

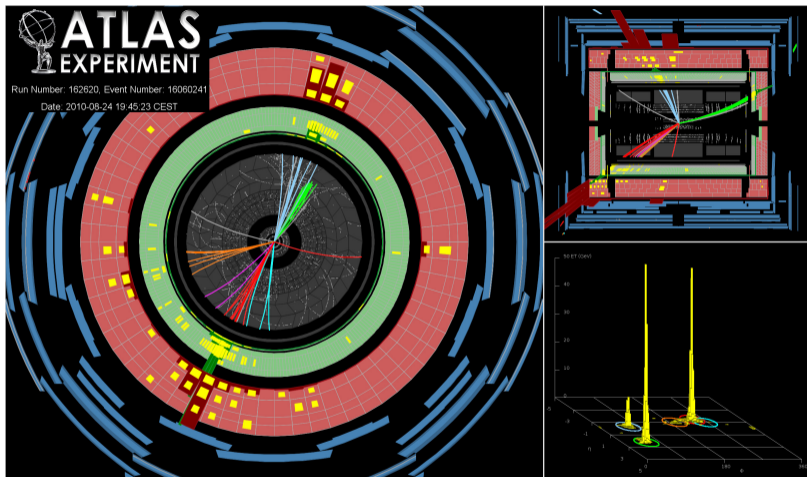
FastJet

Gregory Soyez (with Matteo Cacciari and Gavin Salam)

IPhT, CNRS, CEA Saclay

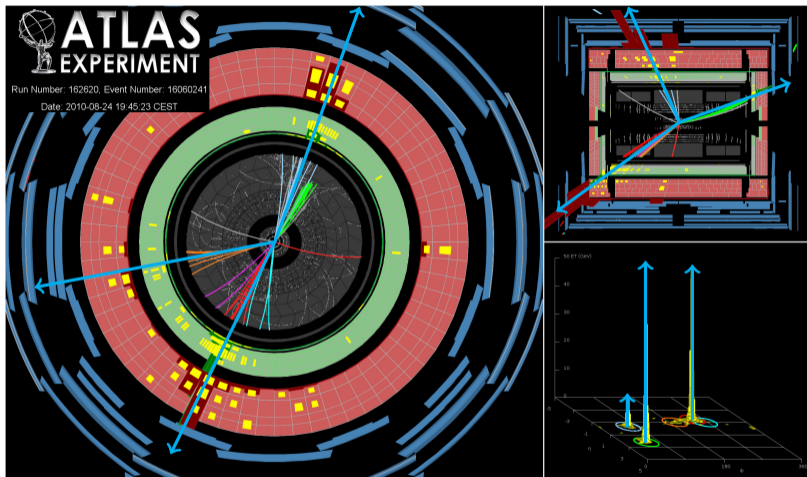
Joint GDR-QCD/Strong2020, May 31-June 4 2021, IJCLab (online)

Some physics background

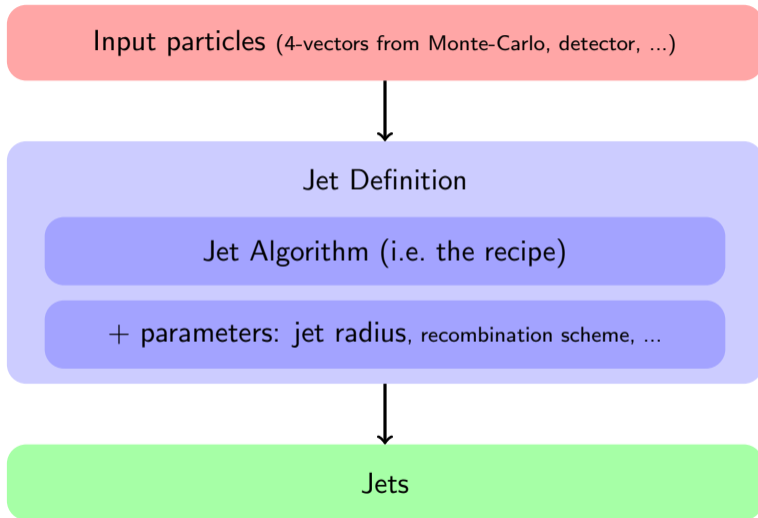


Particles/Energy flow organised in a few dominant directions \Rightarrow JETS

Some physics background



Particles/Energy flow organised in a few dominant directions \Rightarrow JETS



Generalised- k_t algorithm

- From all the objects to cluster, define the distances

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

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

Generalised- k_t algorithm

- From all the objects to cluster, define the distances

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

- 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]

