MCnet School - Lund
July 1st, 2009

Jet Definitions

Matteo Cacciari
LPTHE - Paris 6,7 and CNRS

Contains original artwork
(as well as many ideas)
from Gavin Salam

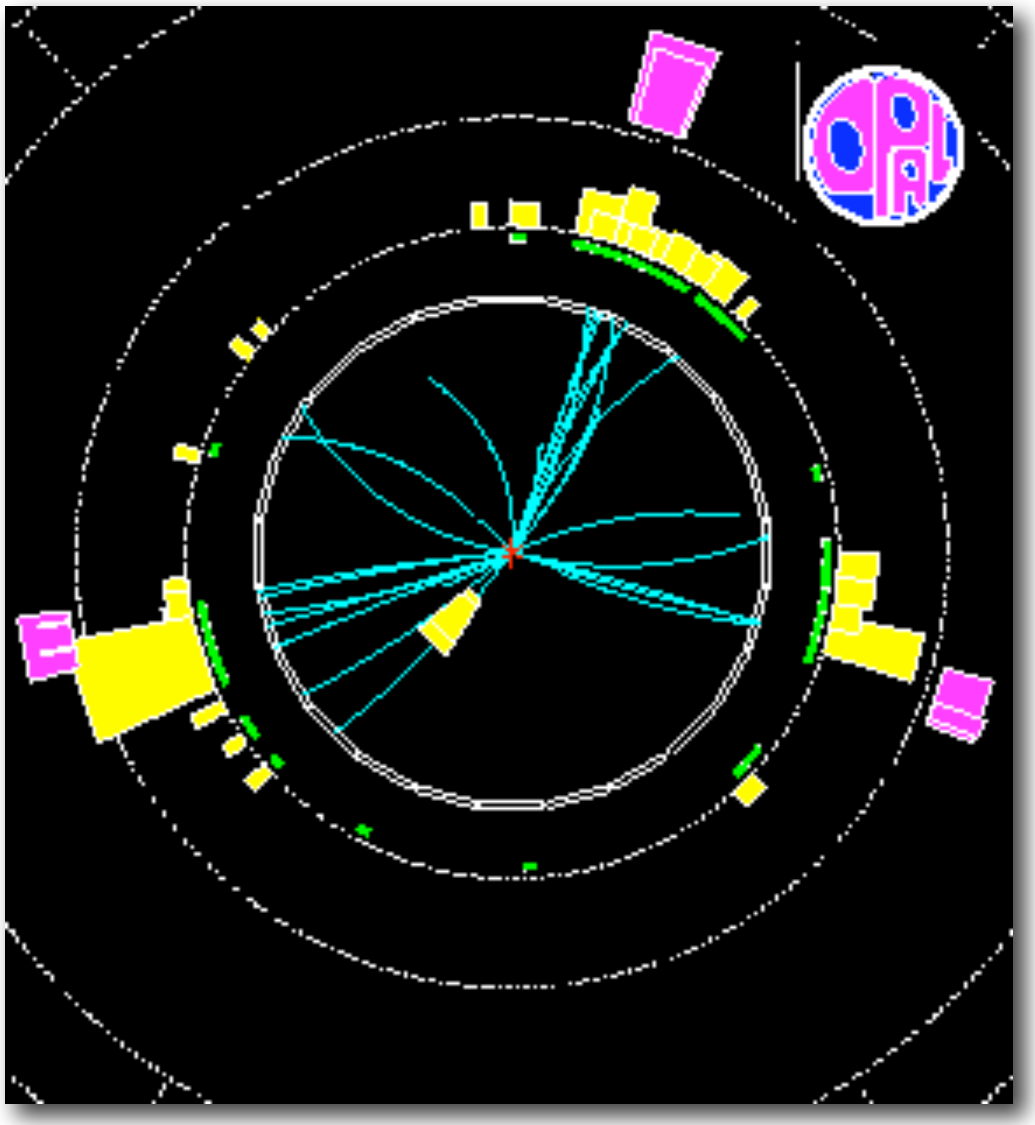
Outline

- Introductory remarks
- Jet algorithms, infrared and collinear (un)safety
  - Cone-type (progressive removal, split-merge, SISCone)
  - Recombination-type (k_t, Cambridge/Aachen, anti-k_t)
  - Third-generation algorithms (C/A-filtering)
- Jet definitions
  - Best R
  - Quality measures (mass-peak reconstruction)
- Jet areas and subtraction
- Summary

Bibliography

- Les Houches 2007 proceedings, arXiv:0803.0678
- Gavin Salam's lectures at CTEQ-MCnet 2008
- Gavin Salam, 'Towards Jetography' arXiv:0906.1833

Why jets

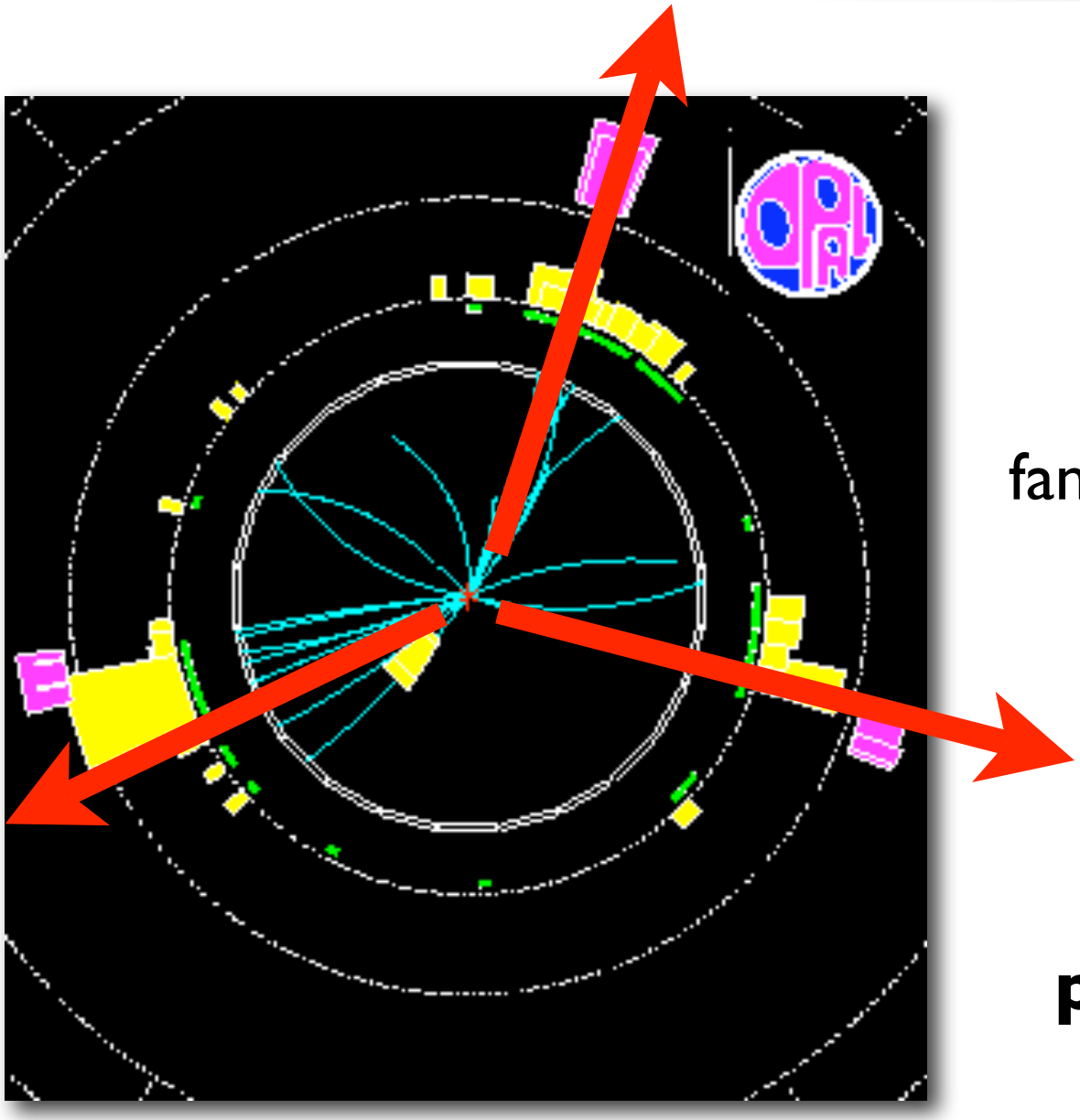


A jet is something that happens in high energy events:

a collimated bunch of hadrons flying roughly in the same direction

Note: hundreds of hadrons contain **a lot** of information. More than we can hope to make use of

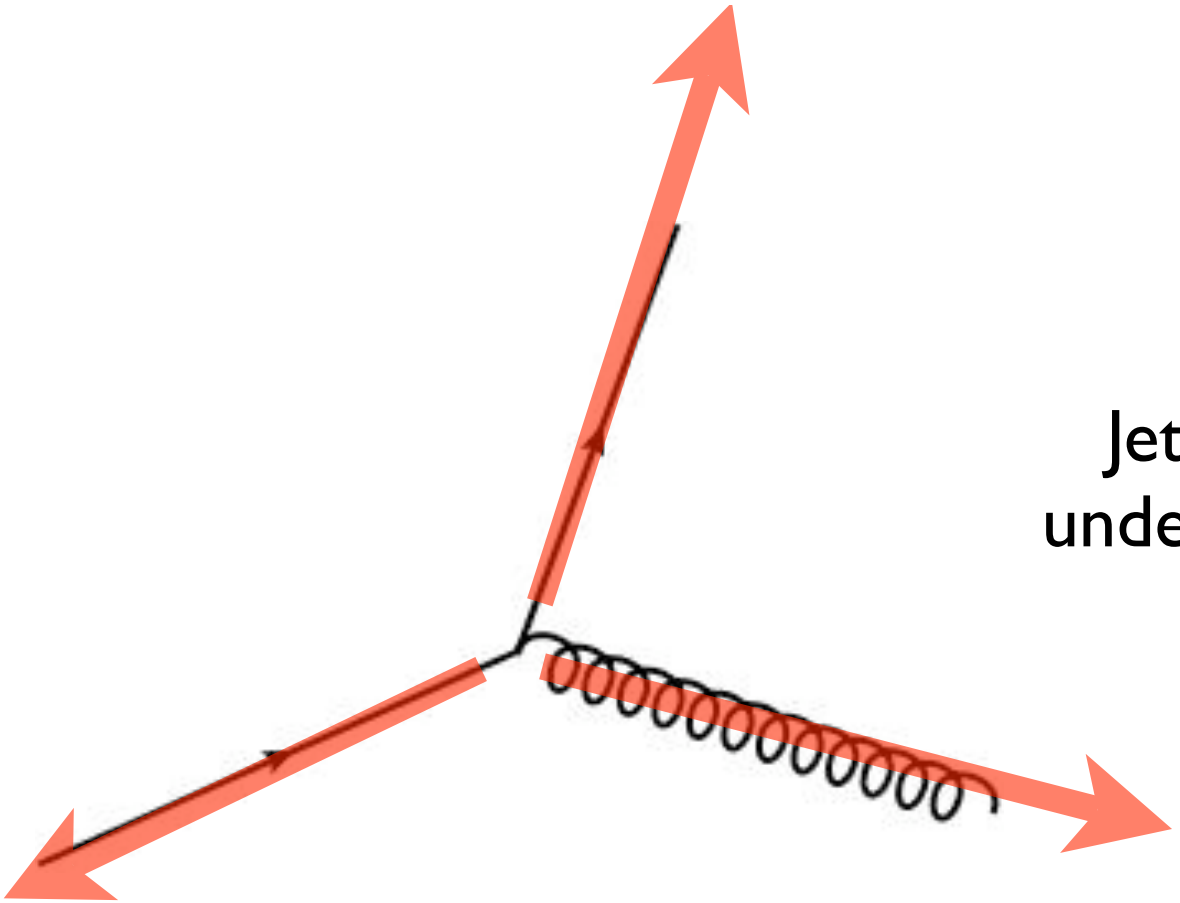
Why jets



Often you don't need a fancy algorithm to 'see' the jets

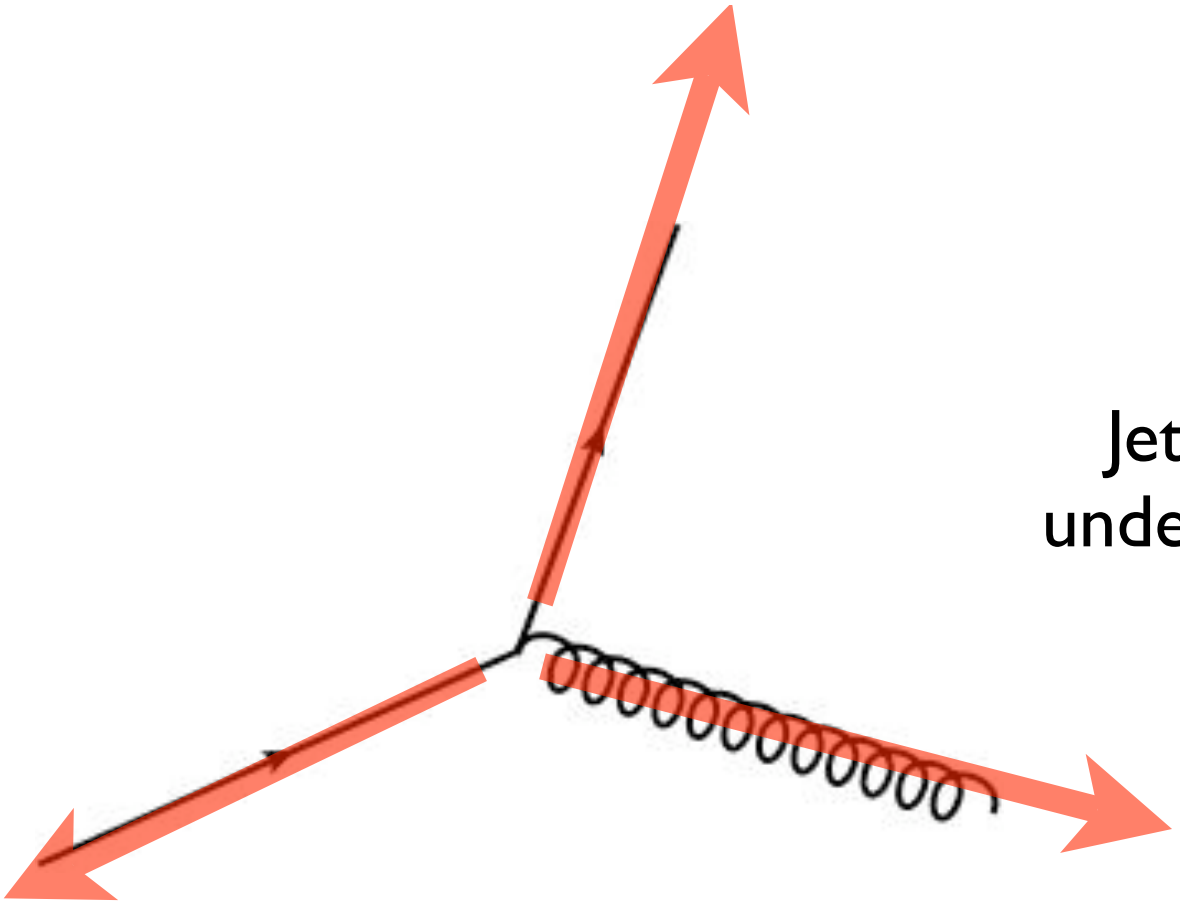
But you do to give them a **precise** and **quantitative** meaning

Why jets



Jets are usually related to an underlying perturbative dynamics (i.e. quarks and gluons)

Why jets

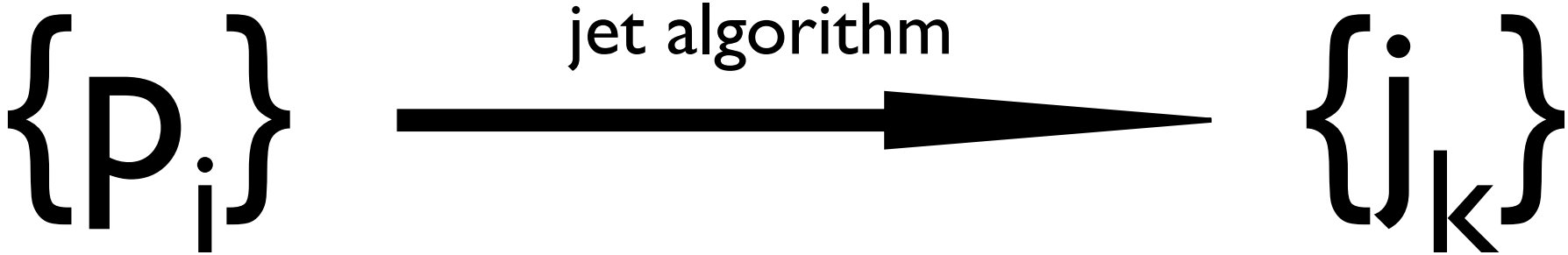


Jets are usually related to an underlying perturbative dynamics (i.e. quarks and gluons)

The purpose of a 'jet clustering' algorithm is then to **reduce the complexity** of the final state, simplifying many hadrons to **simpler objects** that one can hope to **calculate**

Jet algorithm

A **jet algorithm** maps the momenta of the final state particles into the momenta of a certain number of jets:



particles,
4-momenta,
calorimeter towers, ...

jets

Most algorithms contain a resolution parameter, **R**, which controls the extension of the jet
(more about this later on)

Two main classes of jet algorithms

Sequential recombination algorithms

bottom-up approach: combine particles starting from **closest ones**

How? Choose a **distance measure**, iterate recombination until few objects left, call them jets

Work because of mapping closeness ⇔ QCD divergence

Examples: Jade, k_t, Cambridge/Aachen, anti-k_t, …..

Two main classes of jet algorithms

Sequential recombination algorithms

bottom-up approach: combine particles starting from **closest ones**

How? Choose a **distance measure**, iterate recombination until few objects left, call them jets

Work because of mapping closeness ⇔ QCD divergence

Examples: Jade, k_t, Cambridge/Aachen, anti-k_t, …..

Cone algorithms

top-down approach: find coarse regions of energy flow.

How? Find **stable cones** (i.e. their axis coincides with sum of momenta of particles in it)

Work because QCD only modifies energy flow on small scales

Examples: JetClu, MidPoint, ATLAS cone, CMS cone, …...

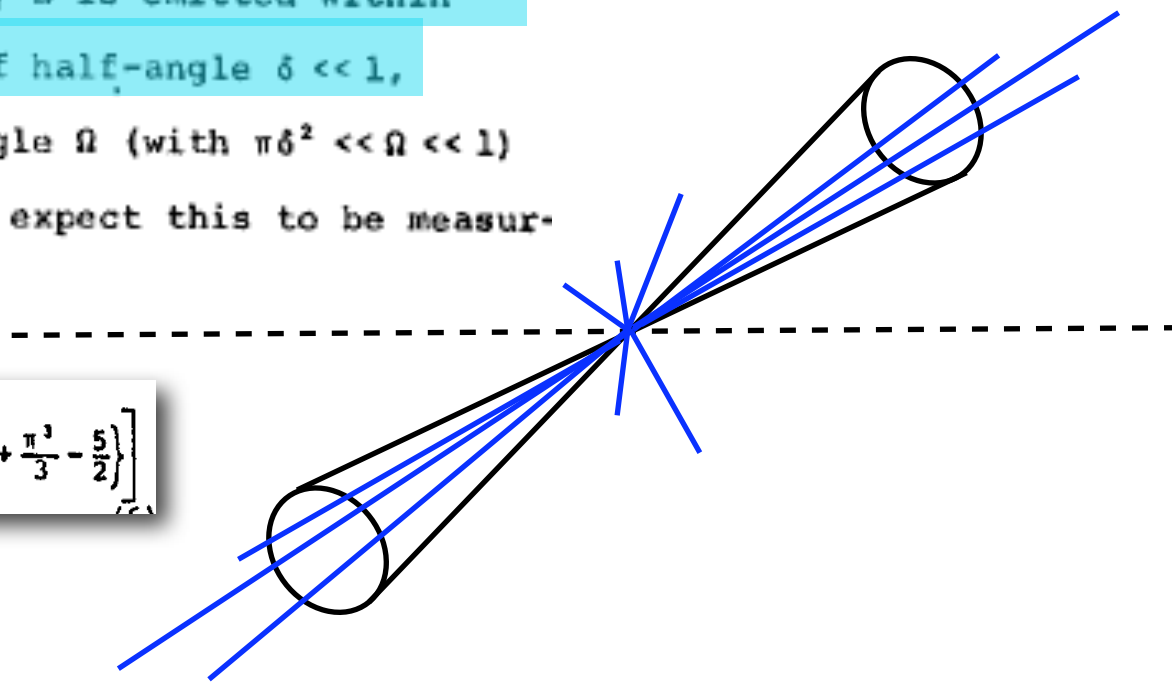
Cone algorithms

The first rigorous definition of cone jets 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 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-

$$\sigma(E,\theta,\Omega,\epsilon,\delta) = (d\sigma/d\Omega)_0\,\Omega\left[1 - (g_E^2/3\pi^2)\left\{3\ln\delta + 4\ln\delta\,\ln 2\epsilon + \frac{\pi^3}{3} - \frac{5}{2}\right\}\right] \quad (6)$$



Good for 2 jets and e^+e^- collisions

In more general cases, **where** do we place the cones? **How many**?

Finding cones

Different procedures for placing the cones lead to **different cone algorithms**

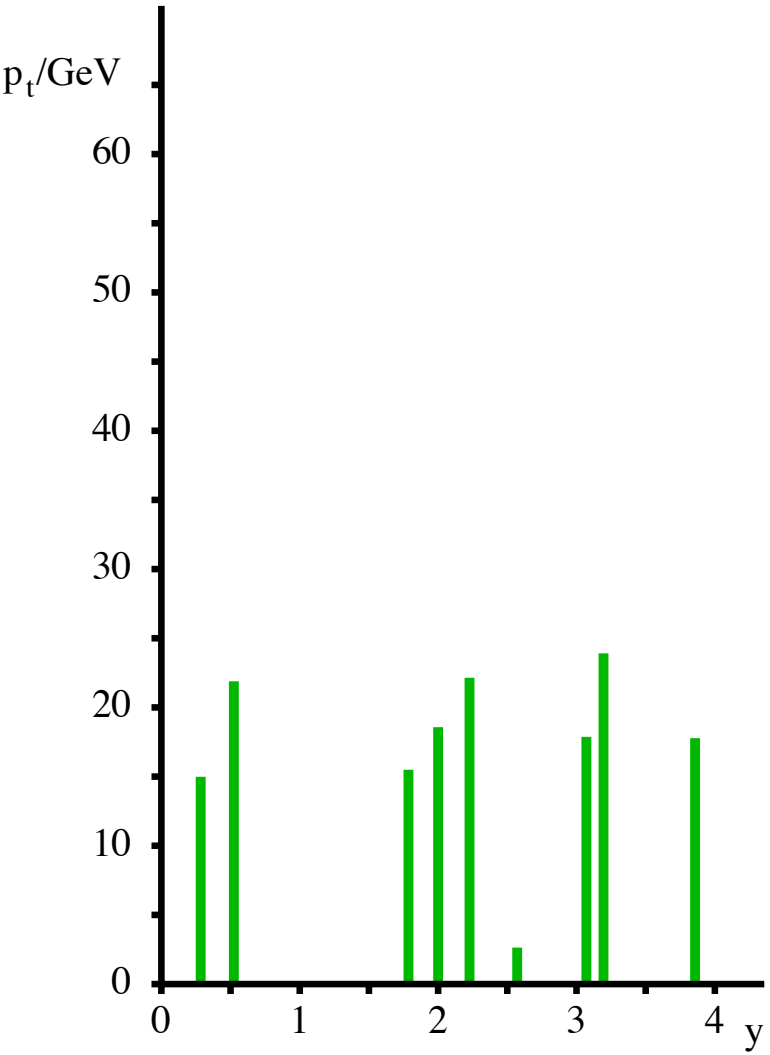
NB: their properties and behaviour can **vastly differ**:
there isn't **'a'** cone algorithm, but rather many of them

The main sub-categories of cone algorithms are:

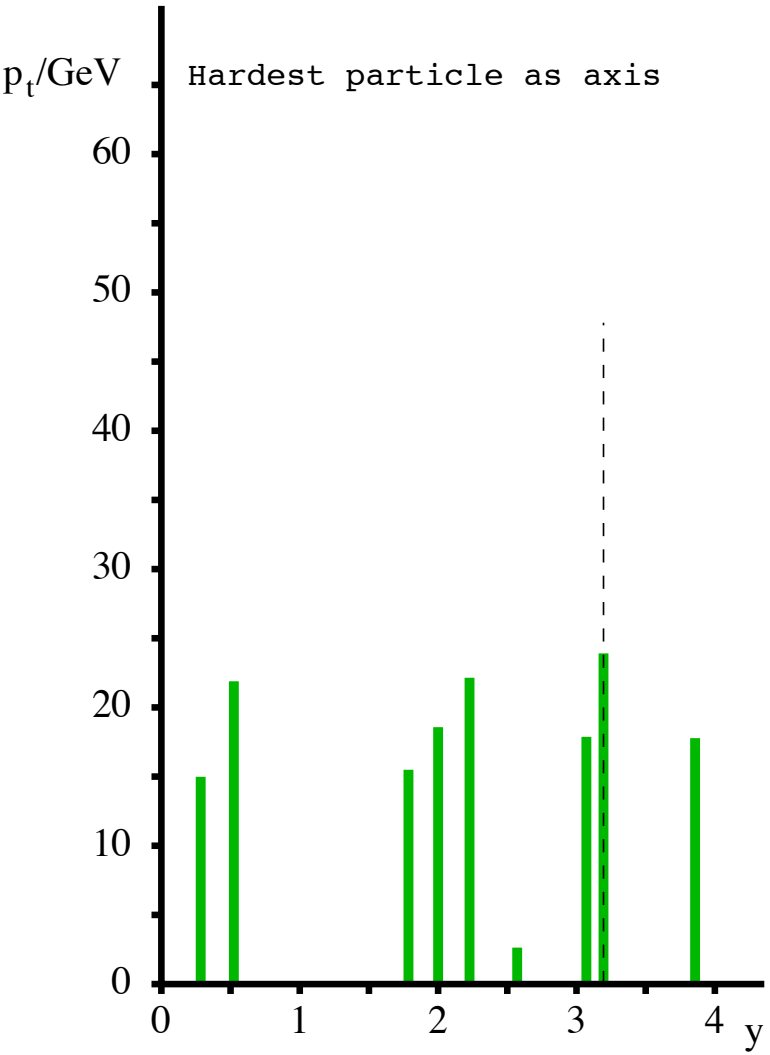
- **Fixed** cone with **progressive removal** (FC-PR) (PyJet, CellJet, GetJet)
- **Iterative** cone with **progressive removal** (IC-PR) (CMS iterative cone)
- **Iterative** cone with **split-merge** (IC-SM) (JetClu, ATLAS cone)
- **IC-SM** with **mid-points** (IC_{mp}-SM) (CDF MidPoint, D0 Run II)
- **IC_{mp}** with **split-drop** (IC_{mp}-SD) (PxCone)
- **Seedless** cone with **split-merge** (SC-SM) (SISCone)

