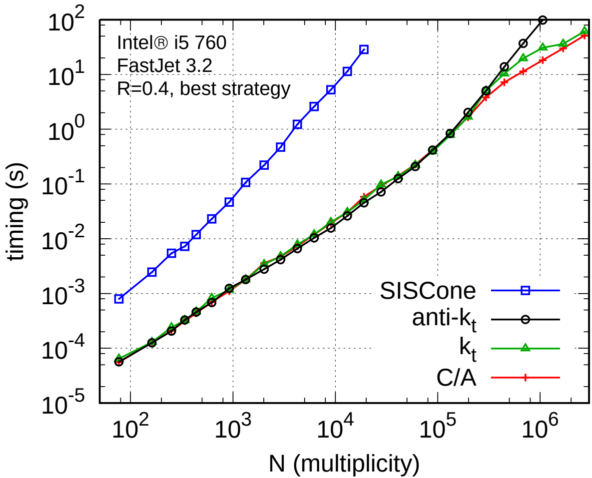
[M.Cacciari, G.Salam, 2005]
[M.Cacciari, G.Salam, GS, 2011]



- N^2 and tiling helps
- tiled variant
- "lazy" version (9 or 25 tiles) New in FJ-3.1
- CGAL, $N \ln N$

http://www.fastjet.fr (2/2)

[M.Cacciari, G.Salam, 2005]
[M.Cacciari, G.Salam, GS, 2011]



- works for $k_t (N \ln N)$ and antik-$k_t (N^{3/2})$
- SISCone: see next slides
- at LHC: 1000× faster than “KtJet”

Stale cones and related problems

- Cone algorithm: find directions of energy flow

Stale cones and related problems

- Cone algorithm: find directions of energy flow
- Stable cone: circle of radius R (in $y - \phi$ plane) s.t. sum of momenta in the cone points towards the centre.

Stale cones and related problems

- Cone algorithm: find directions of energy flow
- Stable cone: circle of radius R (in $y - \phi$ plane) s.t. sum of momenta in the cone points towards the centre.

- Big question: how to find the stable cones?

Stale cones and related problems

- Cone algorithm: find directions of energy flow
- Stable cone: circle of radius R (in $y - \phi$ plane) s.t. sum of momenta in the cone points towards the centre.

- Big question: how to find the stable cones?

- Similar ptoblems: other similar problems where one partition the plane with a circle or a line and check for a condition. (Example: thrust)

Stale cones and related problems

- Cone algorithm: find directions of energy flow
- Stable cone: circle of radius R (in $y - \phi$ plane) s.t. sum of momenta in the cone points towards the centre.

- Big question: how to find the stable cones?

- Similar ptoblems: other similar problems where one partition the plane with a circle or a line and check for a condition. (Example: thrust)

- Naive approach: test every possile partition $\Rightarrow \mathcal{O}\left(N\,2^N\right)$

Stale cones and related problems

- Cone algorithm: find directions of energy flow
- Stable cone: circle of radius R (in $y - \phi$ plane) s.t. sum of momenta in the cone points towards the centre.

- Big question: how to find the stable cones?

- Similar ptoblems: other similar problems where one partition the plane with a circle or a line and check for a condition. (Example: thrust)

- Naive approach: test every possile partition $\Rightarrow \mathcal{O}\left(N 2^N\right)$
- Tevatron solution: iterative approach starting from a set of "seeds"

Stale cones and related problems

- Cone algorithm: find directions of energy flow
- Stable cone: circle of radius R (in $y - \phi$ plane) s.t. sum of momenta in the cone points towards the centre.

- Big question: how to find the stable cones?

- Similar ptoblems: other similar problems where one partition the plane with a circle or a line and check for a condition. (Example: thrust)

- Naive approach: test every possile partition $\Rightarrow$ $\mathcal{O}\left(N\,2^N\right)$
- Tevatron solution: iterative approach starting from a set of "seeds"
  - reasonable speed ($\mathcal{O}\left(N^3\right)$)

Stale cones and related problems

- Cone algorithm: find directions of energy flow
- Stable cone: circle of radius R (in $y - \phi$ plane) s.t. sum of momenta in the cone points towards the centre.