FastJet is a C++ interface (*) for

- 1 fast jet clustering
- 2 jet manipulations

See fastjet.fr

Cite [arXiv:1111.6097](https://arxiv.org/abs/1111.6097)
and [hep-ph/0512210](https://arxiv.org/abs/hep-ph/0512210)

[(*) also available in Python]

Home About Releases Quick start Manual Doxygen Tools Contrib FAQ

FastJet

A software package for jet finding in pp and e^+e^- collisions. It includes fast native implementations of many sequential recombination clustering algorithms, plugins for access to a range of cone jet finders and tools for advanced jet manipulation.

Release of FastJet 3.4.0-beta.1, (new pre-release) 10 March 2021 ([release notes](#)).
This is a (beta) pre-release of FastJet-3.4.0. The main new feature is the support for thread safety (through the `--enable-thread-safety` configure option). Other additions include facilities to get/set the seeds used to generate ghosts for jet area calculations and a new interface for background estimation. See the [full release notes](#) for details. [Download](#)

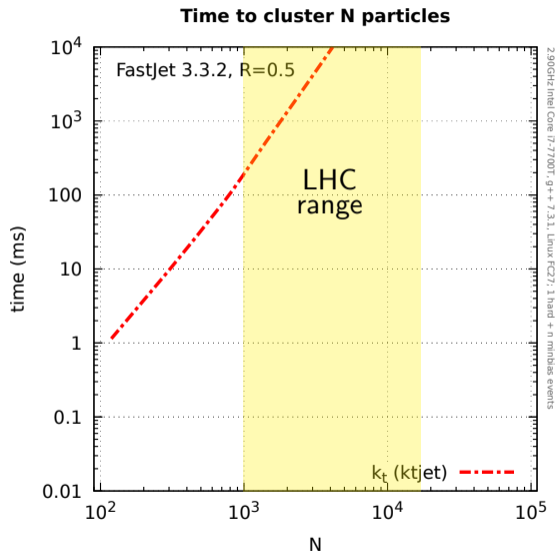
Release of FastJet 3.3.4, (latest stable release) 05 May 2020 ([release notes](#)).
This is a maintenance release with clarifications about CGAL5 support and fixes for certain strict g++ options and Oracle c++. It includes SIScone v3.0.5. [Download](#)

Latest stable release of fjcore (v3.3.4), 05 May 2020
Lightweight access to the core FastJet functionality (PseudoJet, JetDefinition, ClusterSequence and Selector). It consists of just two files, `fjcore.hh` and `fjcore.cc`, which can easily be included in 3rd party projects. Compile time: a few seconds. A fortran interface and basic examples are also included in the distribution. [Download](#) size: 73k.

Release of FastJet Contrib 1.045, 4 August 2020
A package of contributed add-ons to FastJet. This release brings the new Centauro (DIS jet algorithm) contrib, v1.0.0.

© 2005-2021 Matteo Cacciari, Gavin P. Salam, Gregory Soyez - [Bug report](#) - [Subscribe](#) - [Follow @fastjet_fr](#)

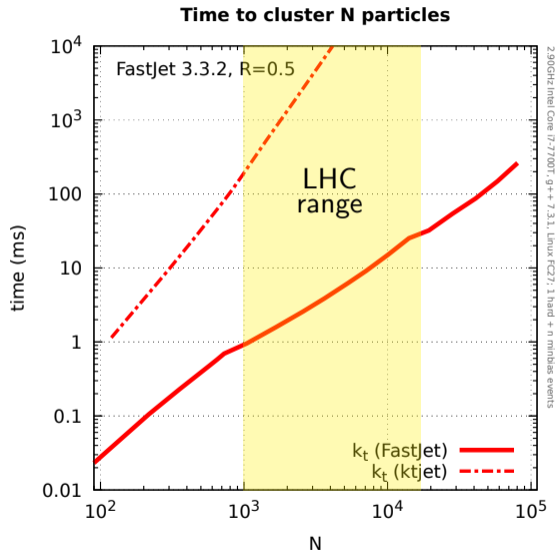
Fast clustering



- k_t before FastJet
Complexity $\propto N^3$ (*)