Fixed Cone, Progressive Removal (FC-PR)

Probably the simplest cone algorithm



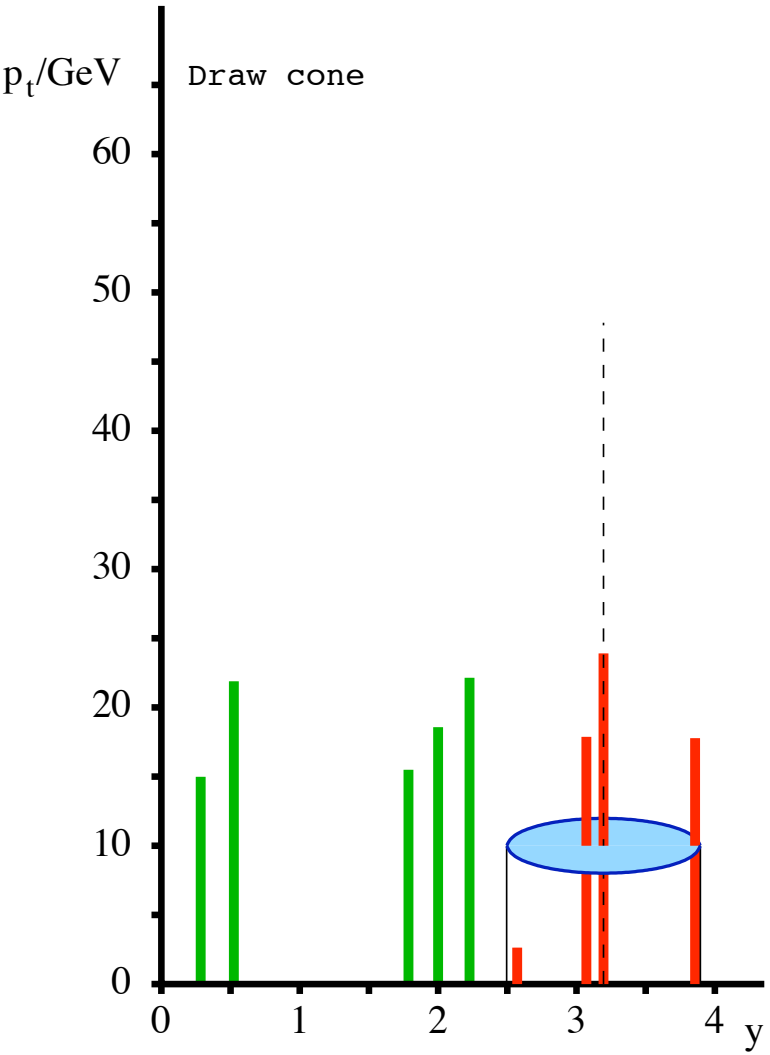
Fixed Cone, Progressive Removal (FC-PR)



Probably the simplest cone algorithm

Choose hardest particle as seed

Fixed Cone, Progressive Removal (FC-PR)

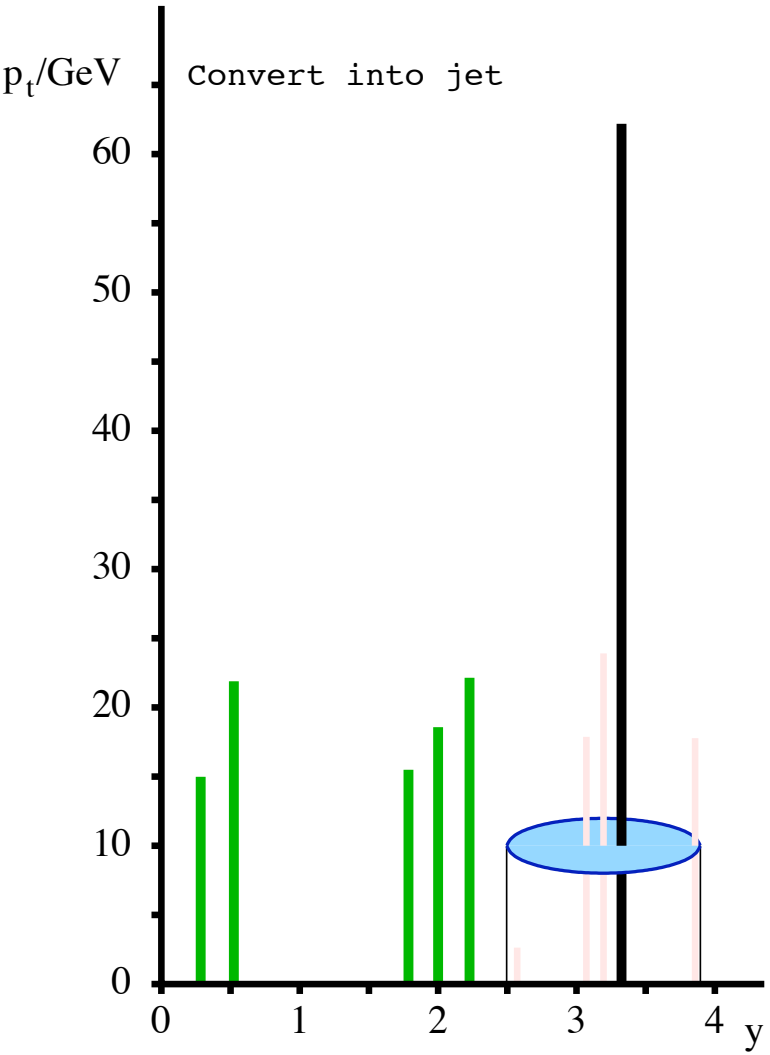


Probably the simplest cone algorithm

Choose hardest particle as seed

Draw cone around it

Fixed Cone, Progressive Removal (FC-PR)



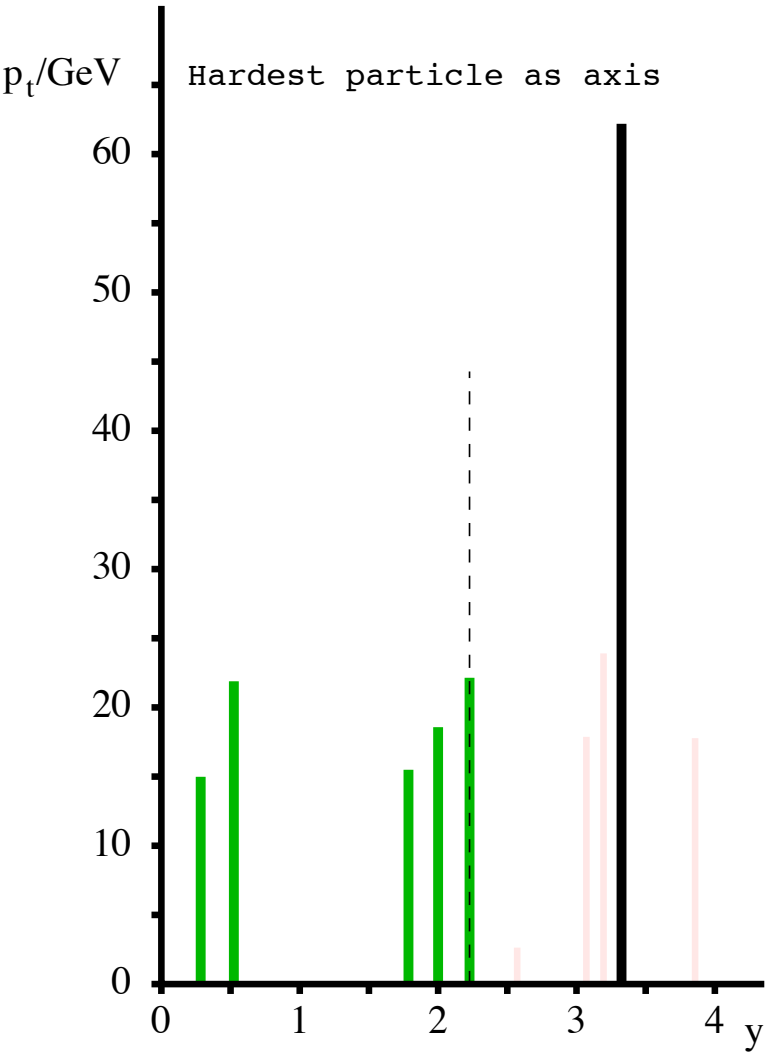
Probably the simplest cone algorithm

Choose hardest particle as seed

Draw cone around it

Call it a jet, remove constituents from set of particles

Fixed Cone, Progressive Removal (FC-PR)



Probably the simplest cone algorithm

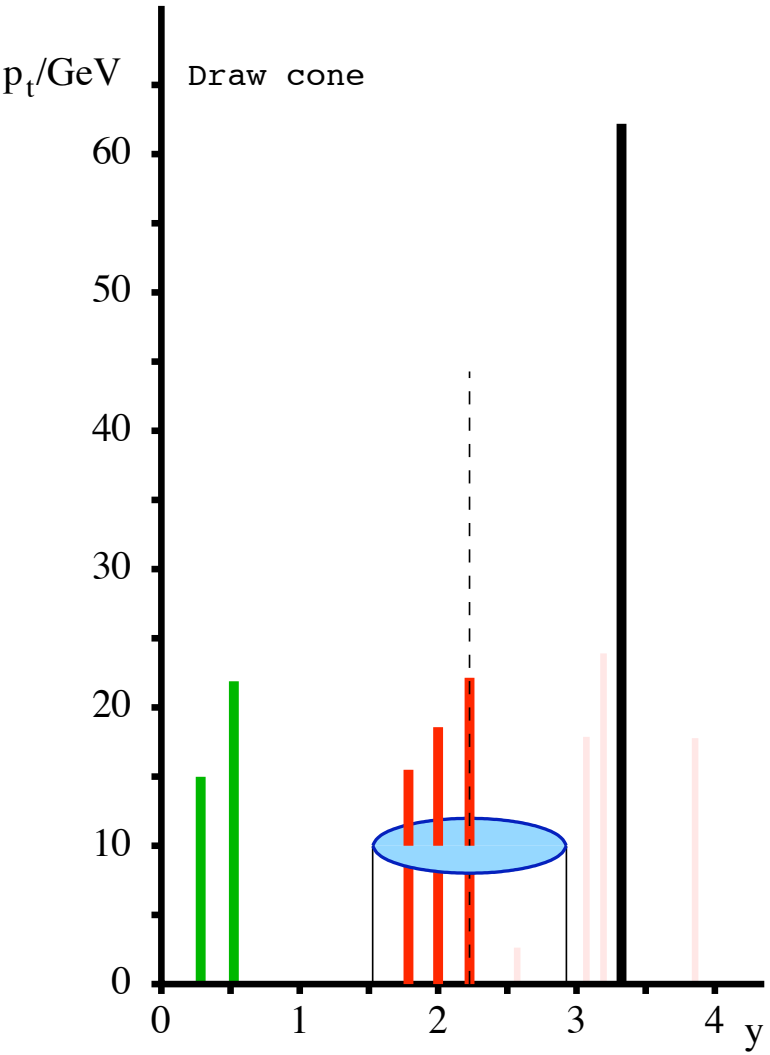
Choose hardest particle as seed

Draw cone around it

Call it a jet, remove constituents from set of particles

Repeat using hardest particle left

Fixed Cone, Progressive Removal (FC-PR)



Probably the simplest cone algorithm

Choose hardest particle as seed

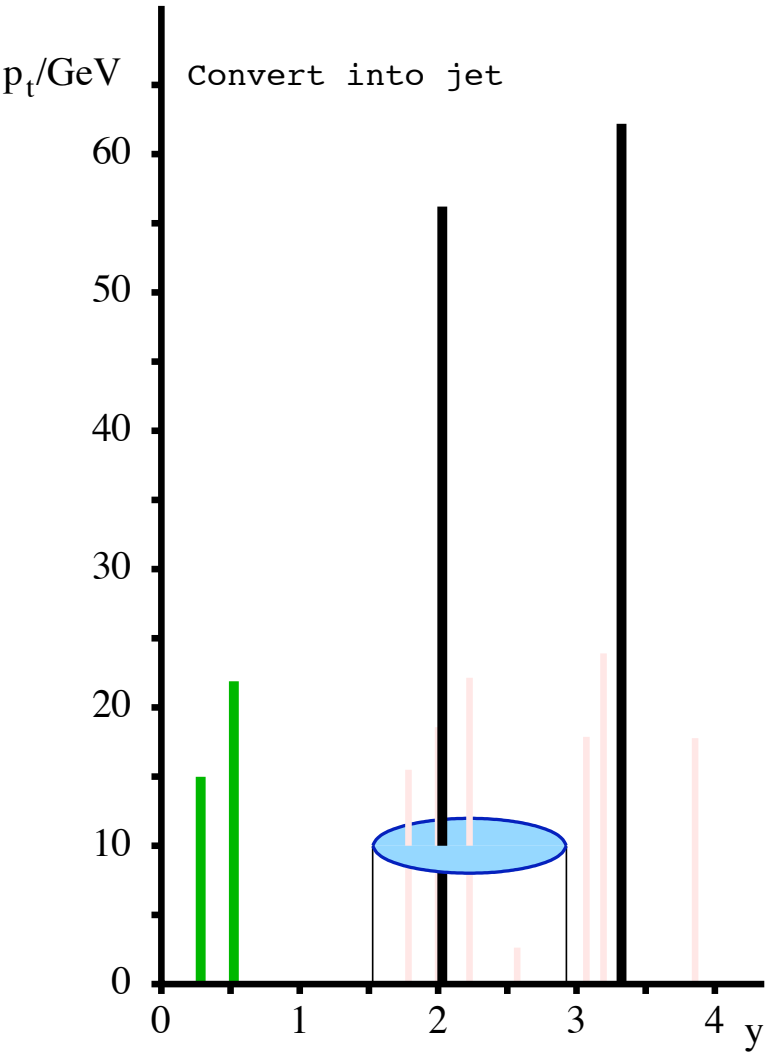
Draw cone around it

Call it a jet, remove constituents from set of particles

Repeat using hardest particle left

Etc, etc

Fixed Cone, Progressive Removal (FC-PR)



Probably the simplest cone algorithm

Choose hardest particle as seed

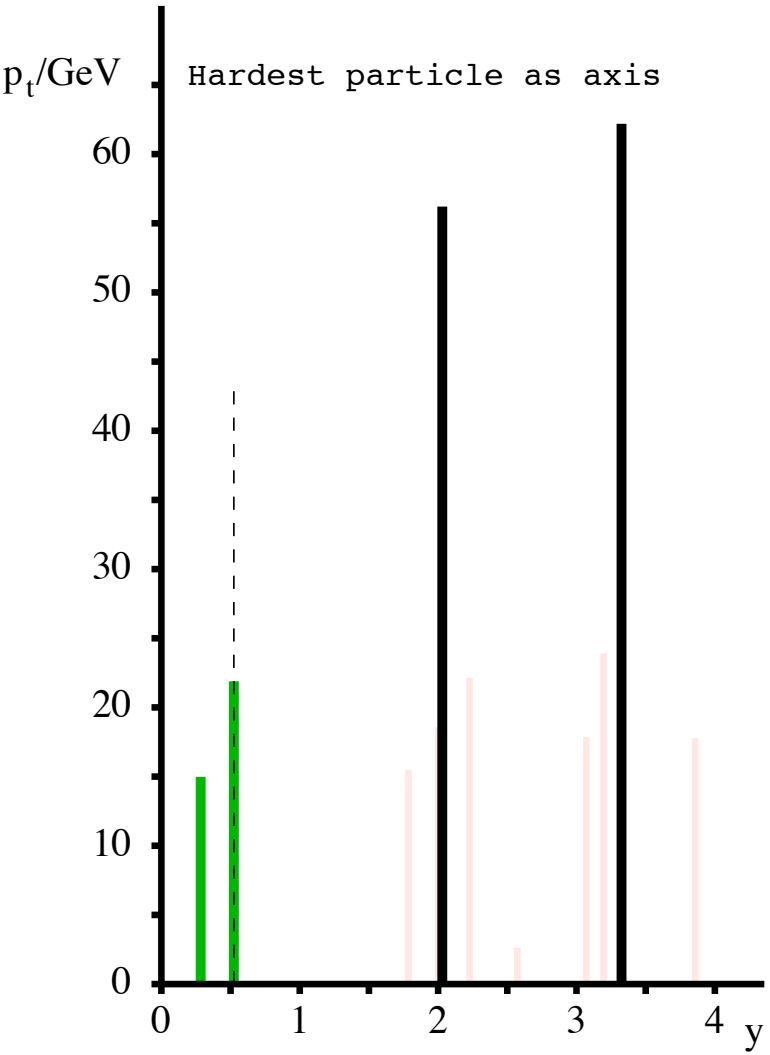
Draw cone around it

Call it a jet, remove constituents from set of particles

Repeat using hardest particle left

Etc, etc

Fixed Cone, Progressive Removal (FC-PR)



Probably the simplest cone algorithm

Choose hardest particle as seed

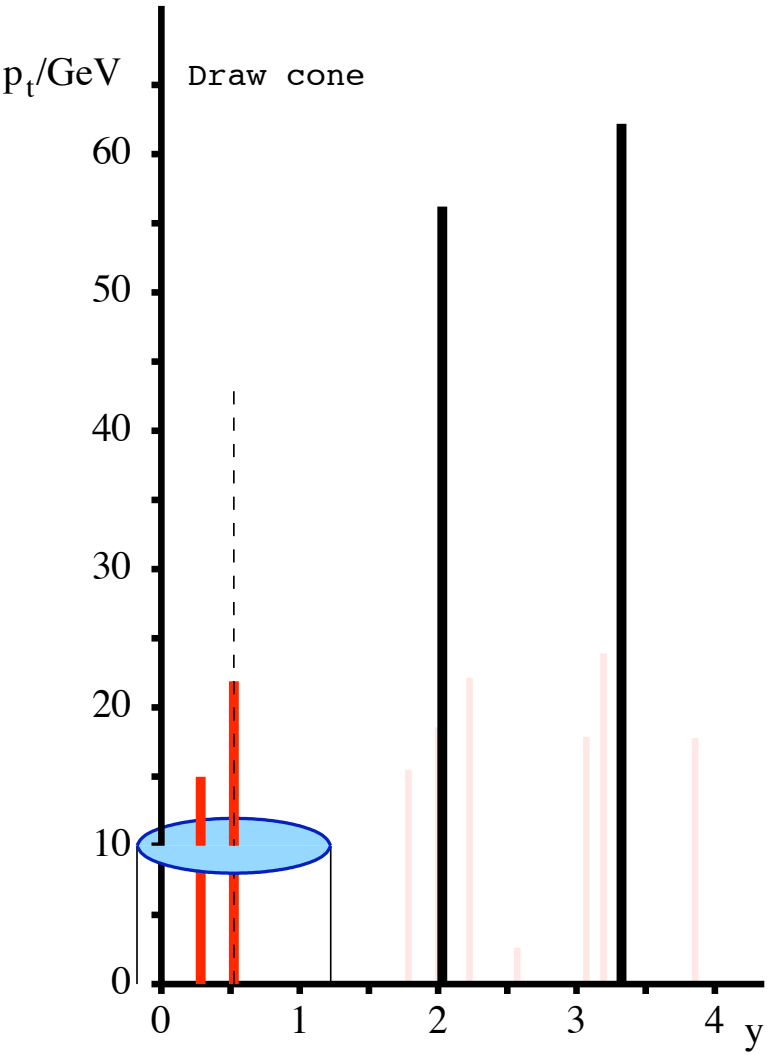
Draw cone around it

Call it a jet, remove constituents from set of particles

Repeat using hardest particle left

Etc, etc

Fixed Cone, Progressive Removal (FC-PR)



Probably the simplest cone algorithm

Choose hardest particle as seed

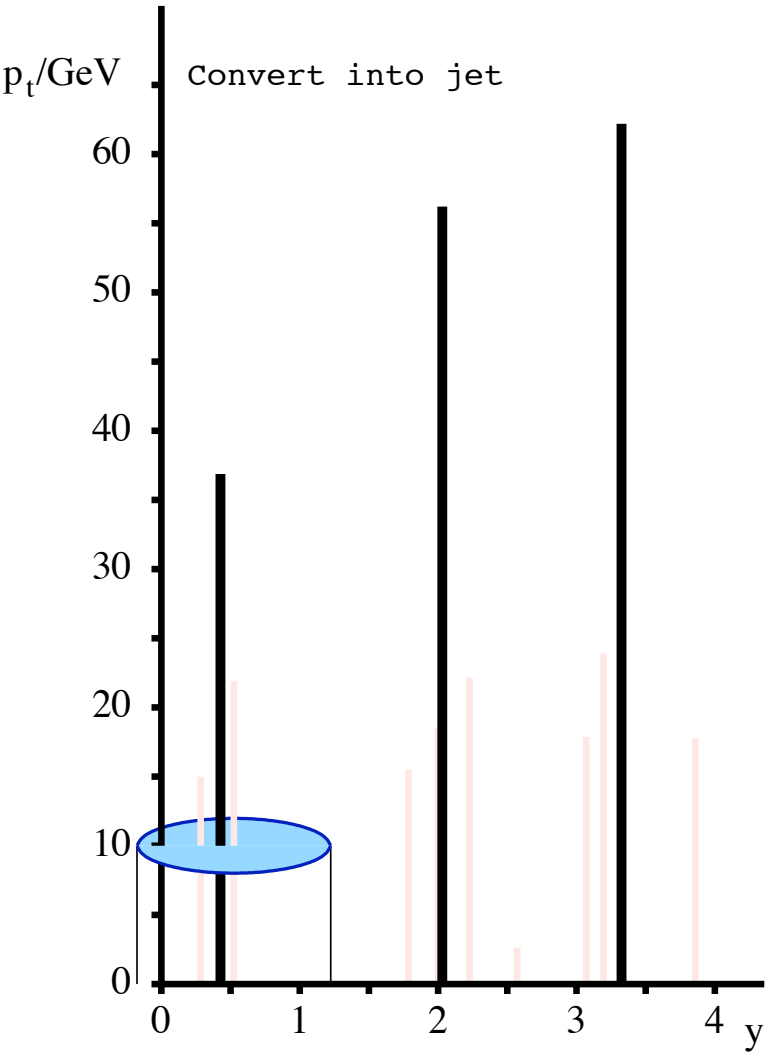
Draw cone around it

Call it a jet, remove constituents from set of particles

Repeat using hardest particle left

Etc, etc

Fixed Cone, Progressive Removal (FC-PR)



Probably the simplest cone algorithm

Choose hardest particle as seed

Draw cone around it

Call it a jet, remove constituents from set of particles

Repeat using hardest particle left

Etc, etc

Until no particles left

FC-PR v. IC-PR

Seed and cone axis may **not coincide**.
Making them do can lead to different jets

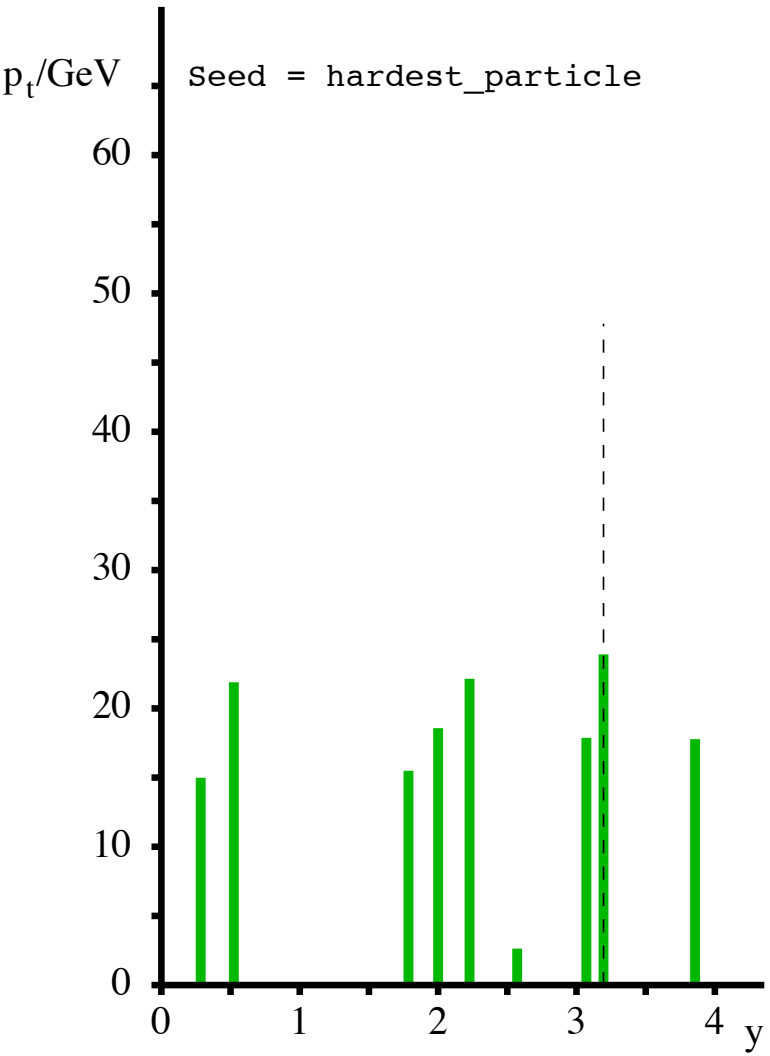
FC-PR v. IC-PR

Seed and cone axis may **not coincide**.
Making them do can lead to different jets