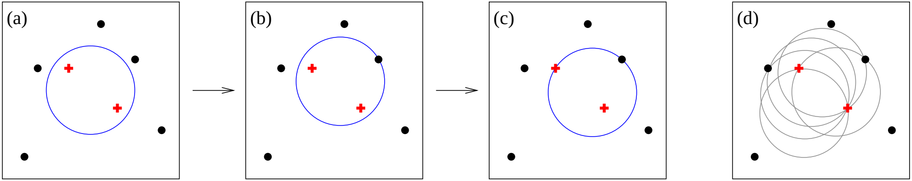
- Big question: how to find the stable cones?

- Similar ptoblems: other similar problems where one partition the plane with a circle or a line and check for a condition. (Example: thrust)

- Naive approach: test every possile partition $\Rightarrow$ $\mathcal{O}\left(N\,2^N\right)$
- Tevatron solution: iterative approach starting from a set of "seeds"
  - reasonable speed ($\mathcal{O}\left(N^3\right)$)
  - Infrared (or collinear) unsafe $\Rightarrow$ not good for theory

The SISCone solution

[G.Salam,GS,2007]

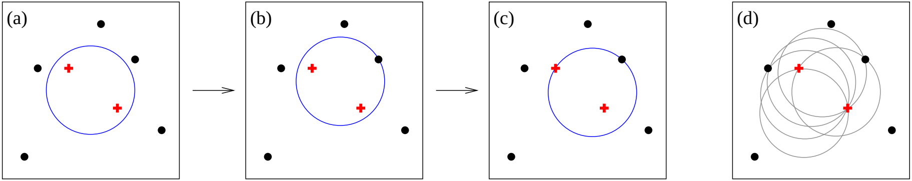


(a) start with a circle

see https://siscone.hepforge.org/

The SISCone solution

[G.Salam,GS,2007]



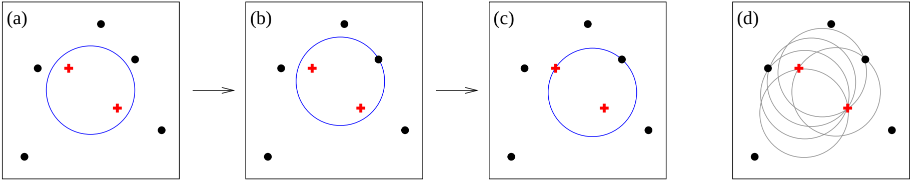
(a) start with a circle

(b) it can be moved until it hits a first point

see `https://siscone.hepforge.org/`

The SISCone solution

[G.Salam,GS,2007]



(a) start with a circle

(b) it can be moved until it hits a first point

(c) it can e rotated until it touches a second

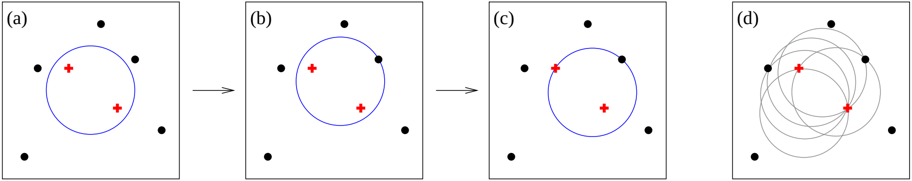
$\Rightarrow$ enumerate circles by enumerating pairs of points

enumerate pairs: N^2, check stability: $N \Rightarrow \mathcal{O}\left(N^3\right)$

see https://siscone.hepforge.org/

The SISCone solution

[G.Salam,GS,2007]



(a) start with a circle

(b) it can be moved until it hits a first point

(c) it can e rotated until it touches a second

$\Rightarrow$ enumerate circles by enumerating pairs of points

enumerate pairs: N^2, check stability: $N \Rightarrow \mathcal{O}\left(N^3\right)$

(d) order the circles in angle $N \Rightarrow \mathcal{O}\left(N^2 \ln N\right)$

see https://siscone.hepforge.org/

Conclusions

Geometrical constructions can help designing powerful algorithms

- tilings, Voronoi graphs for iterative clustering
- circles enumeration for cone algorithms

Thank You!