(*) more on that later if enough time

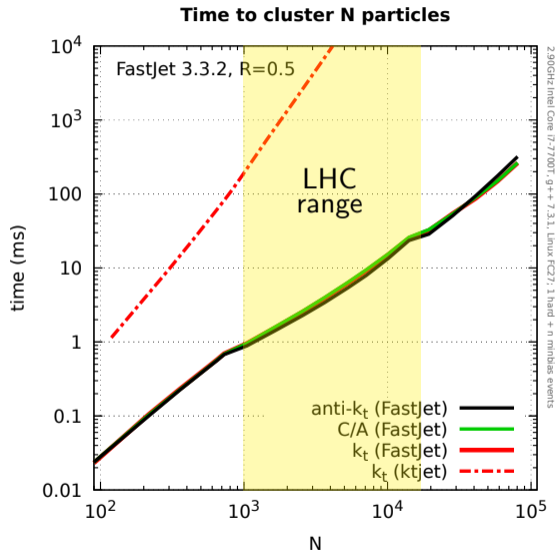
Fast clustering



- k_t before FastJet
Complexity $\propto N^3$ (*)
- k_t Fastjet's implementation
gain 2-3 orders of magnitude
Makes it usable at the LHC
(Trigger spends $\sim 100\text{ms}/\text{event}$)
Complexity $\propto N^2$ or $N \ln N$ (*)

(*) more on that later if enough time

Fast clustering



- k_t before FastJet
Complexity $\propto N^3$ (*)
- k_t Fastjet's implementation
gain 2-3 orders of magnitude
Makes it usable at the LHC
(Trigger spends $\sim 100\text{ms}/\text{event}$)
Complexity $\propto N^2$ or $N \ln N$ (*)
- Similar time for anti- k_t and C/A

(*) more on that later if enough time

Simple code

All done in a few lines of code:

```
// list of input particles/objects
vector<PseudoJet> particles;

// Cluster with anti- $k_t$ ,  $R = 0.5$ 
JetDefinition jet_def(antikt_algorithm, 0.5);

// Get the jets with  $p_t > 50$  GeV,  $|y| < 2.5$ 
Selector jet_selector = SelectorPtMin(50.0) * SelectorAbsRapMax(2.5);
vector<PseudoJet> jets = jet_selector(jet_def(particles));

// Simple manipulations
for (auto &jet : jets){
    double pt = jet.pt(); // jet pt
    vector<PseudoJet> constituents = jet.constituents(); // particles that made up the jet
}
```

Simple code

All done in a few lines of code:

```
// list of input particles/objects
vector<PseudoJet> particles;

// Cluster with anti- $k_t$ ,  $R = 0.5$ 
JetDefinition jet_def(antikt_algorithm, 0.5);

// Get the jets with  $p_t > 50$  GeV,  $|y| < 2.5$ 
Selector jet_selector = SelectorPtMin(50.0) * SelectorAbsRapMax(2.5);
vector<PseudoJet> jets = jet_selector(jet_def(particles));

// Simple manipulations
for (auto &jet : jets){
    double pt = jet.pt(); // jet pt
    vector<PseudoJet> constituents = jet.constituents(); // particles that made up the jet
}
```

In a nutshell: jet reconstruction/selection/operations made easy

Grown into framework for jet manipulation

Two main directions:

- 1 Background mitigation
- 2 Jet substructure

(Area-based) background subtraction (pileup or heavy-ion)

Often (in experimental contexts) events are polluted by large backgrounds

- Pileup in pp (superposition of several concurrent pp events)
- QGP medium/background in HI collisions

Basic approach

Based on the observation that backgrounds are (quasi-)homogeneous in rapidity-azimuth

- 1 Compute jet areas A_i (for each jet)
- 2 Estimate the background ρ for the event
- 3 Subtract each jet

(Area-based) background subtraction (pileup or heavy-ion)

Often (in experimental contexts) events are polluted by large backgrounds

- Pileup in pp (superposition of several concurrent pp events)
- QGP medium/background in HI collisions

Basic approach

Based on the observation that backgrounds are (quasi-)homogeneous in rapidity-azimuth