Let us try an **Iterative** Cone with Progressive Removal (IC-PR)
(e.g. the CMS Iterative Cone)

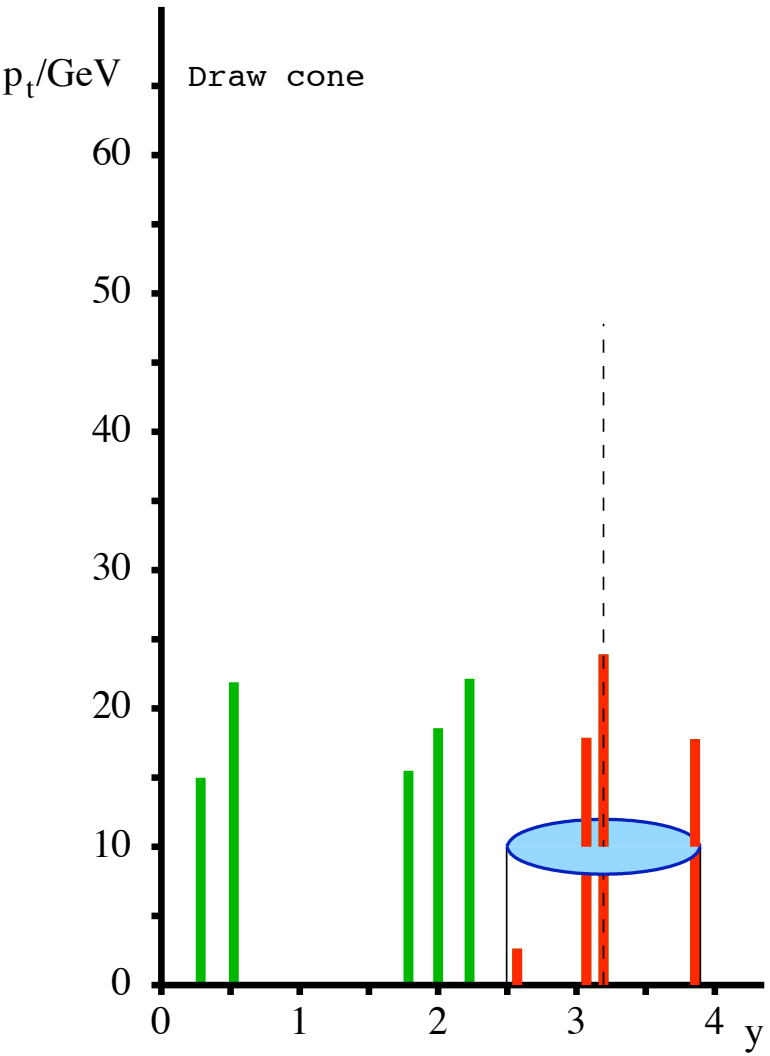
- Begin with **hardest particle** as seed
- Cluster particles into cone if $\Delta R < R$
- **Iterate** until stable (i.e. axis coincide with sum of momenta) cones found
- Eliminate constituents of jet and start over from hardest remaining particle

FC-PR v. IC-PR



Choose hardest particle as seed

FC-PR v. IC-PR

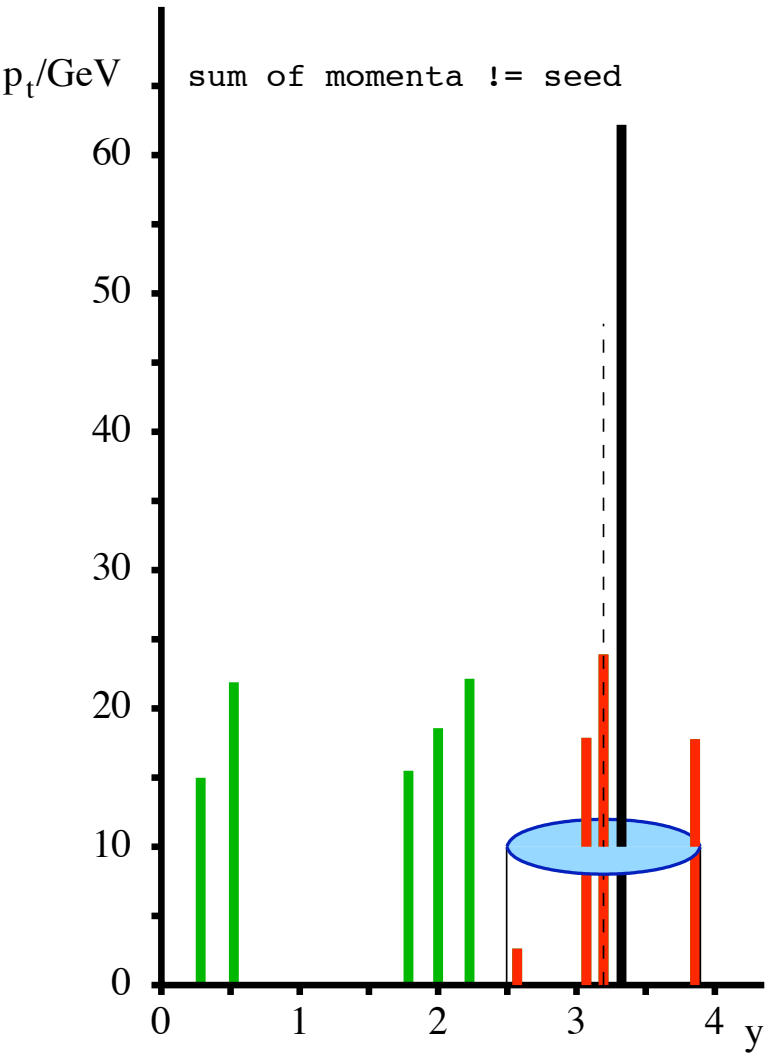


Choose hardest particle as seed

Draw cone around it

So far, identical to FC-PR

FC-PR v. IC-PR

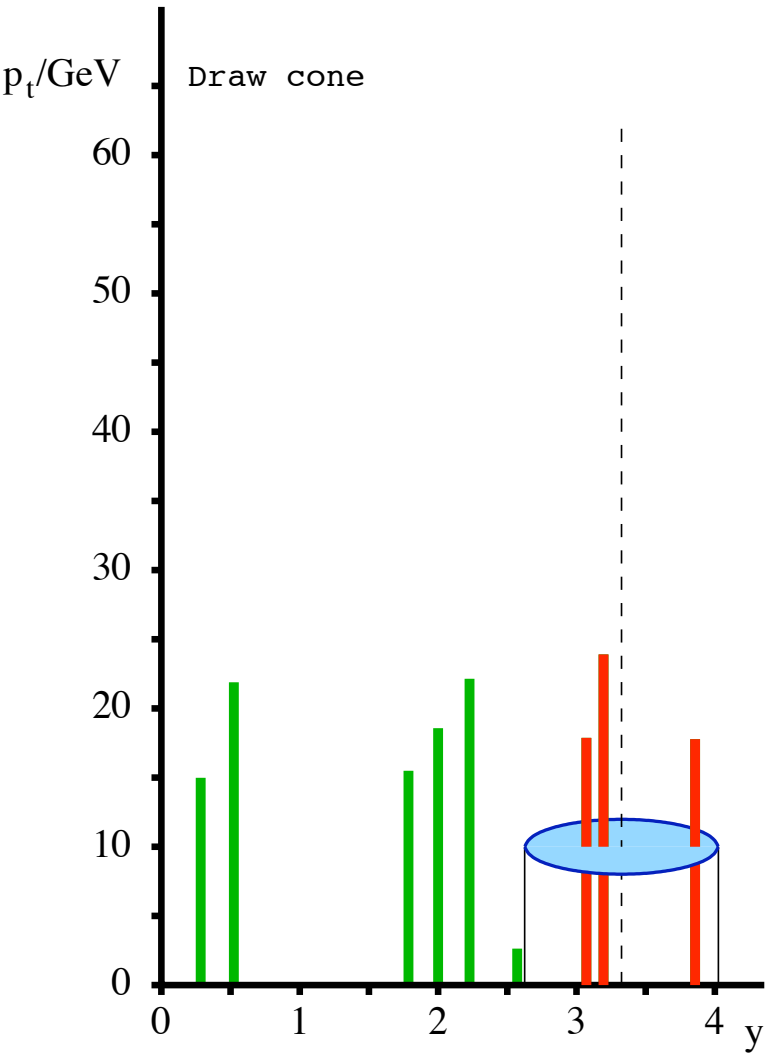


Choose hardest particle as seed

Draw cone around it

Jet axis not centred on seed

FC-PR v. IC-PR



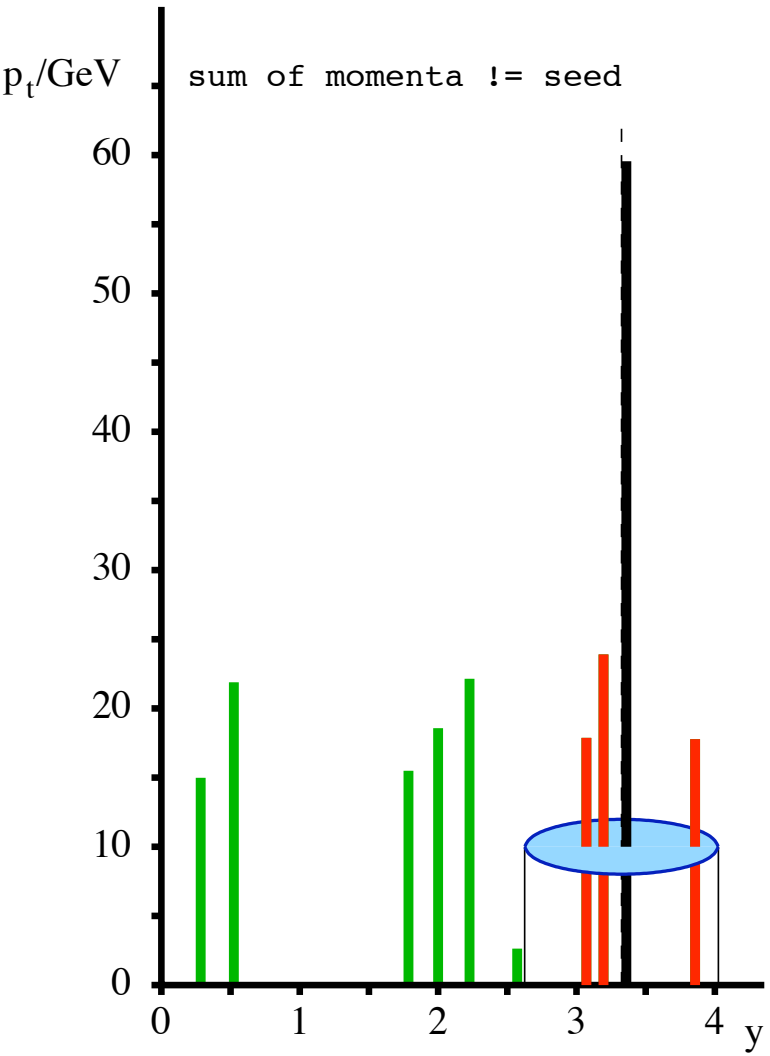
Choose hardest particle as seed

Draw cone around it

Jet axis not centred on seed

Redraw cone around new axis

FC-PR v. IC-PR



Choose hardest particle as seed

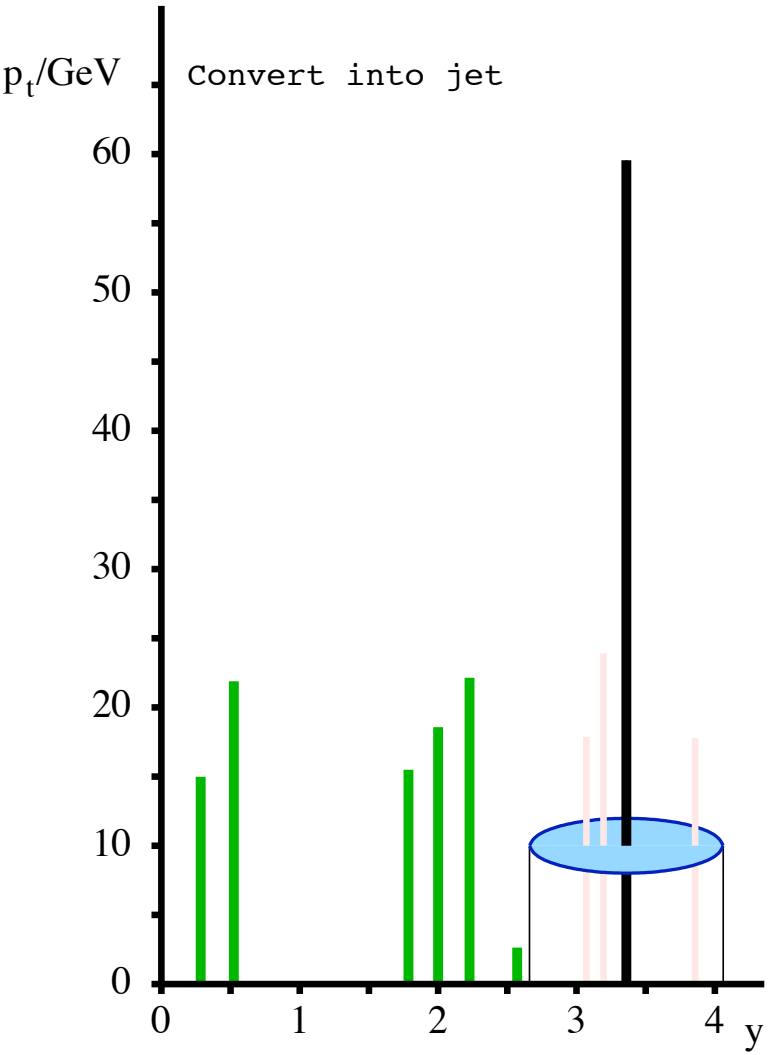
Draw cone around it

Jet axis not centred on seed

Redraw cone around new axis

Still, jet axis not centred on seed

FC-PR v. IC-PR



Choose hardest particle as seed

Draw cone around it

Jet axis not centred on seed

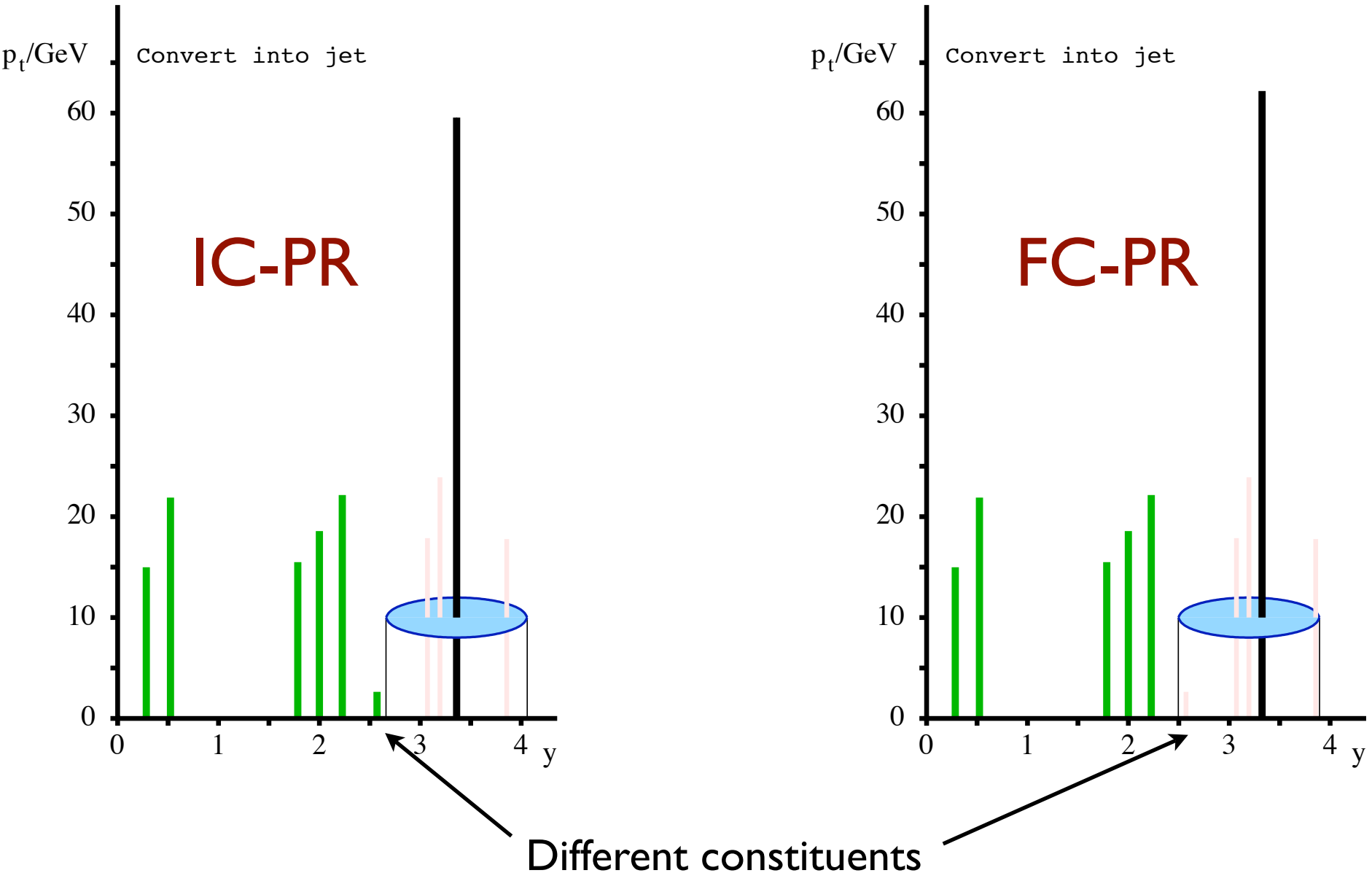
Redraw cone around new axis

Still, jet axis not centred on seed

Repeat until it is, finally get to this

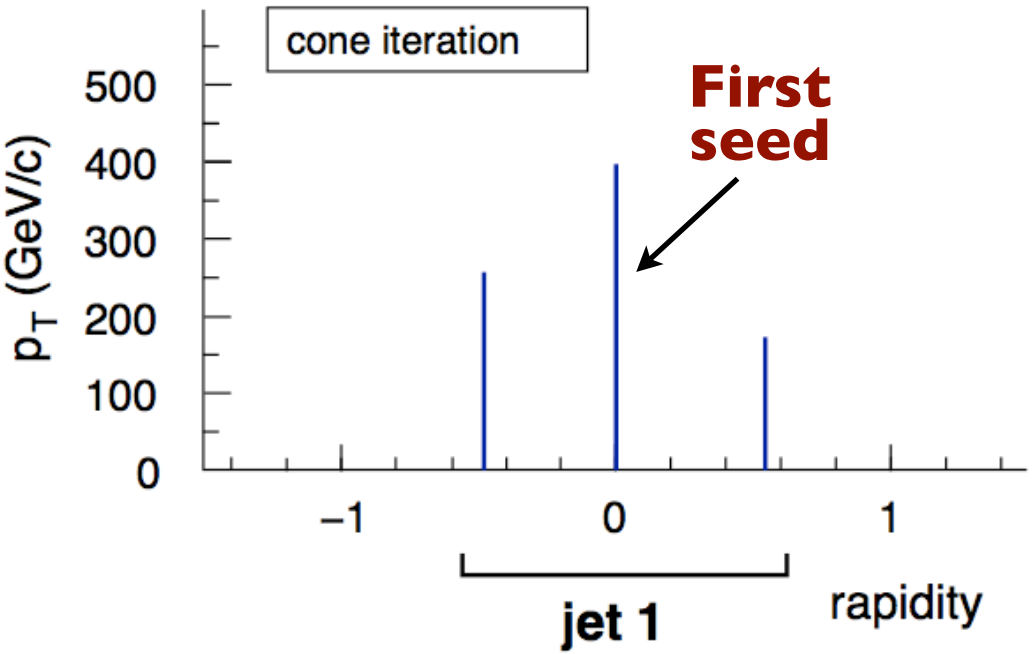
FC-PR v. IC-PR

This jet differs from the corresponding FC-PR one



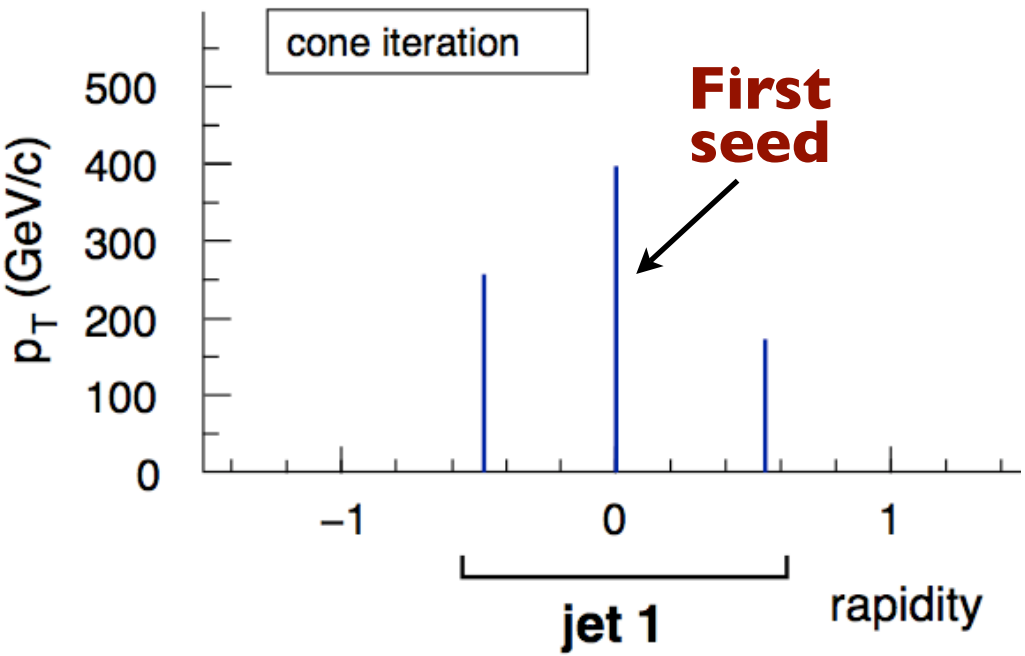
IC-PR cone collinear unsafety

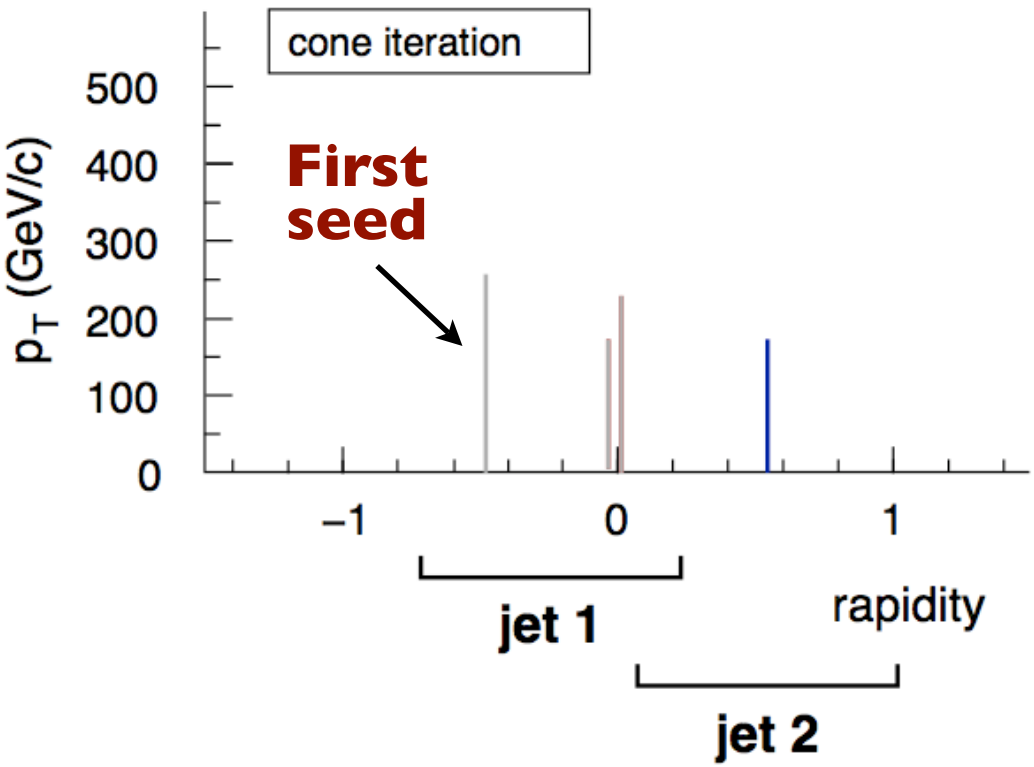
A collinear splitting can change the final state