- 1 Compute jet areas A_i (for each jet)
- 2 Estimate the background ρ for the event
- 3 Subtract each jet

$$\rho = \text{median}_{i \in \text{jets}} \frac{p_{ti}}{A_i}$$

$$p_t^{(\text{sub})} = p_t - \rho A$$

(Area-based) background subtraction (pileup or heavy-ion)

Often (in experimental contexts) events are polluted by large backgrounds

- Pileup in pp (superposition of several concurrent pp events)
- QGP medium/background in HI collisions

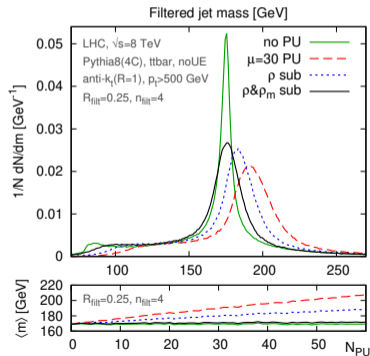
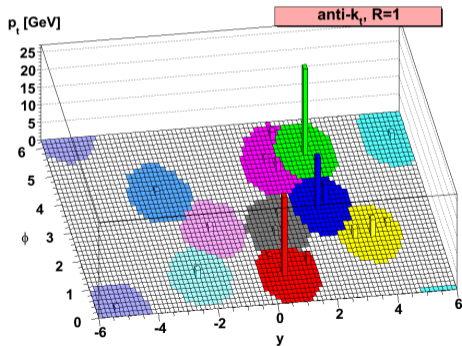
FastJet provides:

- ▶ A computation of the jet areas [ClusterSequenceArea]
- ▶ Tools to estimate the background density [Jet/GridMedianBackgroundEstimator]
- ▶ Tools to perform the subtraction [Subtractor]
- ▶ Facilities to implement user-defined subtraction methods

(Area-based) background subtraction (pileup or heavy-ion)

Often (in experimental contexts) events are polluted by large backgrounds

- Pileup in pp (superposition of several concurrent pp events)
- QGP medium/background in HI collisions



Core interface for substructure tools

One often need post-processing of jets, typically in [jet substructure](#) studies

FastJet provides

- ▶ Basic (historical) tools
- ▶ Common interface for user-defined tools

```
// apply a MassDropTagger
MassDropTagger md_tagger(0.667, 0.09);
PseudoJet tagged = md_tagger(jet);

// test if the tagger succeeded
if (tagged != 0){
    // apply a filter
    Filter filter(some_jet_def, some_selection);
    PseudoJet filtered = filter(tagged);
}
```

Many additional tools in [fastjet-contrib](#)

[\[link\]](#)

Package	Version	Release date
Centauro	1.0.0	2020-08-04
ClusteringVetoPlugin	1.0.0	2015-05-04
ConstituentSubtractor	1.4.5	2020-02-23
EnergyCorrelator	1.3.1	2018-02-10
FlavorCone	1.0.0	2017-09-07
GenericSubtractor	1.3.1	2016-03-30
JetCleanser	1.0.1	2014-08-16
JetFFMoments	1.0.0	2013-02-07
JetsWithoutJets	1.0.0	2014-02-22
LundPlane	1.0.3	2020-02-23
Nsubjettiness	2.2.5	2018-06-06
QCDAwarePlugin	1.0.0	2015-10-08
RecursiveTools	2.0.0	2020-03-03
ScJet	1.1.0	2013-06-03
SoftKiller	1.0.0	2014-08-17
SubjetCounting	1.0.1	2013-09-03
ValenciaPlugin	2.0.2	2018-12-22
VariableR	1.2.1	2016-06-01

Core interface for substructure tools

One often need post-processing of jets, typically in [jet substructure](#) studies

FastJet provides

- ▶ Basic (historical) tools
- ▶ Common interface for user-defined tools

```
// apply a MassDropTagger
contrib::SoftDrop sd(2.0, 0.1);
PseudoJet tagged = sd(jet);

// test if the tagger succeeded
if (tagged != 0){
  // apply a filter
  Filter filter(some_jet_def, some_selection);
  PseudoJet filtered = filter(tagged);
}
```

Many additional tools in [fastjet-contrib](#)

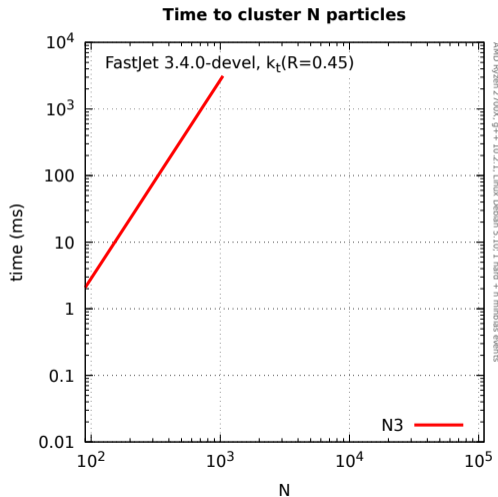
[\[link\]](#)

Package	Version	Release date
Centauro	1.0.0	2020-08-04
ClusteringVetoPlugin	1.0.0	2015-05-04
ConstituentSubtractor	1.4.5	2020-02-23
EnergyCorrelator	1.3.1	2018-02-10
FlavorCone	1.0.0	2017-09-07
GenericSubtractor	1.3.1	2016-03-30
JetCleanser	1.0.1	2014-08-16
JetFFMoments	1.0.0	2013-02-07
JetsWithoutJets	1.0.0	2014-02-22
LundPlane	1.0.3	2020-02-23
Nsubjettiness	2.2.5	2018-06-06
QCDAwarePlugin	1.0.0	2015-10-08
RecursiveTools	2.0.0	2020-03-03
ScJet	1.1.0	2013-06-03
SoftKiller	1.0.0	2014-08-17
SubjetCounting	1.0.1	2013-09-03
ValenciaPlugin	2.0.2	2018-12-22
VariableR	1.2.1	2016-06-01

What makes FastJet fast?



N particles to cluster $\Rightarrow \mathcal{O}(N)$ recombinations steps to get a finite number of jets



Basic approach ($\mathcal{O}(N^3)$)

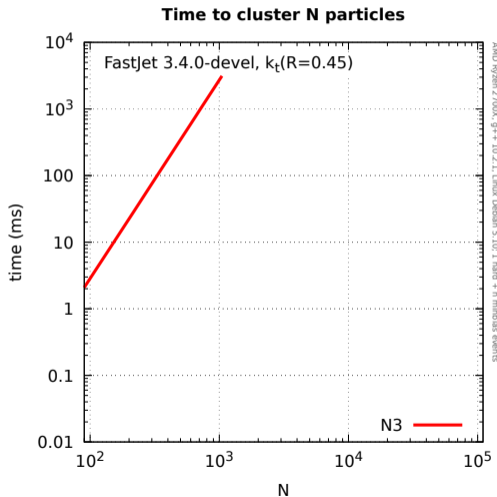
Repeat $\mathcal{O}(N)$ times

- ▶ find minimum d_{ij} [$\mathcal{O}(N^2)$]
- ▶ recombine [$\mathcal{O}(1)$]

Total: $N \times N^2 = N^3$

N particles to cluster

$\Rightarrow \mathcal{O}(N)$ recombinations steps to get a finite number of jets



FastJet lemma

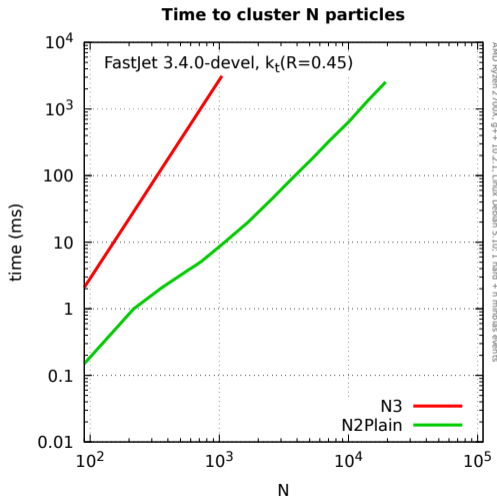
If i, j is the minimum $d_{ij} = \min(p_{ti}^2, p_{tj}^2) \Delta R_{ij}^2$ with $p_{ti} < p_{tj}$, then $\Delta R_{ij} < \Delta R_{ik} \forall k \neq i, j$.
I.e. the (gen-) k_t minimum is one of the geometrical nearest neighbours

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

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

a contradiction

N particles to cluster $\Rightarrow \mathcal{O}(N)$ recombinations steps to get a finite number of jets