IC-PR cone collinear unsafety

A collinear splitting can change the final state



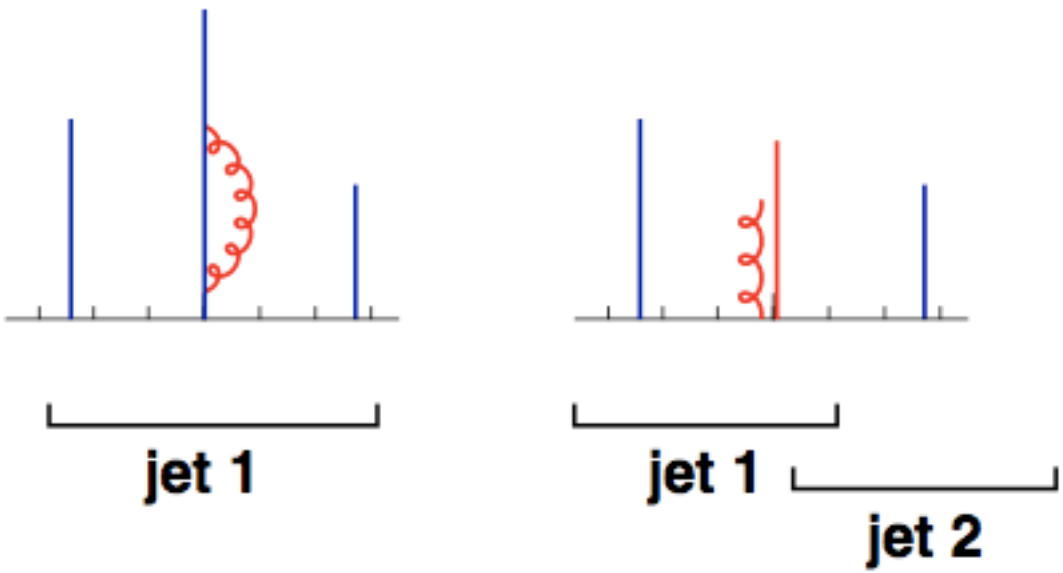


Splitting the hardest particle **collinearly** has changed the number of final jets

Consequences of collinear unsafety

In QCD perturbation theory, virtual and real configurations can only cancel if they lead to the **same** final state

In this example with IC-PR, we have seen that the final state can differ:



⇒ no cancellation of divergencies, no convergence of perturbation theory

Jet algorithms using hardest particles as seeds will generally be susceptible to collinear unsafety

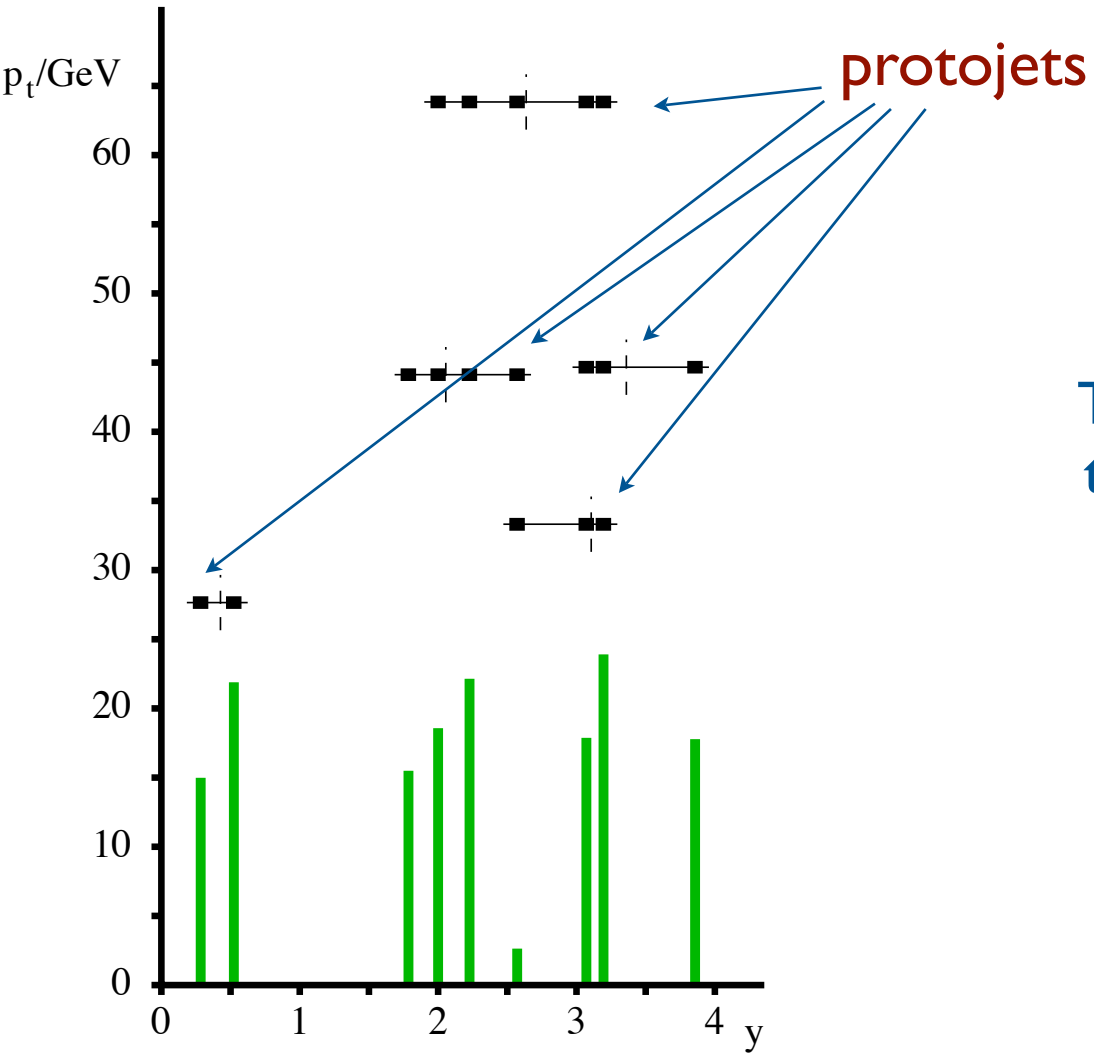
Iterative Cone with Split-Merge (IC-SM)

Choosing hardest particles as seed was an issue (collinear unsafety).
Let us therefore try taking **all particles**

- Use **all particles** as seed
- Cluster particles into cone if $\Delta R < R$
- Iterate until stable (i.e. axis coincide with sum of momenta) cones found
- Split-merge step (see later on)

Examples of this algorithm are JetClu and the ATLAS Cone

Iterating the cones over all particles as seeds returns 5 stable protojets



The lack of 'progressive removal' means that some protojets can be overlapping (i.e. contain the same particles).
Must deal with this: **split-merge**

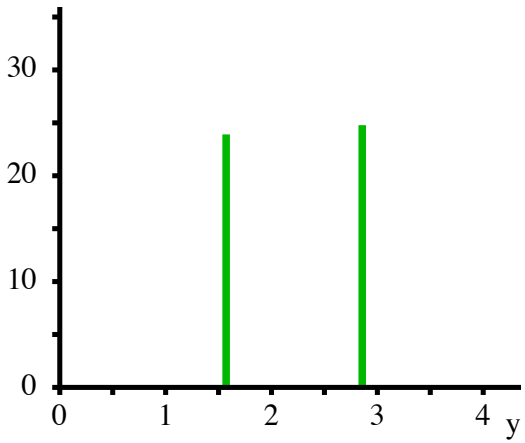
Split-Merge

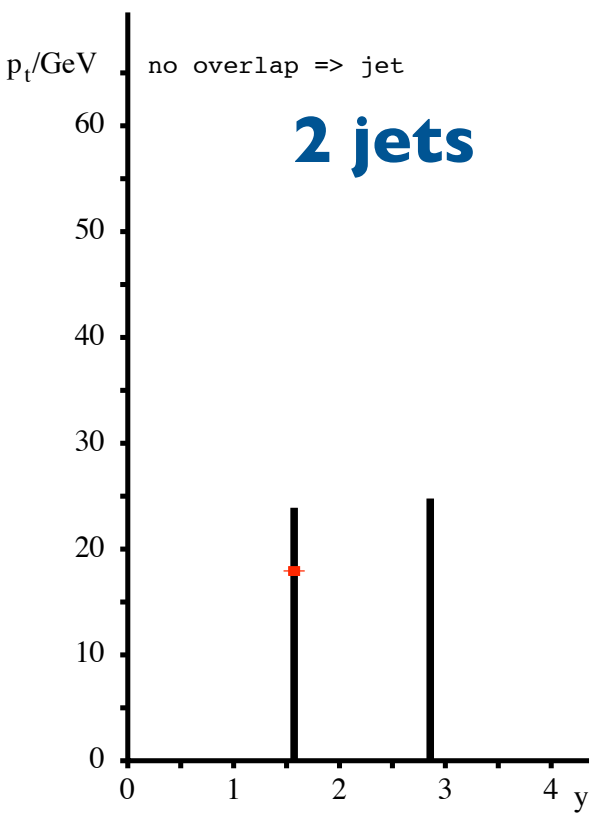
'Split-merge' is a further algorithm aimed at disentangling overlapping protojets.

The Tevatron Run II implementation goes like this:

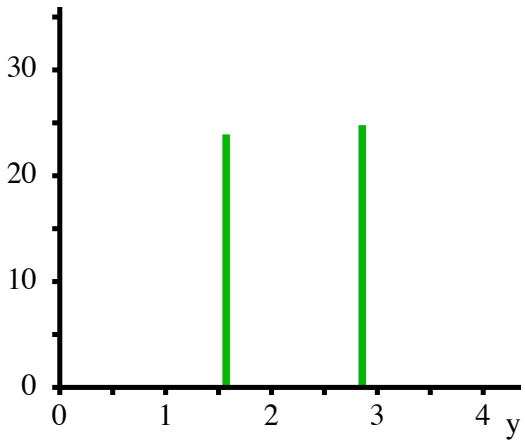
- Choose an **overlap threshold *f***
- Find hardest protojet
- Find hardest other protojet overlapping with it
- Merge is they share a fraction of momentum larger than *f*, split along axis at centre otherwise
- (Call protojet a jet if there are no overlapping protojets)

IC-SM infrared unsafety

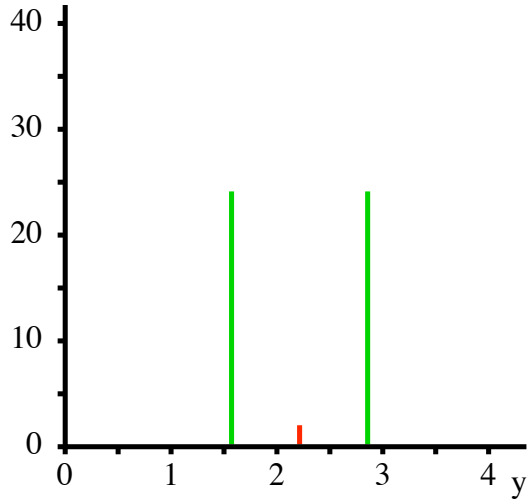


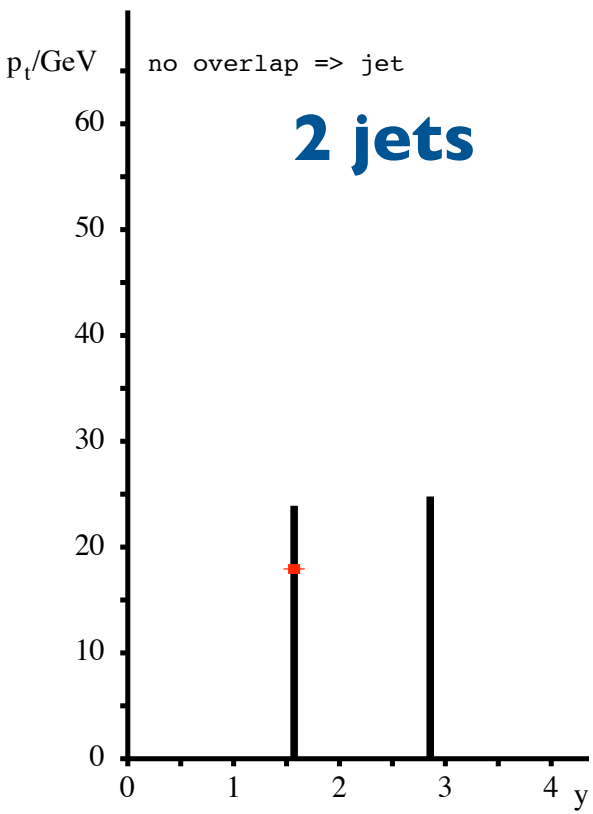


IC-SM infrared unsafety

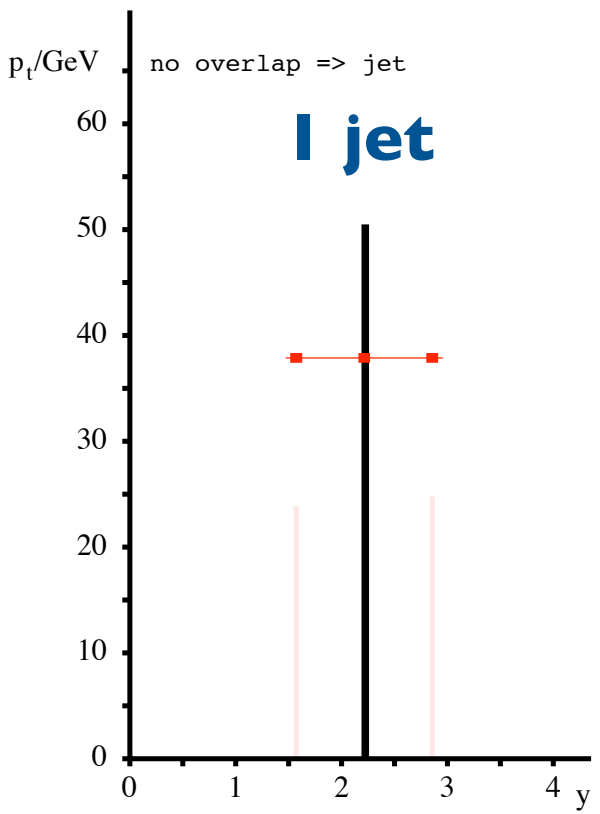


Add a
soft particle





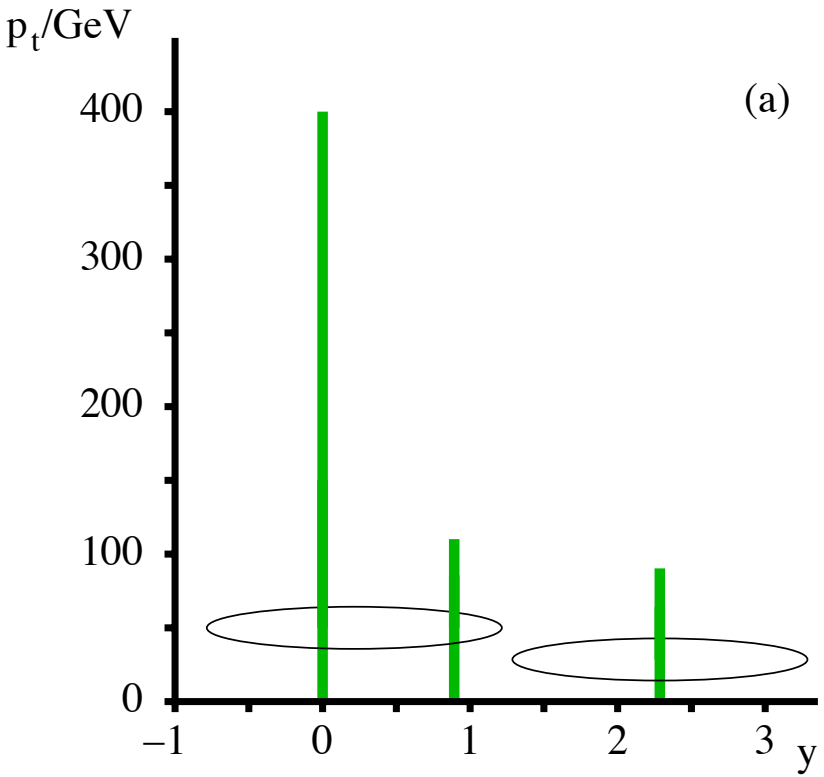
Final state jets differ



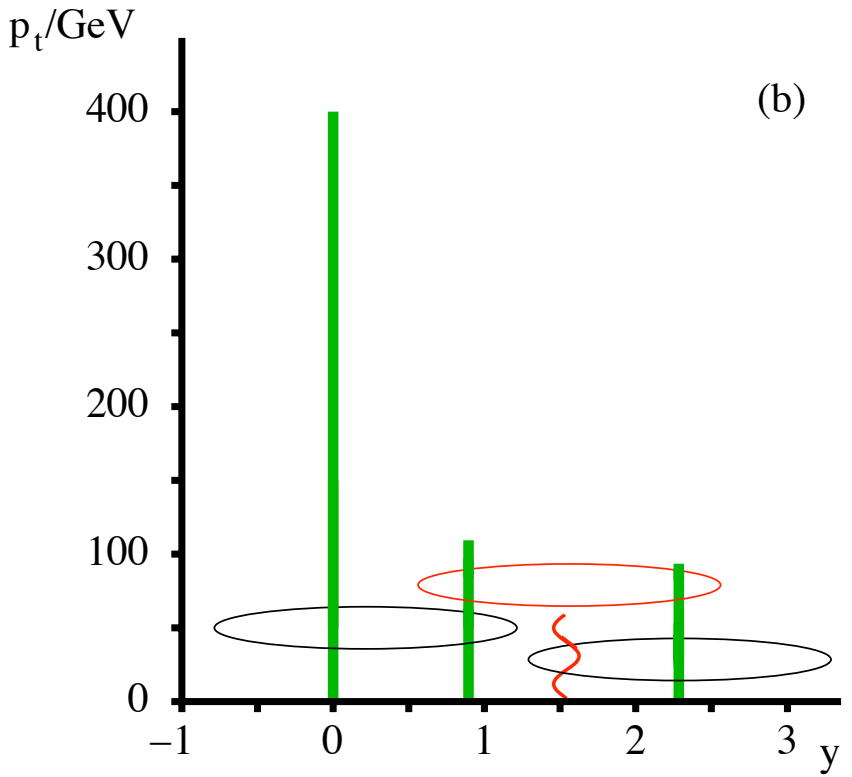
MidPoint (IC_{mp}-SM) infrared unsafety

MidPoint fixes the two-particle configuration IR-safety problem by **adding midpoints** to list of seeds.

But this merely **shifts the problem to three-particle configurations**



Three hard particles clustered into **two** cones by the MidPoint algorithm



Addition of a **soft** particle changes the hard jets: **three** stable cones are now found

The problem is that the stable-cone search procedure used by seeded IC algorithms often cannot find **all** possible stable cones

A long list of cones (all eventually unsafe)

Les Houches 2007 proceedings, arXiv:0803.0678

'First-generation' algorithms

CDF JetClu	IC_r-SM	IR_{2+1}
CDF MidPoint cone	IC_{mp}-SM	IR_{3+1}
CDF MidPoint searchcone	$IC_{se,mp}$-SM	IR_{2+1}
D0 Run II cone	IC_{mp}-SM	IR_{3+1}
ATLAS Cone	IC-SM	IR_{2+1}
PxCone	IC_{mp}-SD	IR_{3+1}
CMS Iterative Cone	IC-PR	$Coll_{3+1}$
PyCell/CellJet (from Pythia)	FC-PR	$Coll_{3+1}$
GetJet (from ISAJET)	FC-PR	$Coll_{3+1}$

IC = Iterative Cone
SM = Split-Merge
SD = Split-Drop
FC = Fixed Cone
PR = Progressive Removal

type of algorithm

safety issue

IR_{n+1} : unsafe when a soft particle is added to n hard particles in a common neighbourhood

$Coll_{n+1}$: unsafe when one of n hard particles in a common neighbourhood is split collinearly

IRC safety does matter

The best cones seen so far fail at (3+1) partons, others already at (2+1)

	Last meaningful order			
	JetClu, ATLAS cone [IC-SM]	MidPoint [IC$_{mp}$-SM]	CMS it. cone [IC-PR]	Known at
Inclusive jets	LO	NLO	NLO	NLO ($\rightarrow$ NNLO)
W/Z + 1 jet	LO	NLO	NLO	NLO
3 jets	**none**	LO	LO	NLO [nlojet++]
W/Z + 2 jets	**none**	LO	LO	NLO [MCFM]
m_{jet} in $2j + X$	**none**	**none**	**none**	LO

Calculations cost real money: ~ 100 theorists ×15 years ≈100 M€

Using unsafe jet tools essentially renders them useless

IRC safety in real life

Strictly speaking, one needs IRC safety not so much to find jets, but to be able to calculate them in pQCD

If you are not interested in thory/data comparisons, you may think of doing well enough with an IRC-unsafe jet algorithm

However

- Detectors may split/merge collinear particles, and be poorly understood for soft ones
- High luminosity (or heavy ions collisions) add a lot of soft particles to hard event

IRC safety provides resiliency to such effects
(plus, at some point in the future you may wish to compare your measurement to a calculation)

Seedless IRC-safe Cone (SC-SM): SISCone

Salam, Soyez, arXiv:0704:0292

Seeds are a problem:
they lead to finding only some of the stable cones

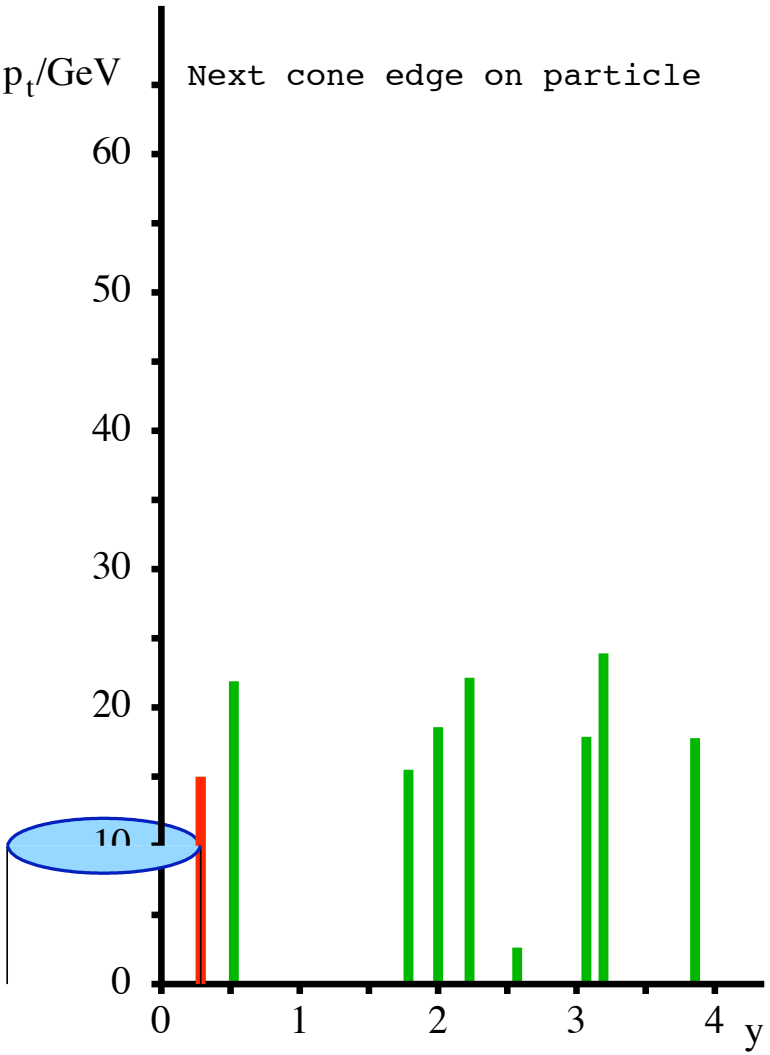
Obvious solution:
find ALL stable cones, testing all possible combinations of N particles

Unfortunately, this takes $N2^N$ operations:
the age of the universe for only 100 particles

Way out: a geometrical solution → SISCone

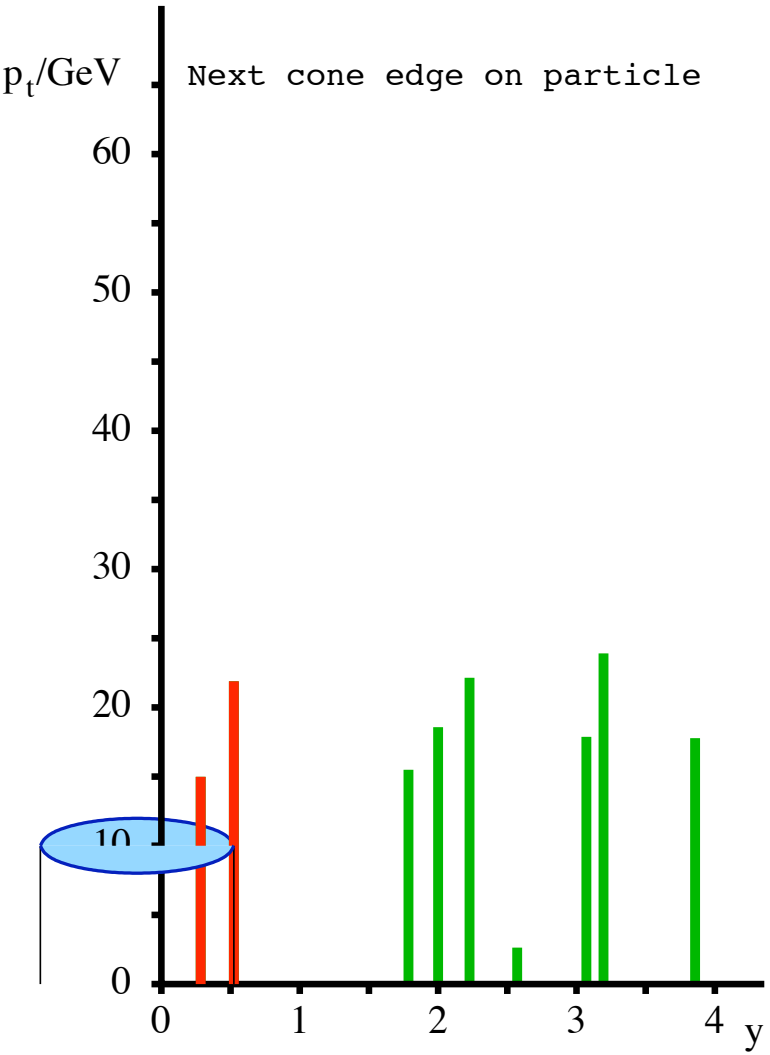
The first (and only?) IRC-safe cone algorithm for hadronic collisions