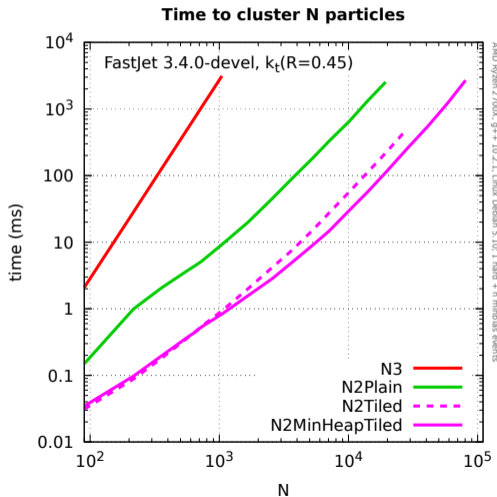
FastJet approach #1 ($\mathcal{O}(N^2)$)

- find NN_i nearest neighbour for each i [$\mathcal{O}(N^2)$]
- Repeat $\mathcal{O}(N)$ times
 - ▶ find minimum d_{ij} in NNs [$\mathcal{O}(N)$]
 - ▶ recombine [$\mathcal{O}(1)$]
 - ▶ update a few NN [$\mathcal{O}(N)$]

Total: $N^2 + N \times N = N^2$

N particles to cluster

$\Rightarrow \mathcal{O}(N)$ recombinations steps to get a finite number of jets



FastJet approach #1 ($\mathcal{O}(N^2)$)

- find NN_i nearest neighbour for each i [$\mathcal{O}(N^2)$]
- Repeat $\mathcal{O}(N)$ times
 - ▶ find minimum d_{ij} in NNs [$\mathcal{O}(N)$]
 - ▶ recombine [$\mathcal{O}(1)$]
 - ▶ update a few NN [$\mathcal{O}(N)$]

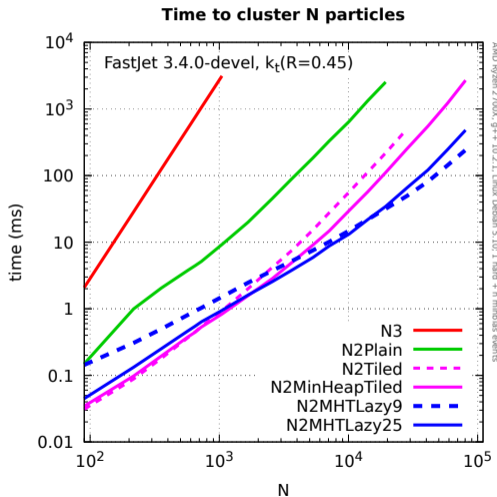
Total: $N^2 + N \times N = N^2$

more $\mathcal{O}(N^2)$ FastJet approaches

- Use tiling to limit to local search

N particles to cluster

$\Rightarrow \mathcal{O}(N)$ recombinations steps to get a finite number of jets



FastJet approach #1 ($\mathcal{O}(N^2)$)

- find NN_i nearest neighbour for each i [$\mathcal{O}(N^2)$]
- Repeat $\mathcal{O}(N)$ times
 - ▶ find minimum d_{ij} in NNs [$\mathcal{O}(N)$]
 - ▶ recombine [$\mathcal{O}(1)$]
 - ▶ update a few NN [$\mathcal{O}(N)$]

Total: $N^2 + N \times N = N^2$

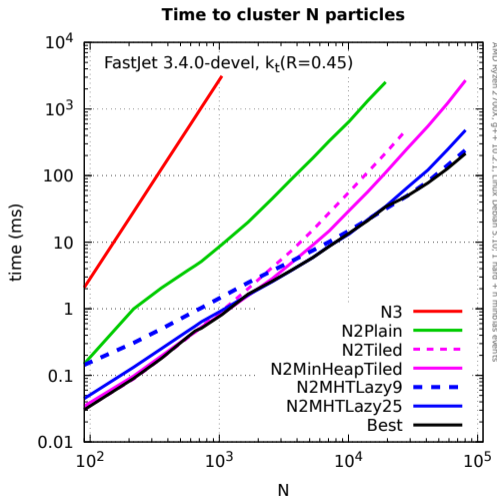
more $\mathcal{O}(N^2)$ FastJet approaches