Key innovation: circular enclosures



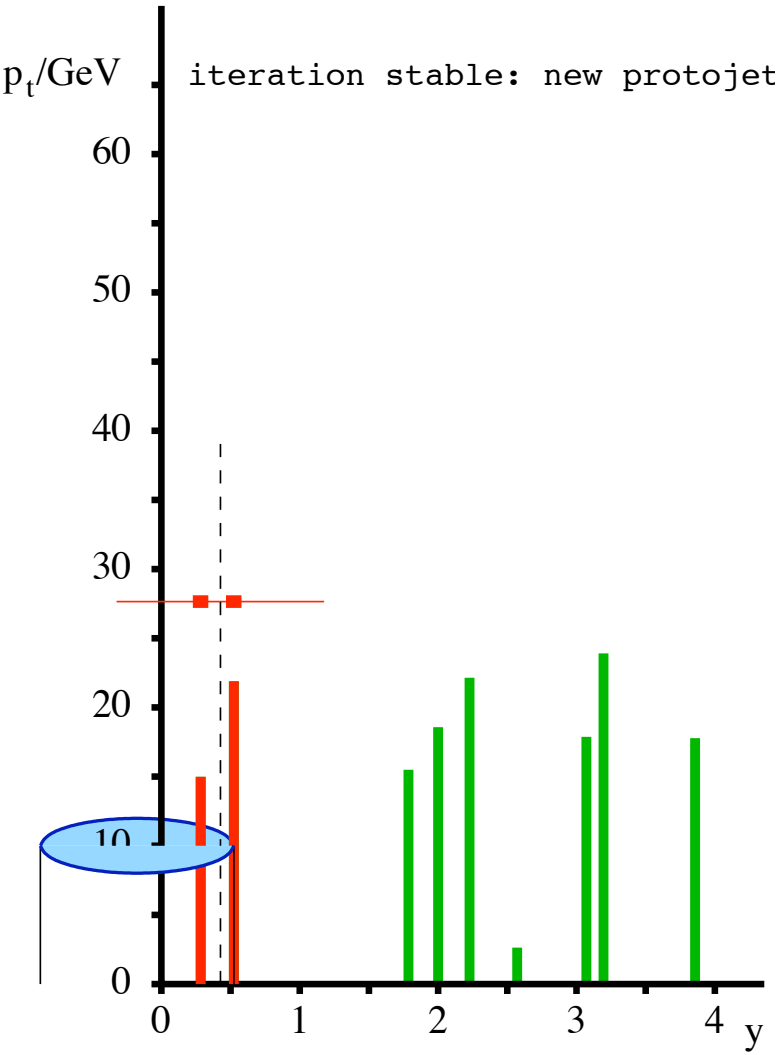
In 1-dim: slide a 'cone' along the axis

Key innovation: circular enclosures



In 1-dim: slide a 'cone' along the axis

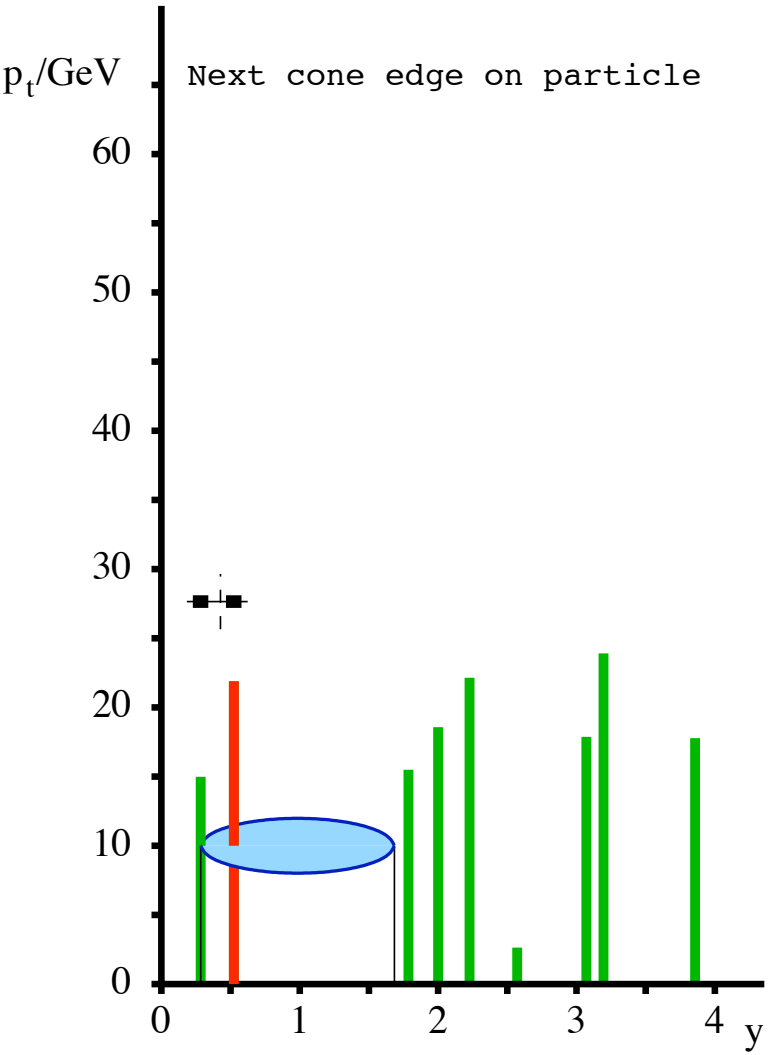
Key innovation: circular enclosures



In 1-dim: slide a 'cone' along the axis

Check stability: OK. New protojet

Key innovation: circular enclosures

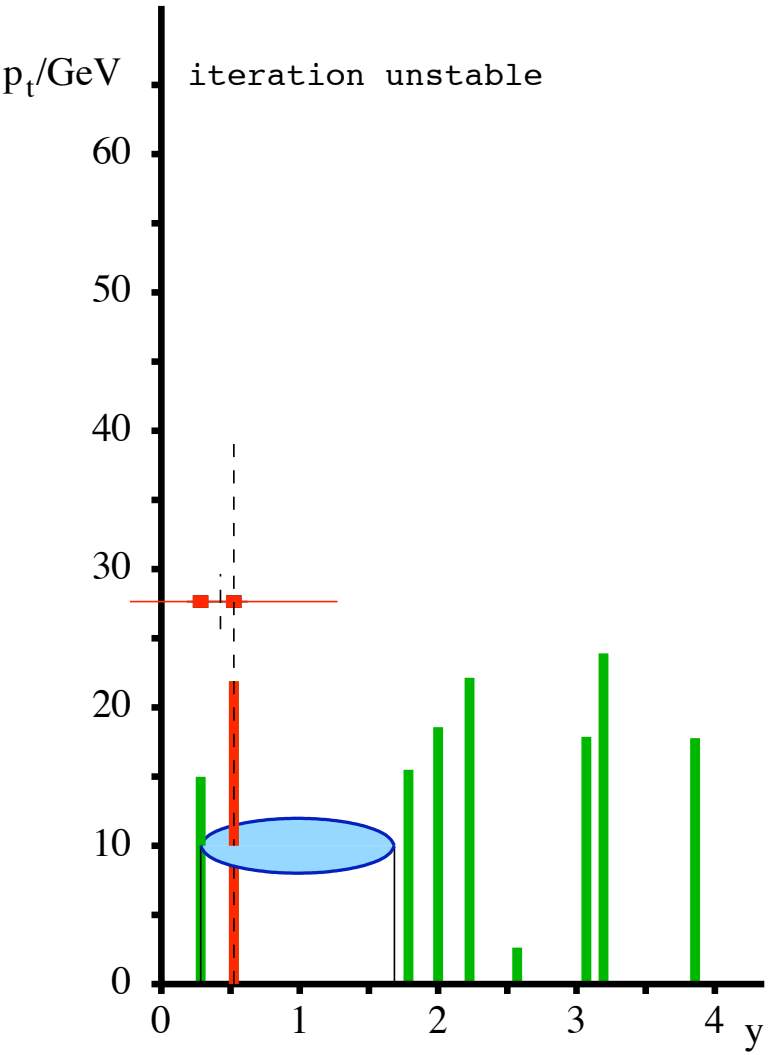


In 1-dim: slide a 'cone' along the axis

Check stability: OK. New protojet

Move on: first particle on edge: new enclosure

Key innovation: circular enclosures



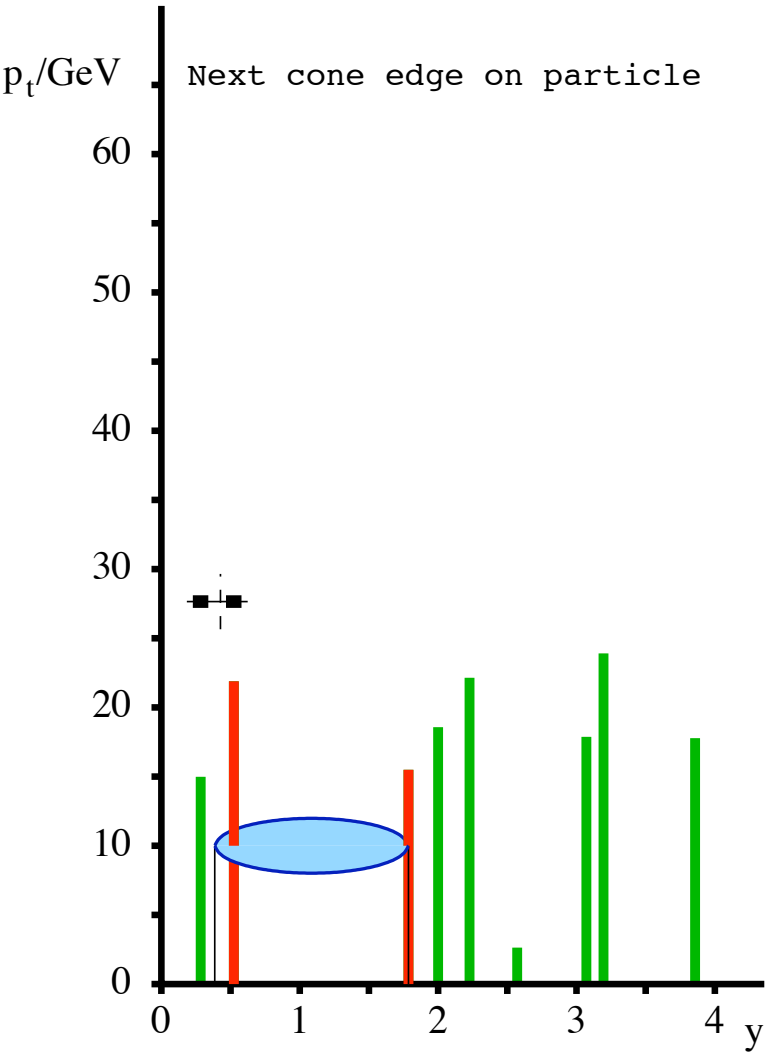
In 1-dim: slide a 'cone' along the axis

Check stability: OK. New protojet

Move on: first particle on edge: new enclosure

Unstable. Keep going

Key innovation: circular enclosures



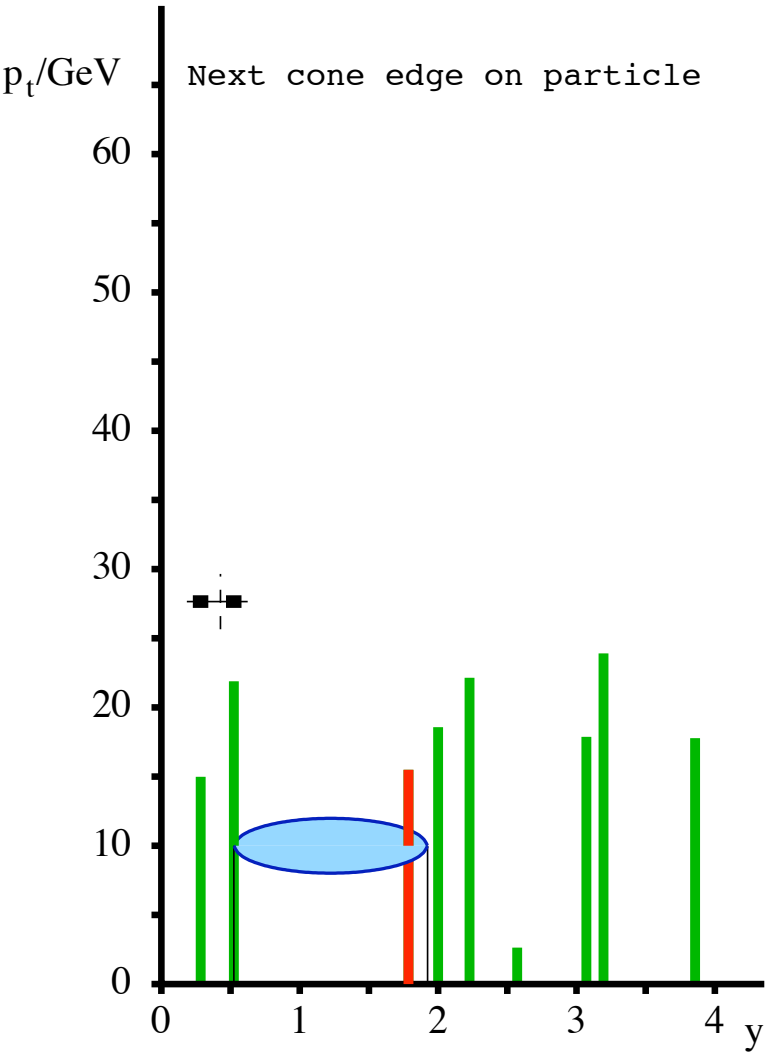
In 1-dim: slide a 'cone' along the axis

Check stability: OK. New protojet

Move on: first particle on edge: new enclosure

Unstable. Keep going

Key innovation: circular enclosures



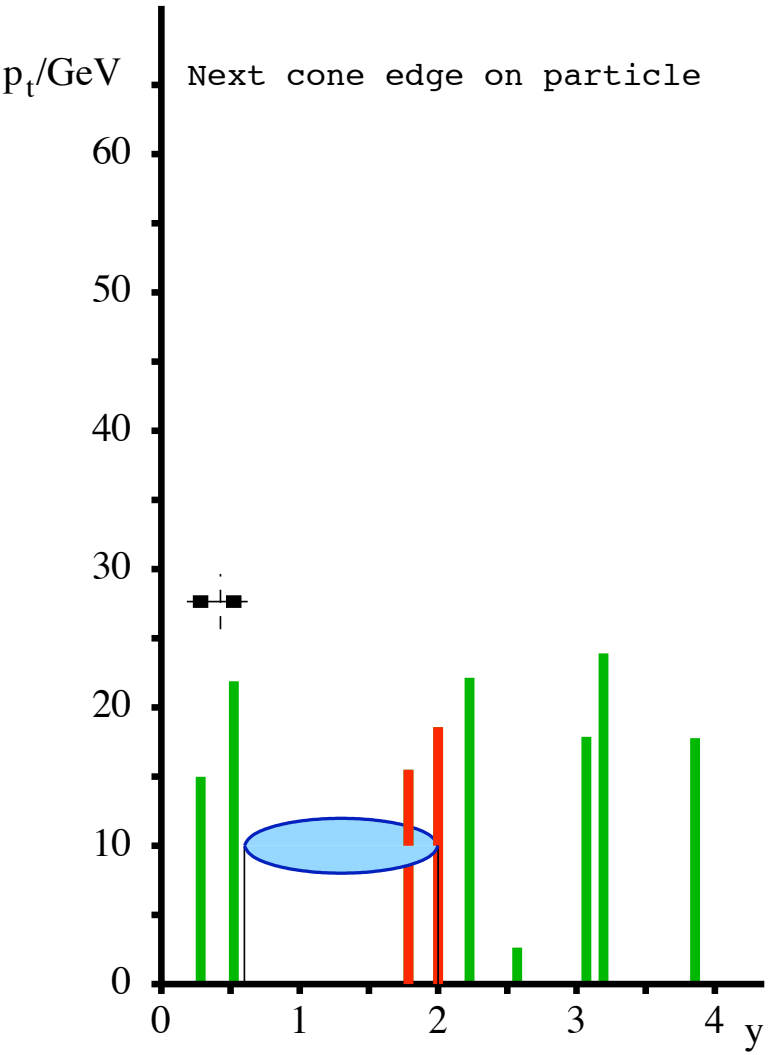
In 1-dim: slide a 'cone' along the axis

Check stability: OK. New protojet

Move on: first particle on edge: new enclosure

Unstable. Keep going

Key innovation: circular enclosures



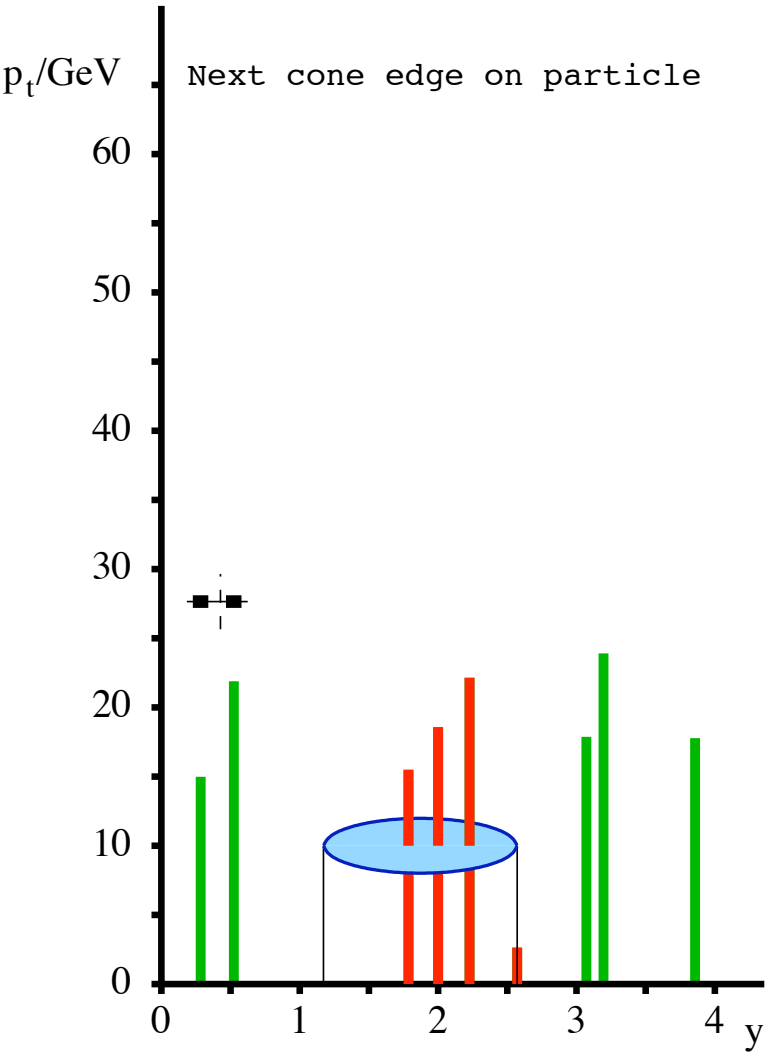
In 1-dim: slide a 'cone' along the axis

Check stability: OK. New protojet

Move on: first particle on edge: new enclosure

Unstable. Keep going

Key innovation: circular enclosures



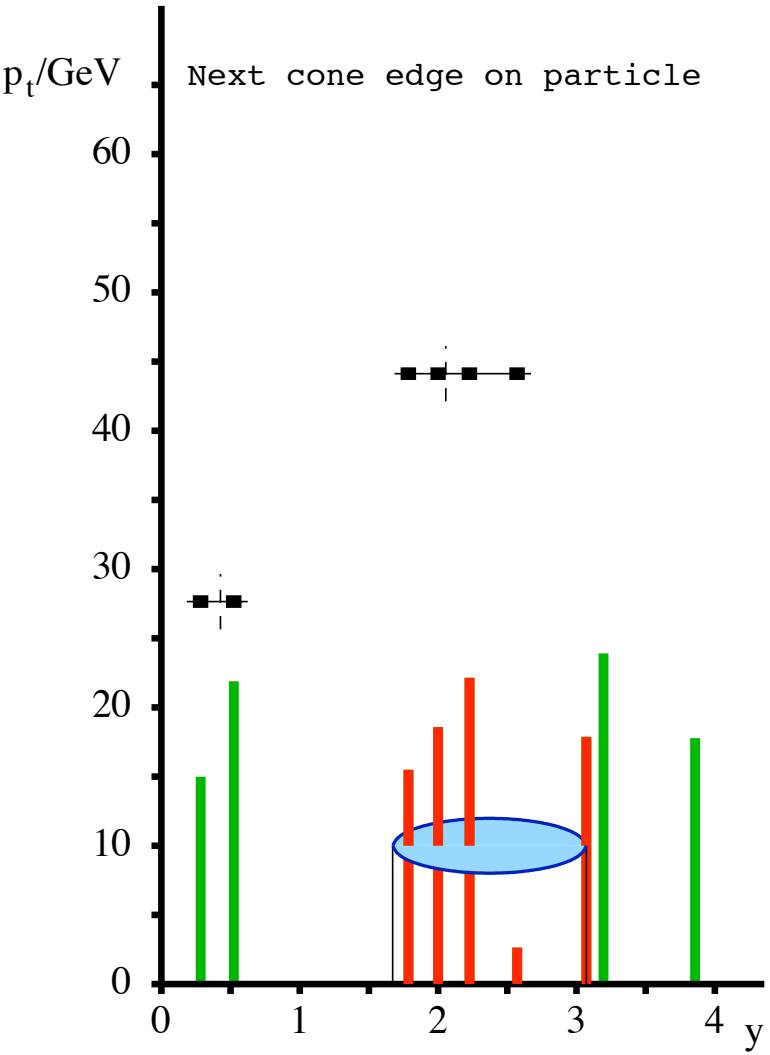
In 1-dim: slide a 'cone' along the axis

Check stability: OK. New protojet

Move on: first particle on edge: new enclosure

Unstable. Keep going

Key innovation: circular enclosures



In 1-dim: slide a 'cone' along the axis

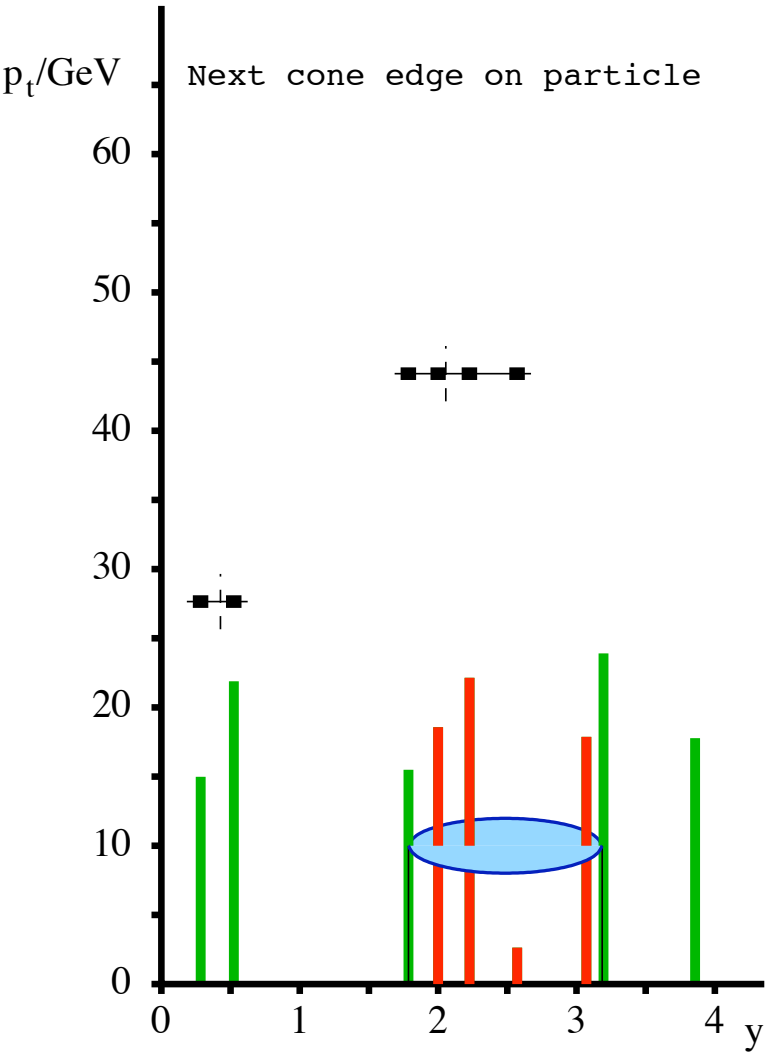
Check stability: OK. New protojet

Move on: first particle on edge: new enclosure

Unstable. Keep going

Found another stable one

Key innovation: circular enclosures



In 1-dim: slide a 'cone' along the axis

Check stability: OK. New protojet

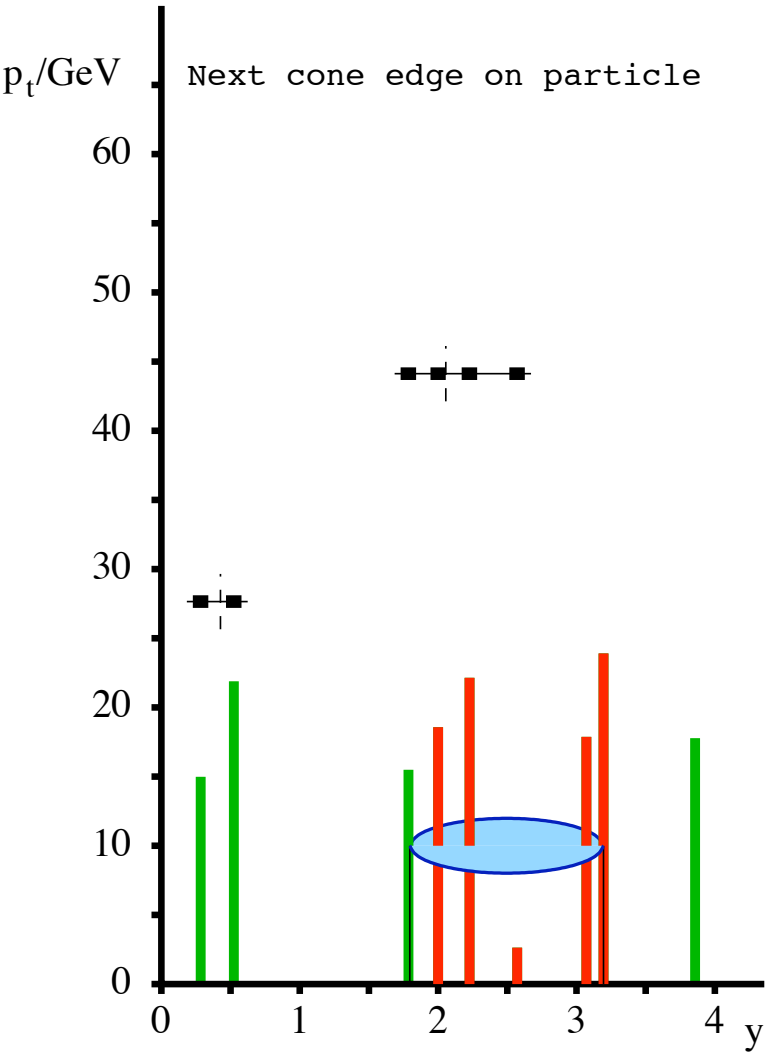
Move on: first particle on edge: new enclosure

Unstable. Keep going

Found another stable one

Etc, etc

Key innovation: circular enclosures



In 1-dim: slide a 'cone' along the axis

Check stability: OK. New protojet

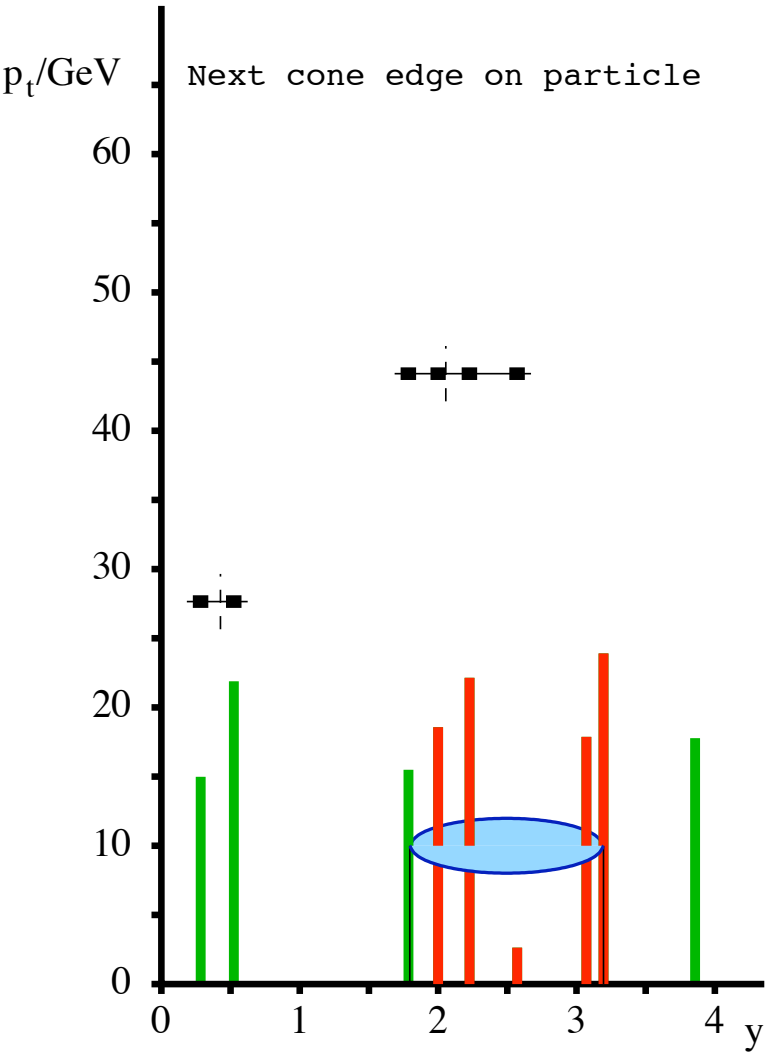
Move on: first particle on edge: new enclosure

Unstable. Keep going

Found another stable one

Etc, etc

Key innovation: circular enclosures



In 1-dim: slide a 'cone' along the axis

Check stability: OK. New protojet

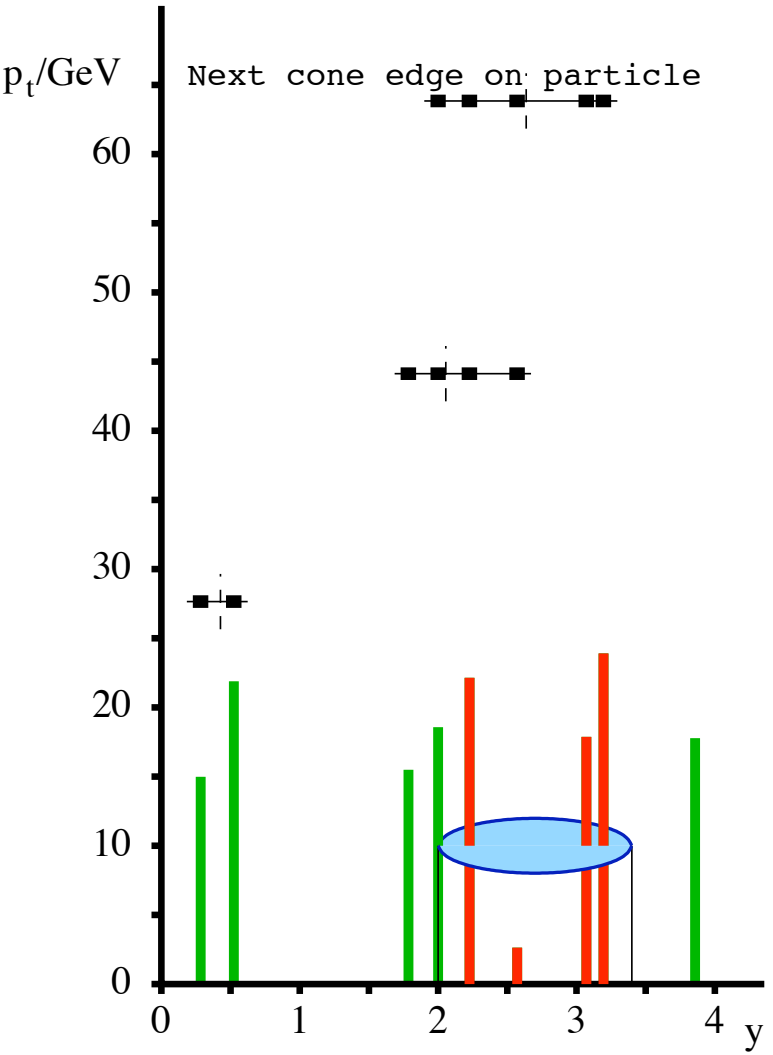
Move on: first particle on edge: new enclosure

Unstable. Keep going

Found another stable one

Etc, etc

Key innovation: circular enclosures



In 1-dim: slide a 'cone' along the axis

Check stability: OK. New protojet

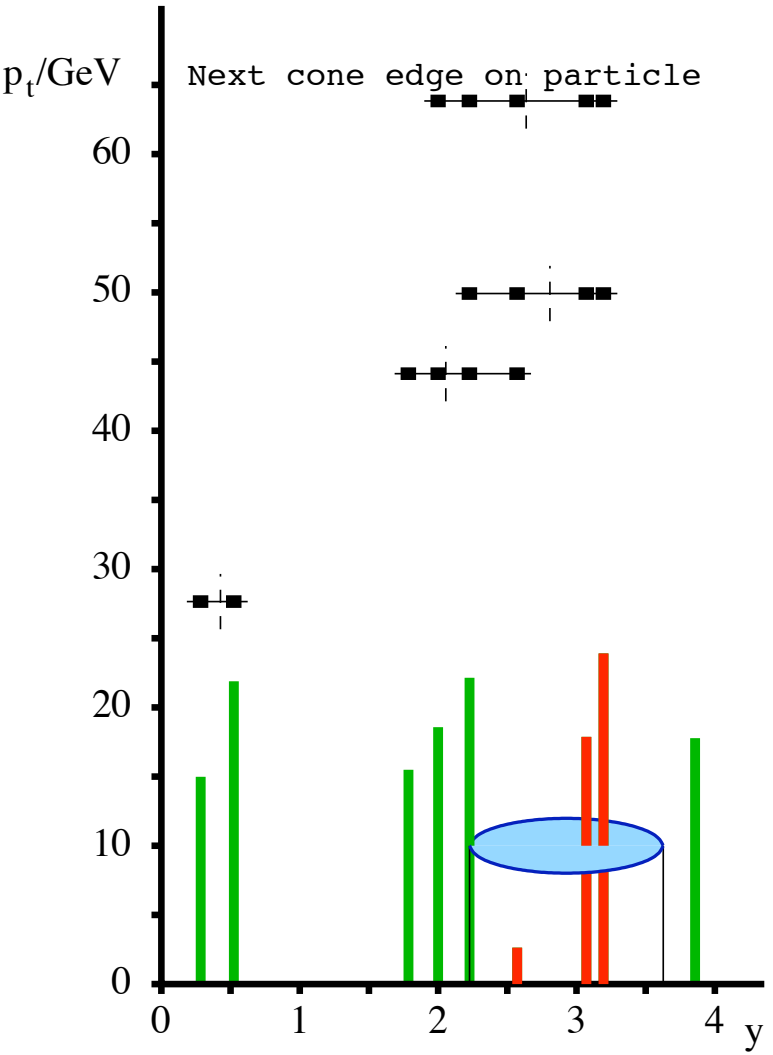
Move on: first particle on edge: new enclosure

Unstable. Keep going

Found another stable one

Etc, etc

Key innovation: circular enclosures



In 1-dim: slide a 'cone' along the axis

Check stability: OK. New protojet

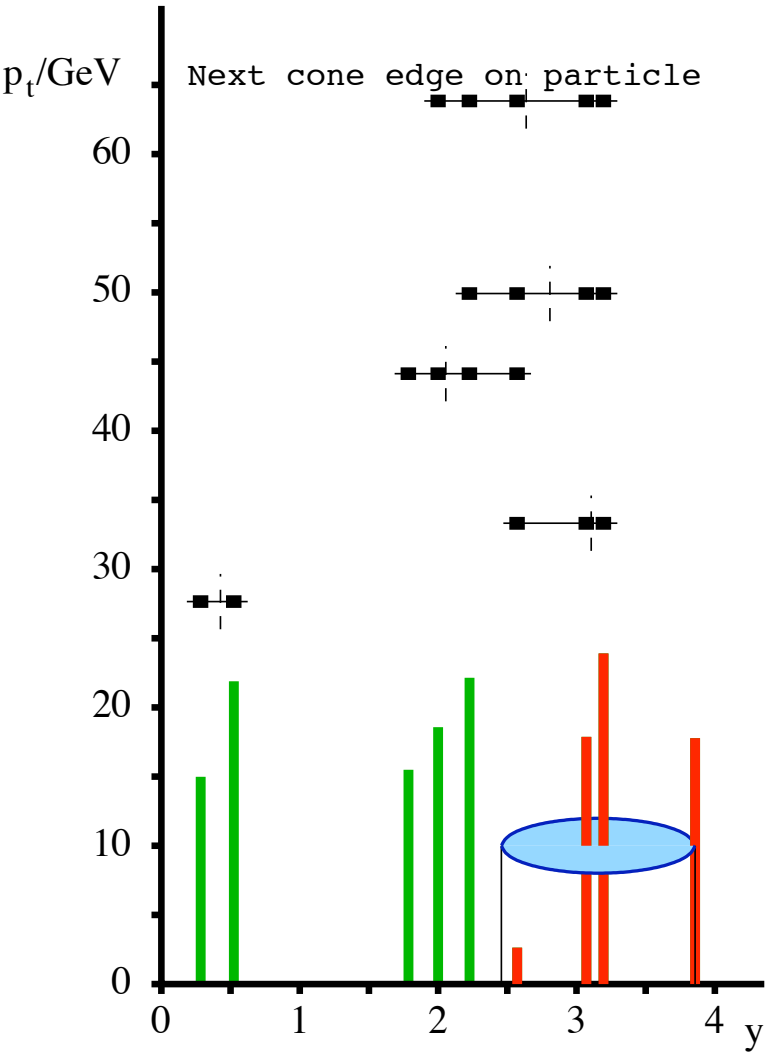
Move on: first particle on edge: new enclosure

Unstable. Keep going

Found another stable one

Etc, etc

Key innovation: circular enclosures



In 1-dim: slide a 'cone' along the axis

Check stability: OK. New protojet

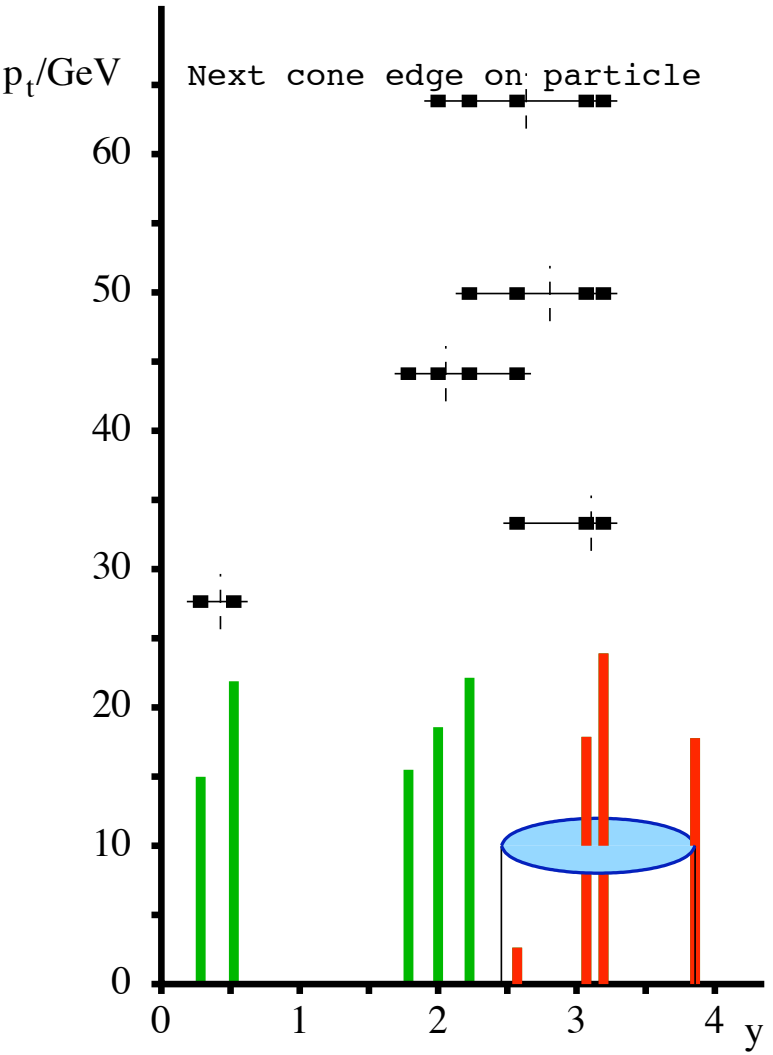
Move on: first particle on edge: new enclosure

Unstable. Keep going

Found another stable one

Etc, etc

Key innovation: circular enclosures



In 1-dim: slide a 'cone' along the axis

Check stability: OK. New protojet

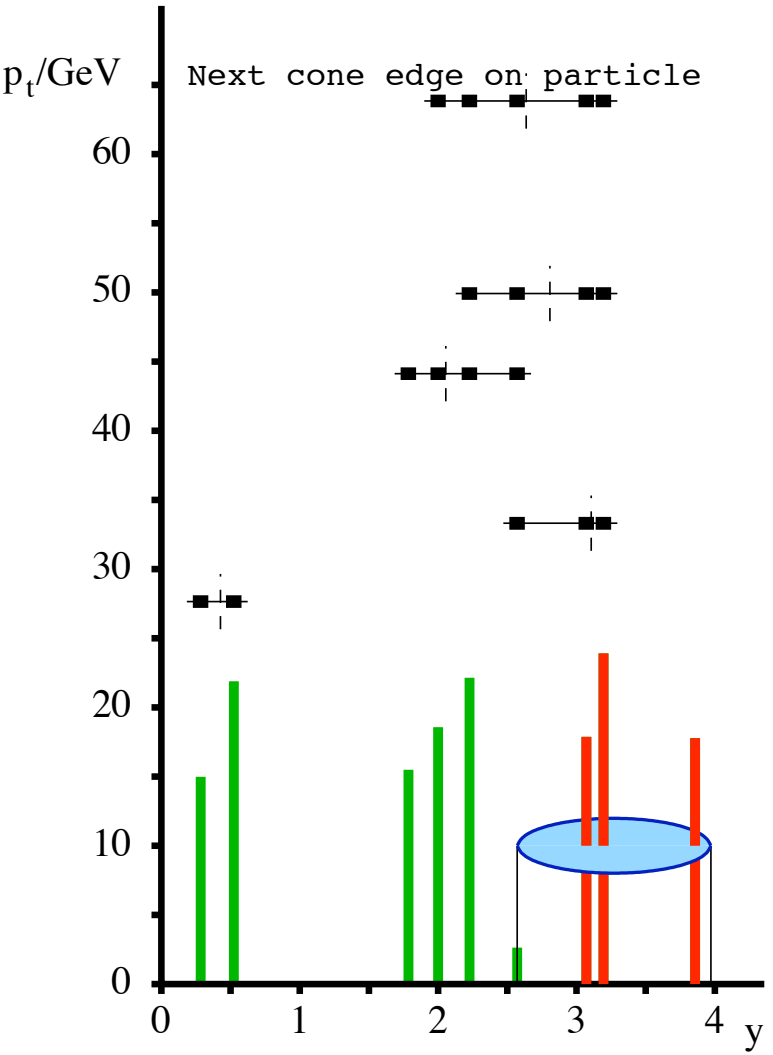
Move on: first particle on edge: new enclosure

Unstable. Keep going

Found another stable one

Etc, etc

Key innovation: circular enclosures



In 1-dim: slide a 'cone' along the axis

Check stability: OK. New protojet

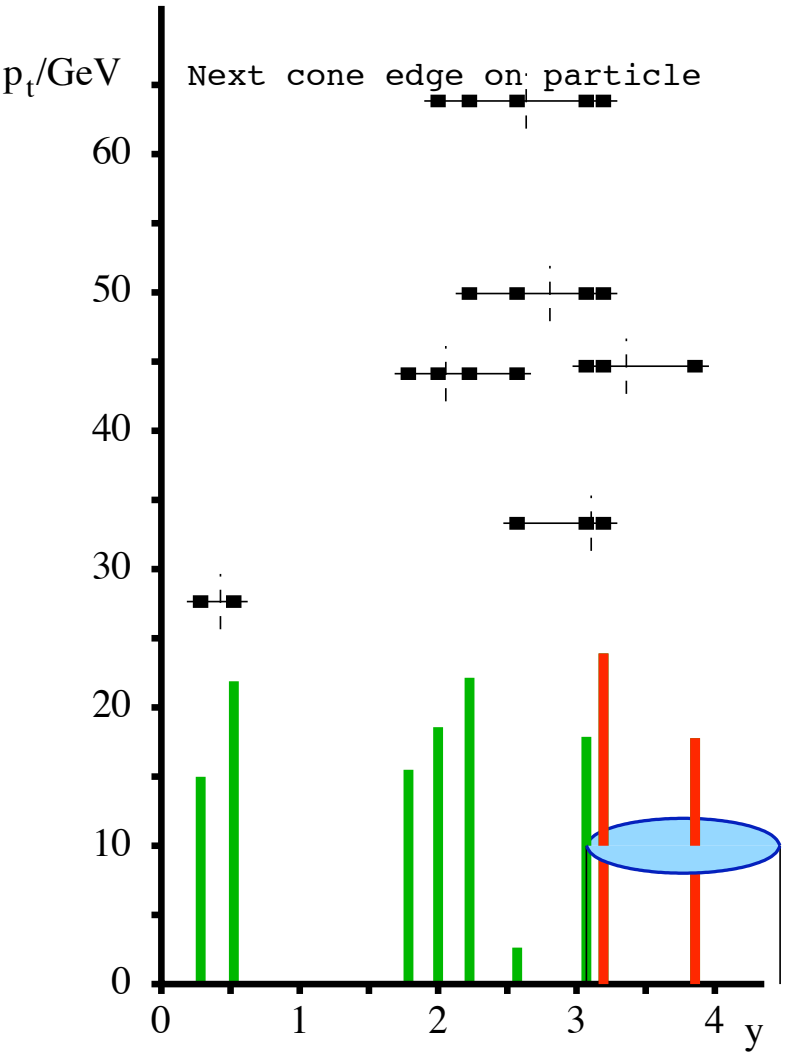
Move on: first particle on edge: new enclosure

Unstable. Keep going

Found another stable one

Etc, etc

Key innovation: circular enclosures



In 1-dim: slide a 'cone' along the axis

Check stability: OK. New protojet

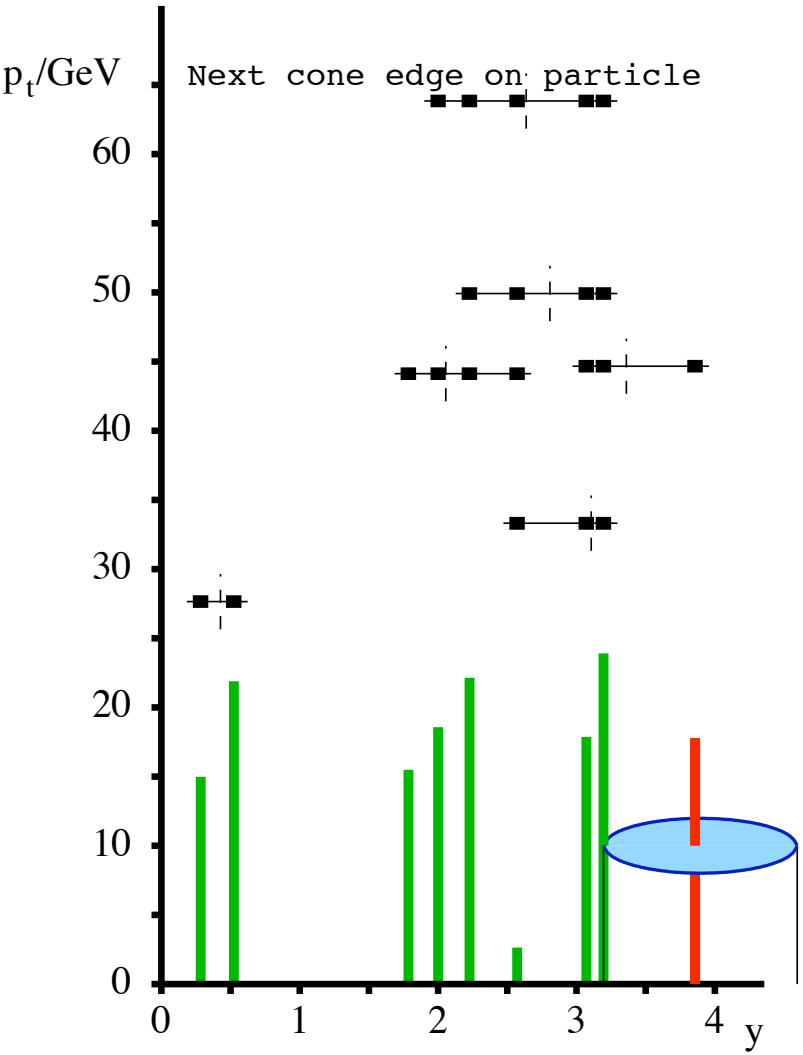
Move on: first particle on edge: new enclosure