- Use tiling to limit to local search
- Limit unnecessary searches

Discussion about complexity

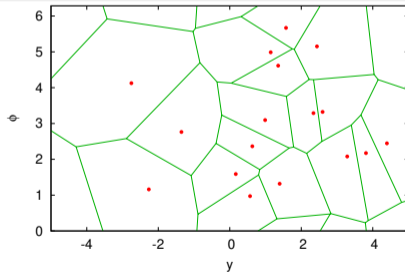
N particles to cluster

$\Rightarrow \mathcal{O}(N)$ recombinations steps to get a finite number of jets



FastJet ultimate: $\mathcal{O}(N \ln N)$

Use Voronoi graphs (Delaunay triangulations)
Limits the initialisation to $\mathcal{O}(N \ln N)$ and the search to $\ln N$. (Optimal at larger N)



FastJet is a C++ interface (*) for

- 1 fast jet clustering
- 2 jet manipulations

See fastjet.fr

Cite [arXiv:1111.6097](https://arxiv.org/abs/1111.6097)
and [hep-ph/0512210](https://arxiv.org/abs/hep-ph/0512210)

[(*) also available in Python]

[Home](#) [About](#) [Releases](#) [Quick start](#) [Manual](#) [Doxygen](#) [Tools](#) [Contrib](#) [FAQ](#)

FastJet

A software package for jet finding in pp and e^+e^- collisions. It includes fast native implementations of many sequential recombination clustering algorithms, plugins for access to a range of cone jet finders and tools for advanced jet manipulation.

Release of **FastJet 3.4.0-beta.1**, (new pre-release) 10 March 2021 ([release notes](#)).

This is a (beta) pre-release of FastJet-3.4.0. The main new feature is the support for thread safety (through the `--enable-thread-safety` configure option). Other additions include facilities to get/set the seeds used to generate ghosts for jet area calculations and a new interface for background estimation. See the [full release notes](#) for details.

[Download](#)

Release of **FastJet 3.3.4**, (latest stable release) 05 May 2020 ([release notes](#)).

This is a maintenance release with clarifications about CGAL5 support and fixes for certain strict g++ options and Oracle c++. It includes SIScone v3.0.5.

[Download](#)

Latest stable release of **fjcore** (v3.3.4), 05 May 2020

Lightweight access to the core FastJet functionality (PseudoJet, JetDefinition, ClusterSequence and Selector).

It consists of just two files, `fjcore.hh` and `fjcore.cc`, which can easily be included in 3rd party projects. Compile time: a few seconds. A fortran interface and basic examples are also included in the distribution. [Download](#) size: 73k.

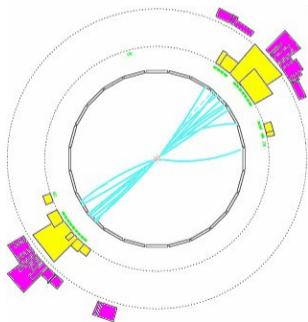
Release of **FastJet Contrib 1.045**, 4 August 2020

A package of contributed add-ons to FastJet. This release brings the new Centauro (DIS jet algorithm) contrib, v1.0.0.

© 2005-2021 Matteo Cacciari, Gavin P. Salam, Gregory Soyez - [Bug report](#) - [Subscribe](#) - [Follow @fastjet_fr](#)

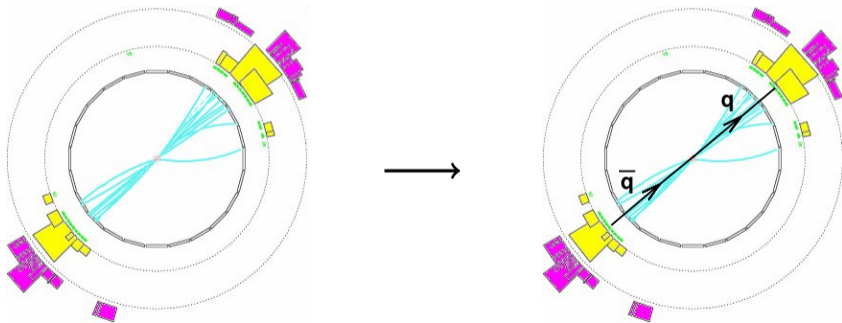
“Jets” \equiv bunch of collimated particles \cong hard partons

How many jets?