Unstable. Keep going

Found another stable one

Etc, etc

Key innovation: circular enclosures



In 1-dim: slide a 'cone' along the axis

Check stability: OK. New protojet

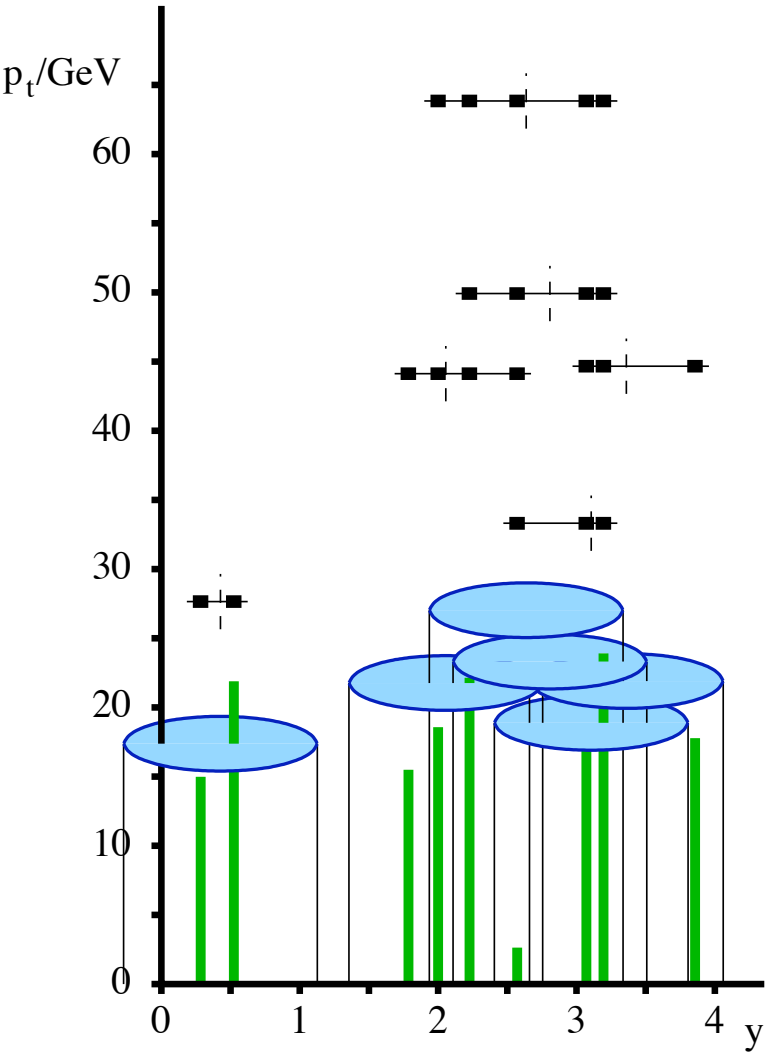
Move on: first particle on edge: new enclosure

Unstable. Keep going

Found another stable one

Etc, etc

Key innovation: circular enclosures



In 1-dim: slide a 'cone' along the axis

Check stability: OK. New protojet

Move on: first particle on edge: new enclosure

Unstable. Keep going

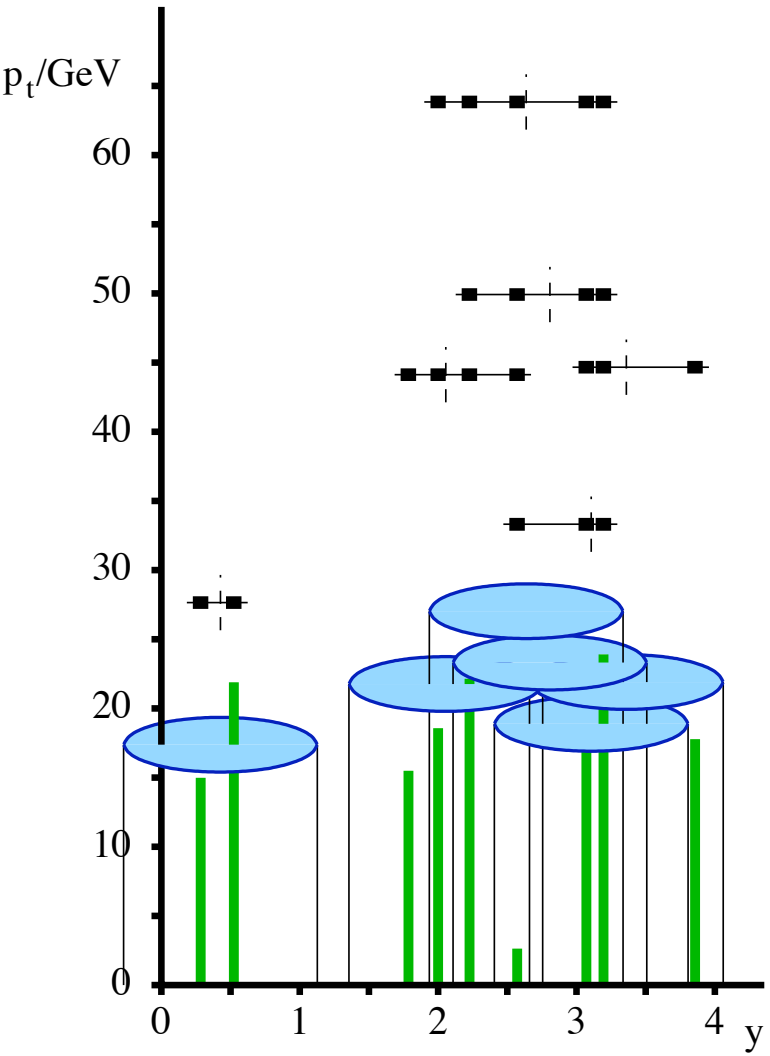
Found another stable one

Etc, etc

And finally we get to this

SISCone v. IC-SM

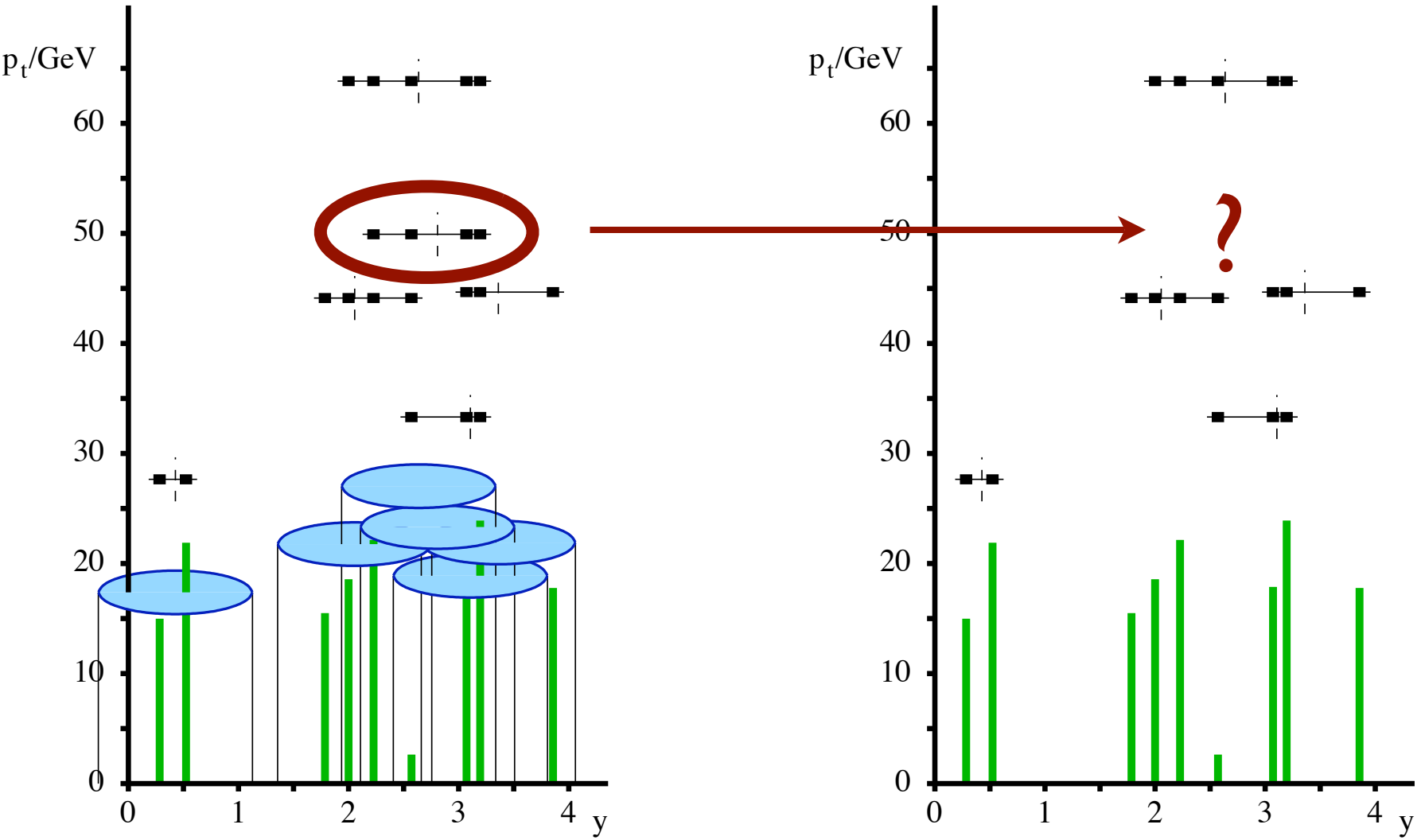
These are ALL the stable cones



SISCone v. IC-SM

These are ALL the stable cones

Compare to those found by IC-SM: one is missing



Cones Infrared (un)safety

Q: How often are the hard jets changed by the addition of a soft particle?

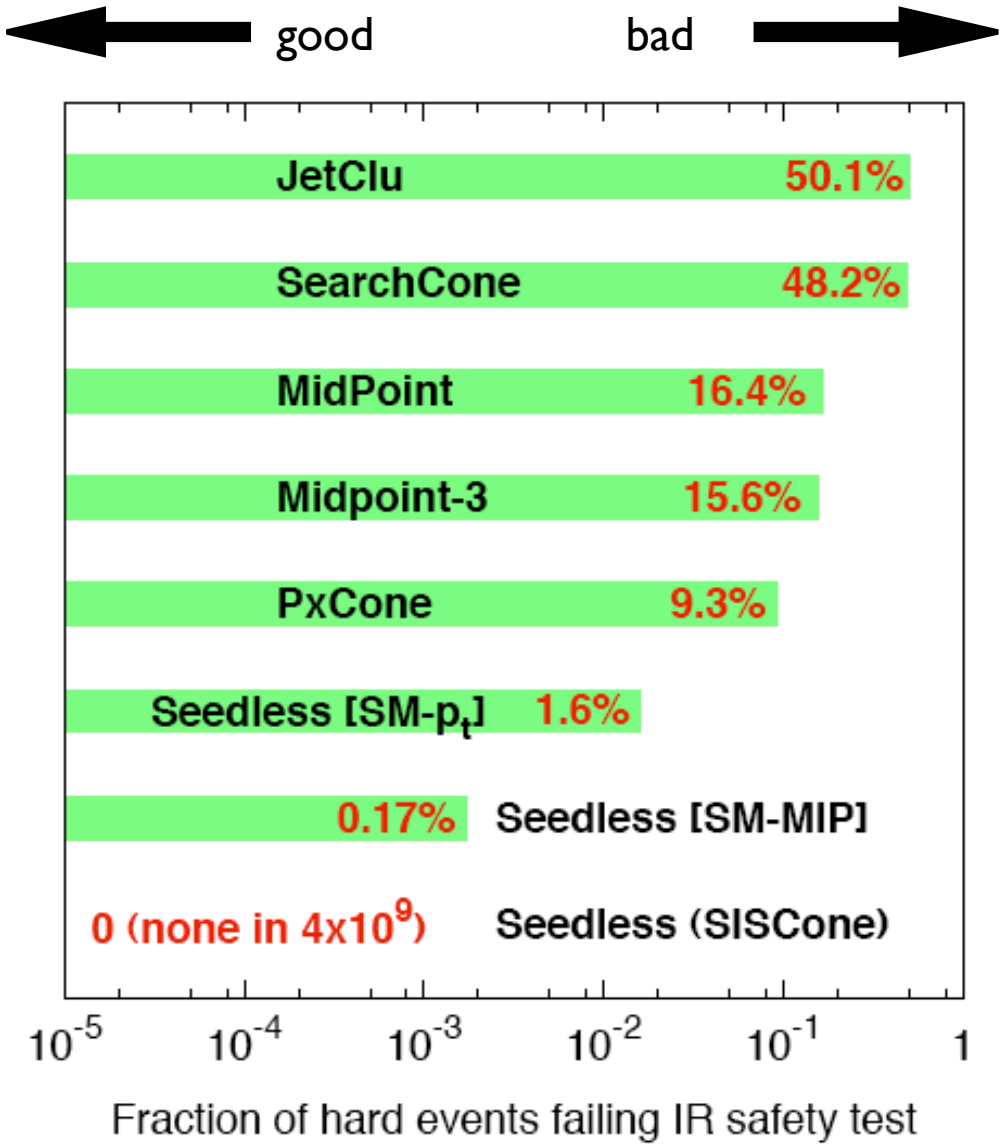
- Generate event with $2 < N < 10$ hard particles, find jets
- Add $1 < N_{soft} < 5$ soft particles, find jets again

A: [repeatedly]

- If the jets are different, algorithm is IR unsafe.

Unsafety level	failure rate
2 hard + 1 soft	$\sim 50\%$
3 hard + 1 soft	$\sim 15\%$
SISCone	IR safe !

Be careful with split–merge too



Recombination algorithms

(Inclusive version)

- Calculate the distances between the particles: $\mathbf{d_{ij}}$
- Calculate the beam distances: $\mathbf{d_{iB}}$
- 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

IRC safety can usually be seen to be trivially guaranteed

The k_t algorithm and its siblings

One can generalise the k_t distance measure:

$$d_{ij} = \min(k_{ti}^{2p}, k_{tj}^{2p}) \frac{\Delta y^2 + \Delta\phi^2}{R^2} \qquad d_{iB} = k_{ti}^{2p}$$

p = 1 k_t algorithm

S. Catani, Y. Dokshitzer, M. Seymour and B. Webber, Nucl. Phys. B406 (1993) 187
S.D. Ellis and D.E. Soper, Phys. Rev. D48 (1993) 3160

The k_t algorithm and its siblings

One can generalise the k_t distance measure:

$$d_{ij} = \min(k_{ti}^{2p}, k_{tj}^{2p}) \frac{\Delta y^2 + \Delta\phi^2}{R^2} \qquad d_{iB} = k_{ti}^{2p}$$

p = 1 k_t algorithm — S. Catani, Y. Dokshitzer, M. Seymour and B. Webber, Nucl. Phys. B406 (1993) 187; S.D. Ellis and D.E. Soper, Phys. Rev. D48 (1993) 3160

p = 0 Cambridge/Aachen algorithm — Y. Dokshitzer, G. Leder, S.Moretti and B. Webber, JHEP 08 (1997) 001; M. Wobisch and T. Wengler, hep-ph/9907280

The k_t algorithm and its siblings

One can generalise the k_t distance measure:

$$d_{ij} = \min(k_{ti}^{2p}, k_{tj}^{2p}) \frac{\Delta y^2 + \Delta\phi^2}{R^2} \qquad d_{iB} = k_{ti}^{2p}$$

p = 1 k_t algorithm — S. Catani, Y. Dokshitzer, M. Seymour and B. Webber, Nucl. Phys. B406 (1993) 187; S.D. Ellis and D.E. Soper, Phys. Rev. D48 (1993) 3160

p = 0 Cambridge/Aachen algorithm — Y. Dokshitzer, G. Leder, S.Moretti and B. Webber, JHEP 08 (1997) 001; M. Wobisch and T. Wengler, hep-ph/9907280

p = -1 **anti-k_t algorithm** — MC, G. Salam and G. Soyez, arXiv:0802.1189

NB: in anti-kt pairs with a **hard** particle will cluster first: if no other hard particles are close by, the algorithm will give **perfect cones**

Quite ironically, a sequential recombination algorithm is the 'perfect' cone algorithm

The IRC safe algorithms

k_t	SR $d_{ij} = \min(k_{ti}^2, k_{tj}^2)\Delta R_{ij}^2/R^2$ hierarchical in rel p_t	Catani et al '91 Ellis, Soper '93	NlnN
Cambridge/ Aachen	SR $d_{ij} = \Delta R_{ij}^2/R^2$ hierarchical in angle	Dokshitzer et al '97 Wengler, Wobish '98	NlnN
anti-k_t	SR $d_{ij} = \min(k_{ti}^{-2}, k_{tj}^{-2})\Delta R_{ij}^2/R^2$ gives perfectly conical hard jets	MC, Salam, Soyez '08 (Delsart, Loch)	$N^{3/2}$
SISCone	Seedless iterative cone with split-merge gives 'economical' jets	Salam, Soyez '07	N^2lnN

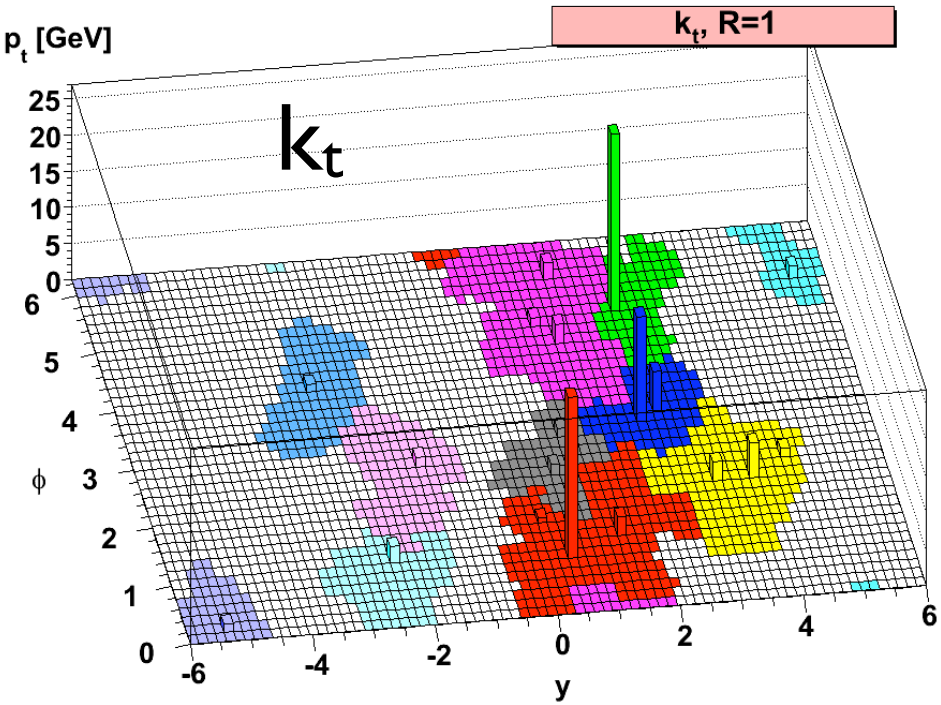
The IRC safe algorithms

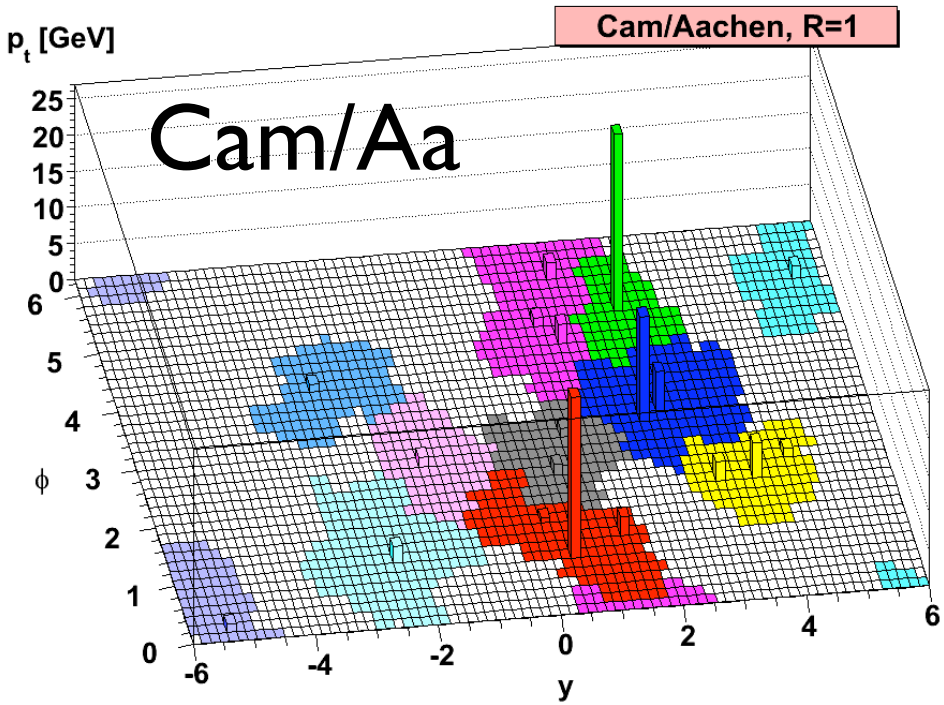
k_t	SR $d_{ij} = \min(k_{ti}^2, k_{tj}^2)\Delta R_{ij}^2/R^2$ hierarchical in rel p_t	Catani et al '91 Ellis, Soper '93	NlnN
Cambridge/ Aachen	SR $d_{ij} = \Delta R_{ij}^2/R^2$ hierarchical in angle	Dokshitzer et al '97 Wengler, Wobish '98	NlnN
anti-k_t	SR $d_{ij} = \min(k_{ti}^{-2}, k_{tj}^{-2})\Delta R_{ij}^2/R^2$ gives perfectly conical hard jets	MC, Salam, Soyez '08 (Delsart, Loch)	$N^{3/2}$
SISCone	Seedless iterative cone with split-merge gives 'economical' jets	Salam, Soyez '07	N^2lnN

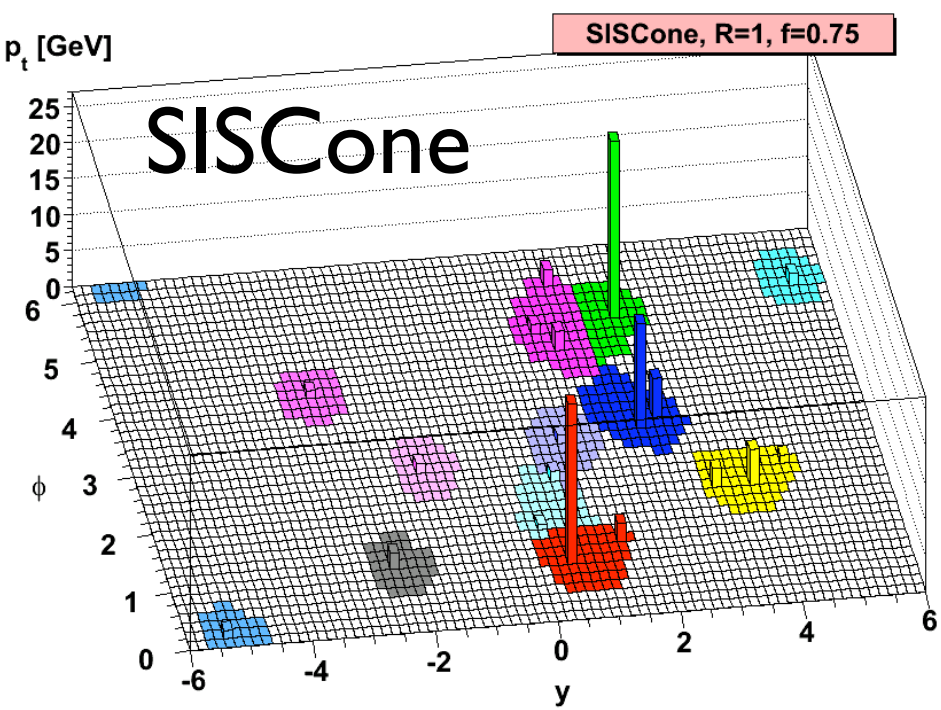
We call these algs '**second-generation**' ones

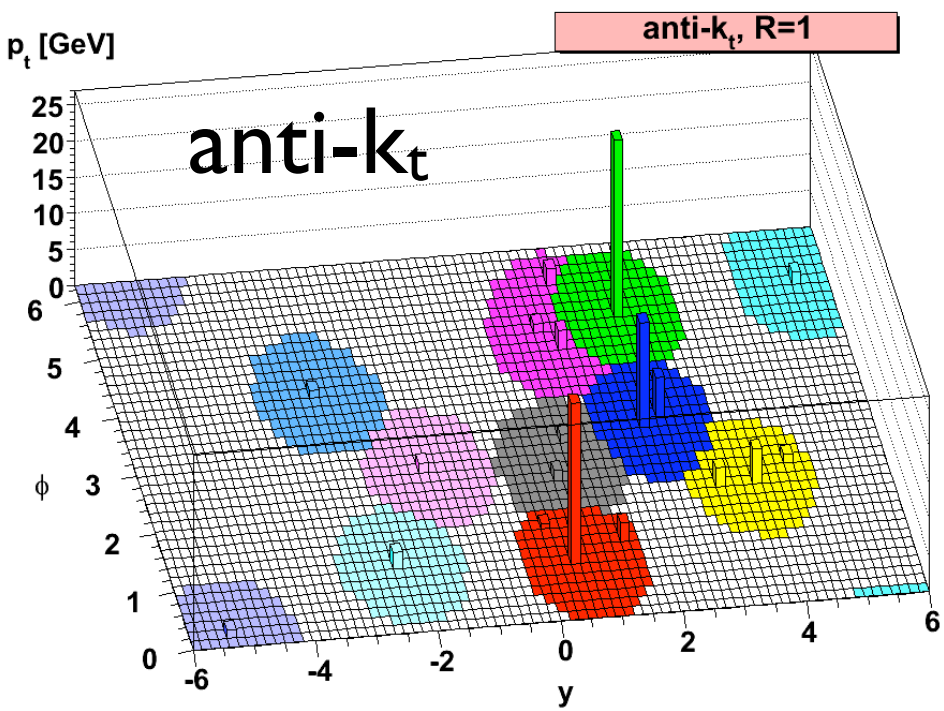
All are available in FastJet, http://fastjet.fr

(As well as many IRC unsafe ones)


k_t, R=1
p_t [GeV]
k_t
y
ϕ


Cam/Aachen, R=1
p_t [GeV]
Cam/Aa
y
ϕ


SISCone, R=1, f=0.75
p_t [GeV]
SISCone
y
ϕ


anti-k_t, R=1
p_t [GeV]
anti-k_t
y
ϕ

Replacements

If you care about IRC safety but don't want to stray too far from algorithms used so far, these are possible replacements:

JetClu
ATLAS Cone
MidPoint
→ SISCone (As fast, but IRC safe)

Iterative Cone (PR) → anti-k_t (Gives regular cones too, but IRC safe)

In addition, k_t and Cambridge/Aachen can provide further flexibility

Different algorithms spanning a series of *different and complementary characteristics*: should be enough for most purposes