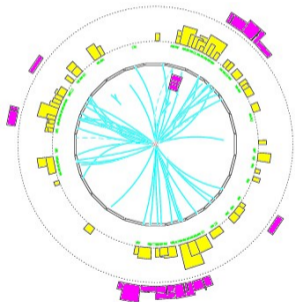
“Jets” \equiv bunch of collimated particles \cong hard partons

obviously 2 jets



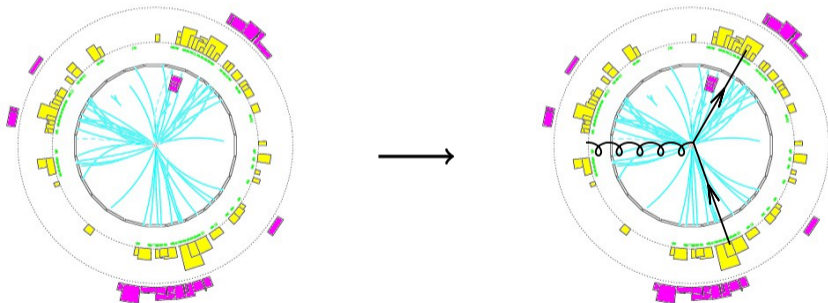
“Jets” \equiv bunch of collimated particles \cong hard partons

How many jets?



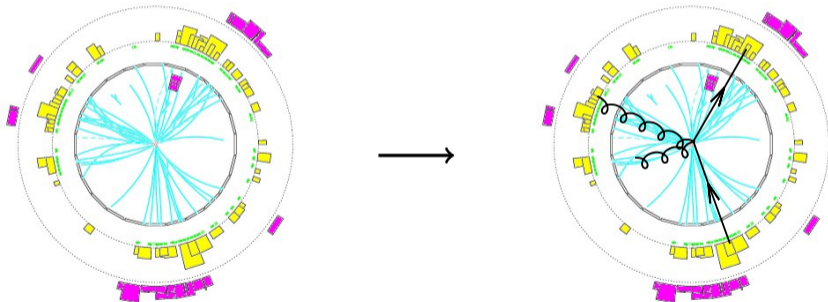
“Jets” \equiv bunch of collimated particles \cong hard partons

3 jets



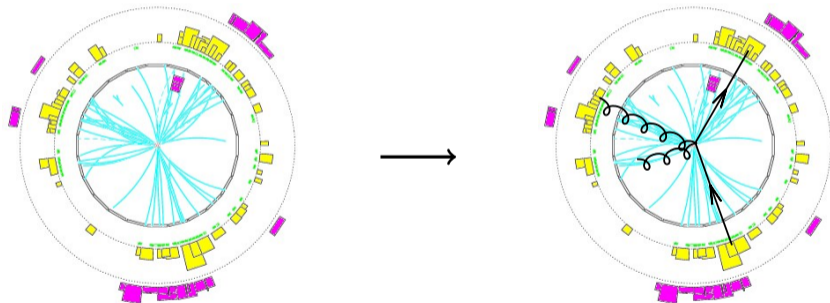
“Jets” \equiv bunch of collimated particles \cong hard partons

3 jets... or 4?



“Jets” \equiv bunch of collimated particles \cong hard partons

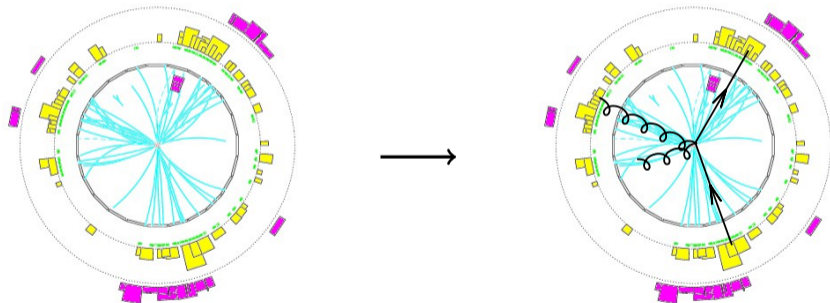
3 jets... or 4?



- “collinear” is arbitrary + “parton” concept strictly valid only at LO

“Jets” \equiv bunch of collimated particles \cong hard partons

3 jets... or 4?



- “collinear” is arbitrary + “parton” concept strictly valid only at LO
- Define jets instead