One should probably try to concentrate on these, both for analytical understanding and practical use in experiments, rather than using IRC unsafe ones

Cambridge/Aachen with filtering

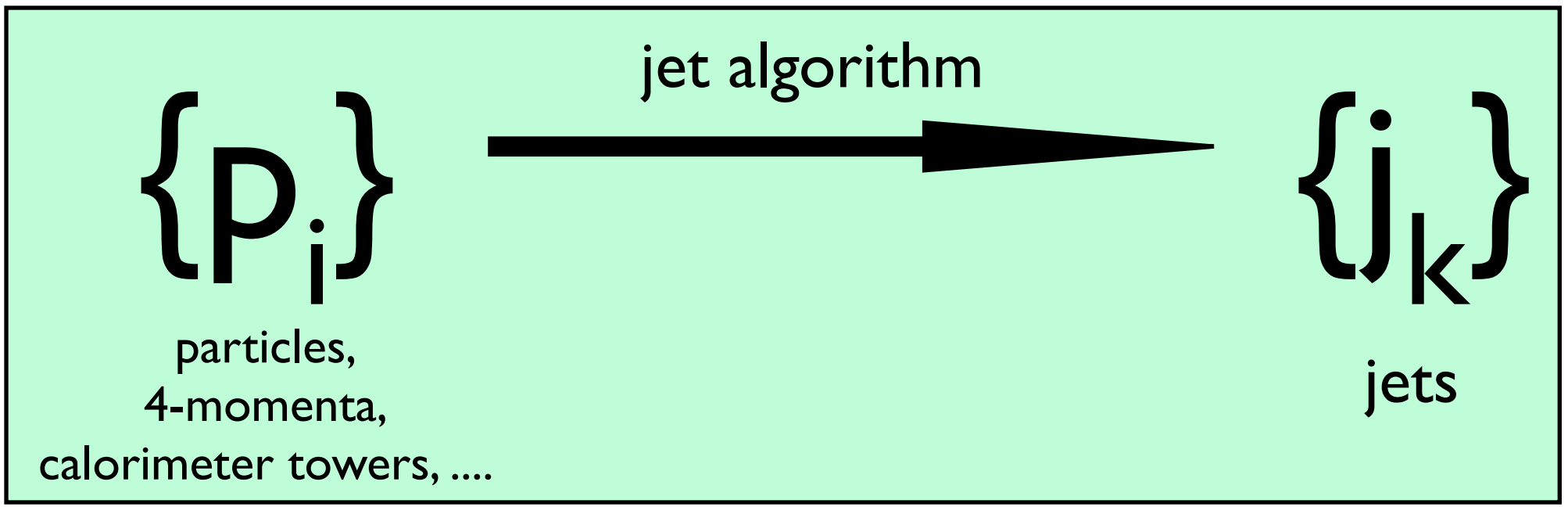
Butterworth, Davison, Rubin, Salam, arXiv:0802.2470

An example of a **third-generation** jet algorithm

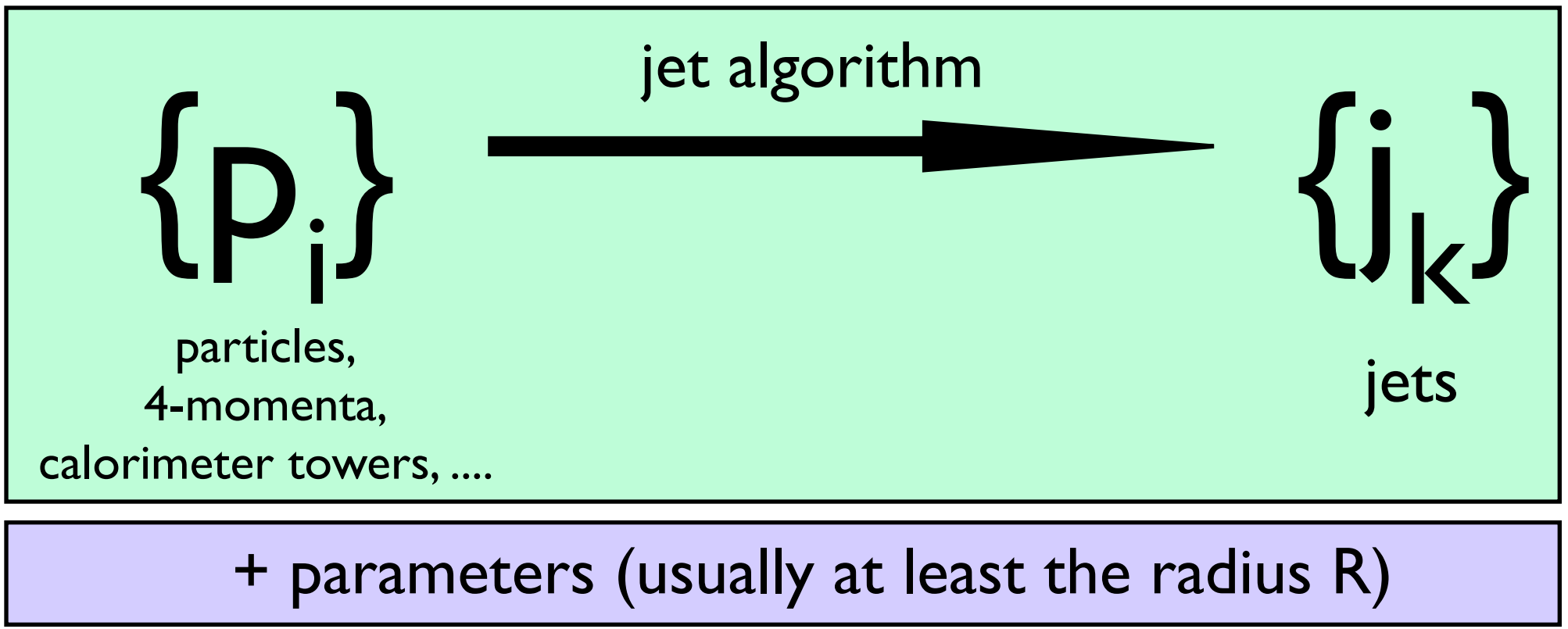
- Cluster with C/A and a given R
- Undo the clustering of each jet down to subjets with radius $x_{filt}R$
- Retain only the n_{filt} hardest subjets

Aim: limit sensitivity to background while retaining bulk of perturbative radiation

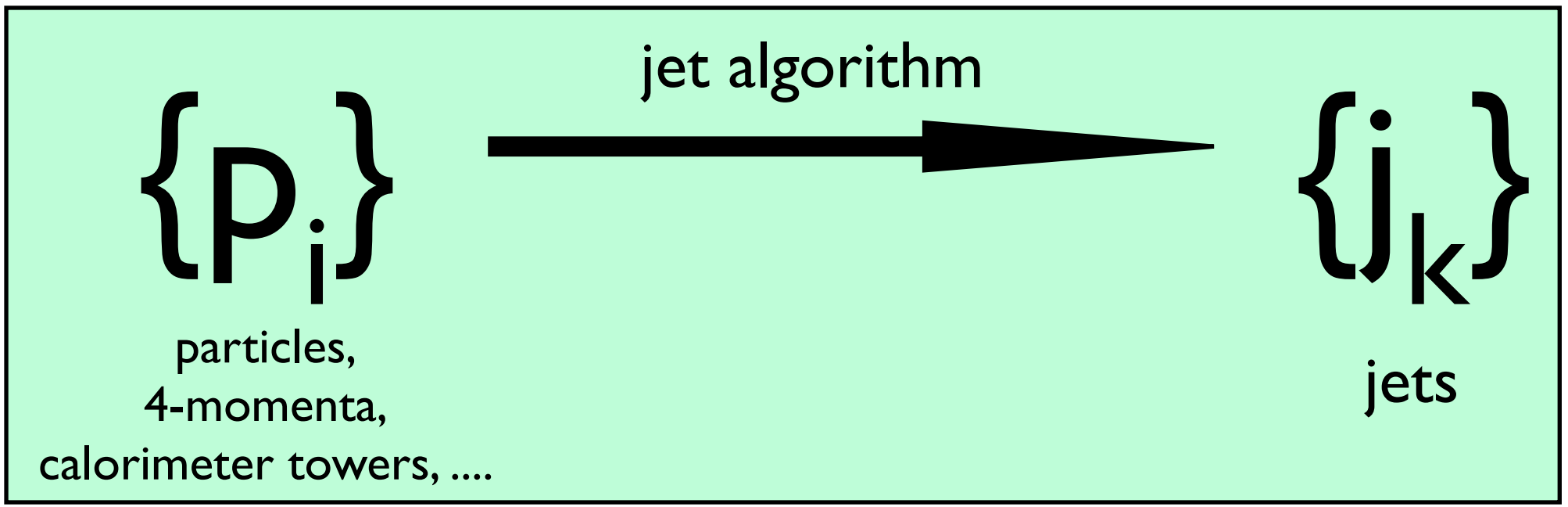
Jet definition



Jet definition



Jet definition



+ parameters (usually at least the radius R)

+ recombination scheme

Jet definition

Les Houches 2007 proceedings, arXiv:0803.0678

jet definition

jet algorithm

$\{p_i\}$ → $\{j_k\}$

particles,
4-momenta,
calorimeter towers, ...

jets

+ parameters (usually at least the radius R)

+ recombination scheme

Which R to choose?

The value of R matters because it affects,
in opposite ways, a number of things:

Small R:

Limit underlying event and pileup contamination

Better resolve many-jets events

Large R:

Limit perturbative radiation loss ('out-of-cone')

Limit non-perturbative hadronisation effects

The best compromise will in general
depend on the specific observable

R-dependent effects

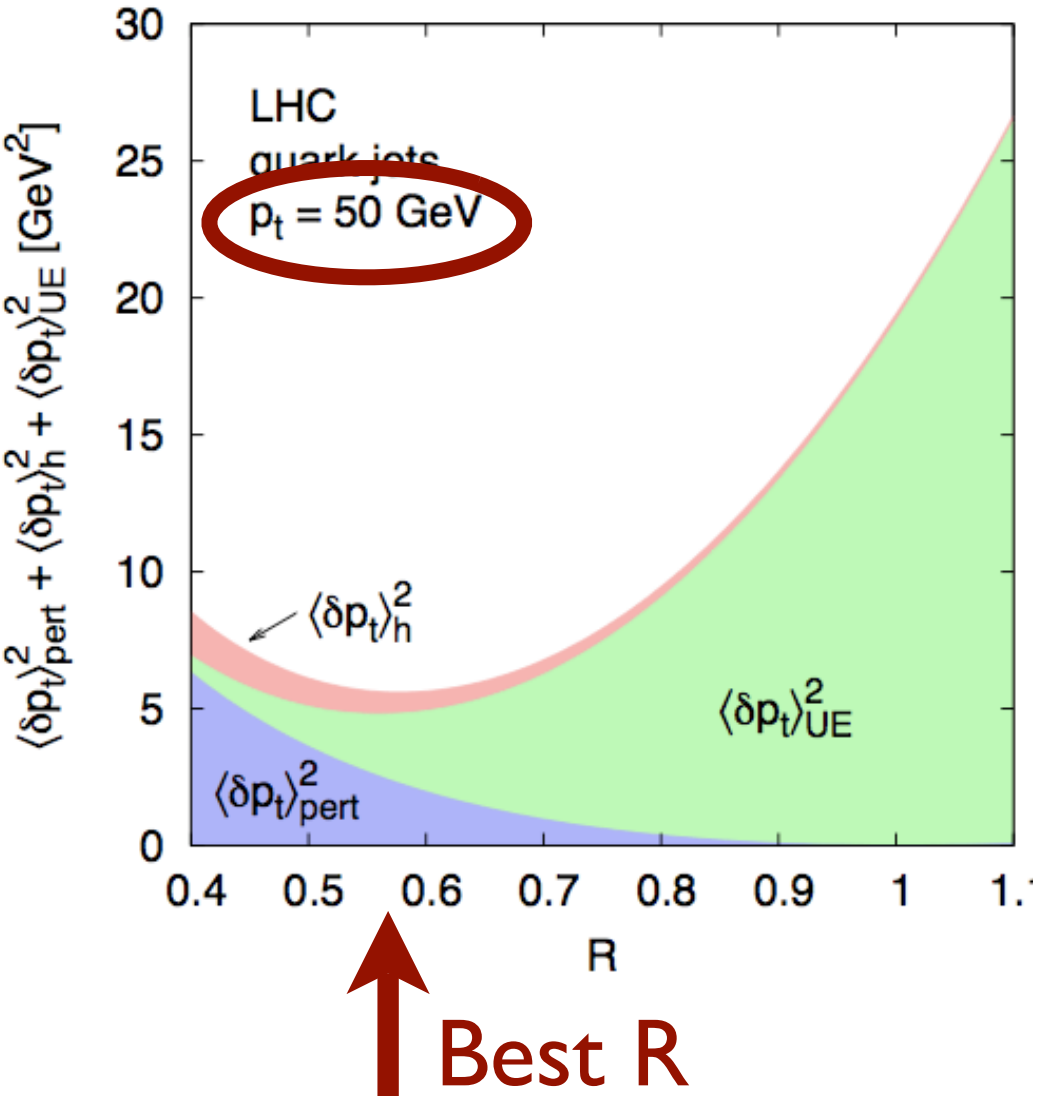
Perturbative radiation: $$\Delta p_t \simeq \frac{\alpha_s(C_F, C_A)}{\pi} p_t \ln R$$

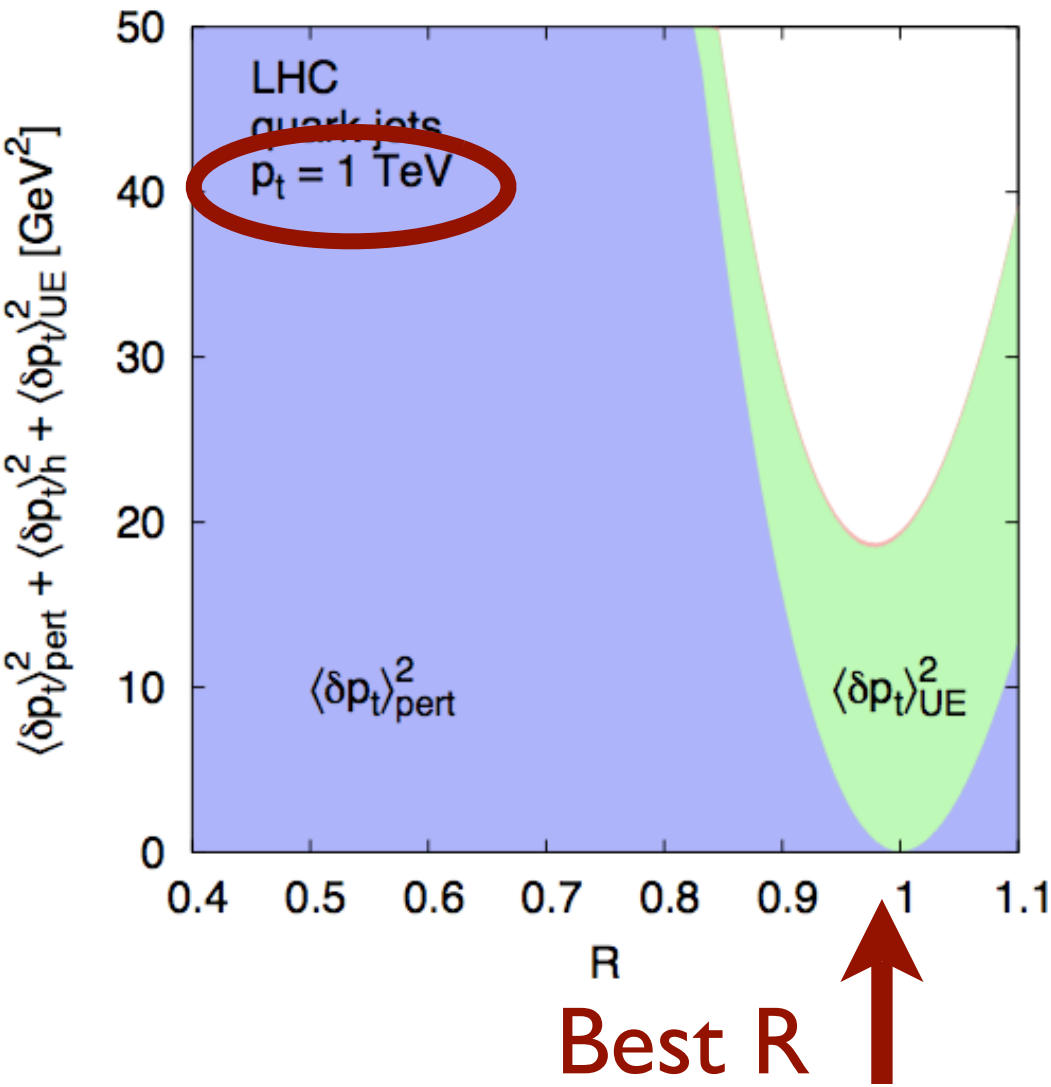
Hadronisation: $$\Delta p_t \simeq \frac{(C_F, C_A)}{R} \times 0.4 \text{ GeV}$$

Underlying Event: $$\Delta p_t \simeq \frac{R^2}{2} \times (2.5 \text{ (Tevatron)} - 15 \text{ (LHC) GeV})$$

Analytical estimates,
Dasgupta, Magnea, Salam, arXiv:0712.3014

Minimize $\Sigma(\Delta p_t)^2$





Dasgupta, Magnea, Salam, arXiv:0712.3014

All IRC safe algorithms are equal, but some are more equal than others

Depending on the analysis you wish to perform,
a jet definition may give better results than others

Reconstruction of a di-jet mass peak

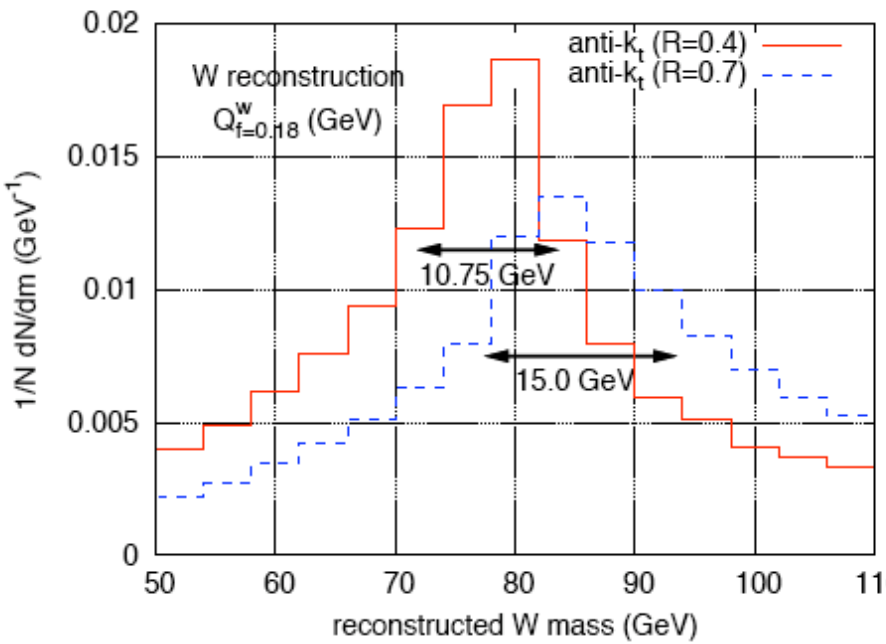
To compare different algorithms, define **figures of merit**:

MC, Rojo, Salam, Soyez, arXiv:0810.1304
Les Houches 2007 proceedings, arXiv:0803:0678

1. $Q^{w}_{f=z}$

Smallest width of an histogram window containing a fraction f=z of the generated objects, i.e.

$$f = \left(\frac{\#\text{ reconstructed massive objects in window of width } w}{\text{Total } \#\text{ generated massive objects}} \right)$$



2. Maximum fraction of events in window of given width:

$$Q^{1/f}_{w=x\sqrt{M}} \equiv \left(\frac{\text{Max } \#\text{ reco. massive objects in window of width } w = x\sqrt{M}}{\text{Total } \#\text{ generated massive objects}} \right)^{-1}$$

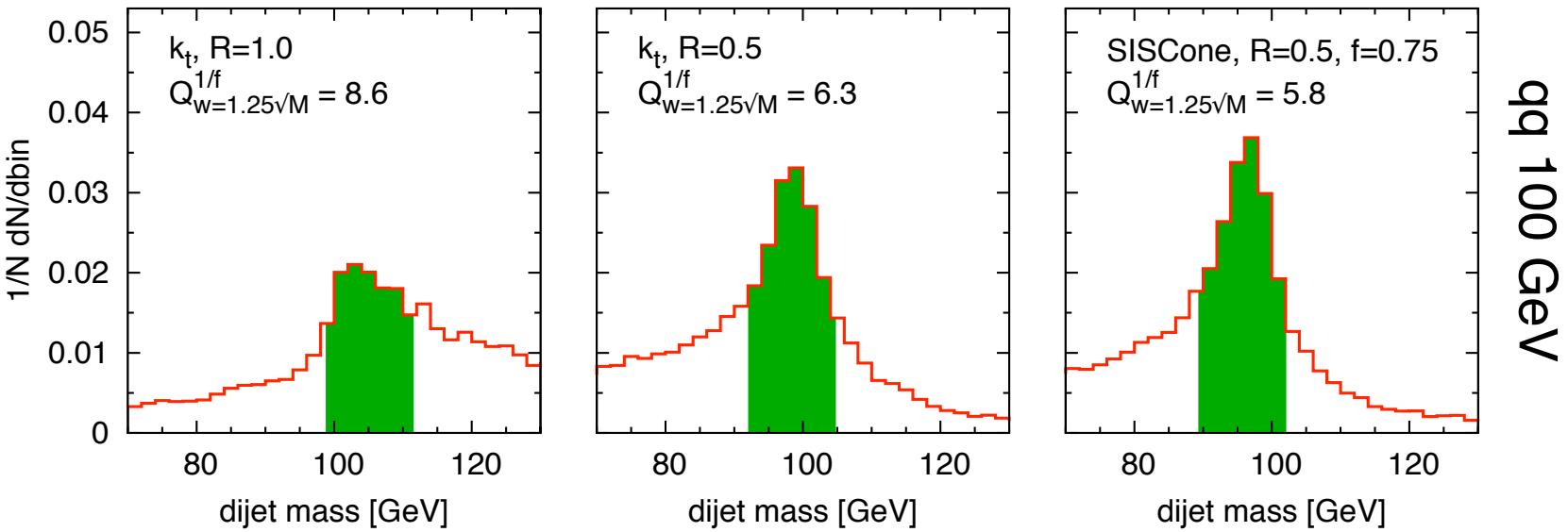
Smaller is better

Reconstruction of a di-jet mass peak

R=1: BAD

R=0.5: BETTER

MC, Rojo, Salam, Soyez, arXiv:0810.1304



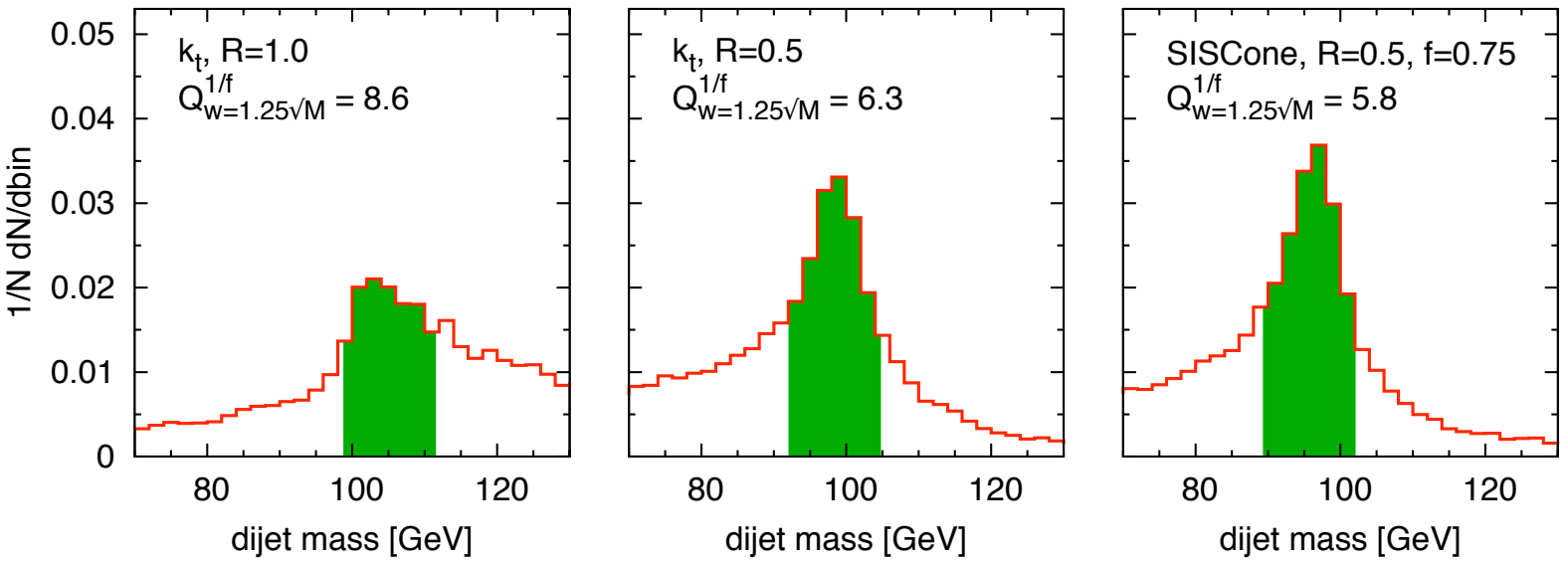
qq jets at 100 GeV

Reconstruction of a di-jet mass peak

MC, Rojo, Salam, Soyez, arXiv:0810.1304

R=1: BAD

R=0.5: BETTER

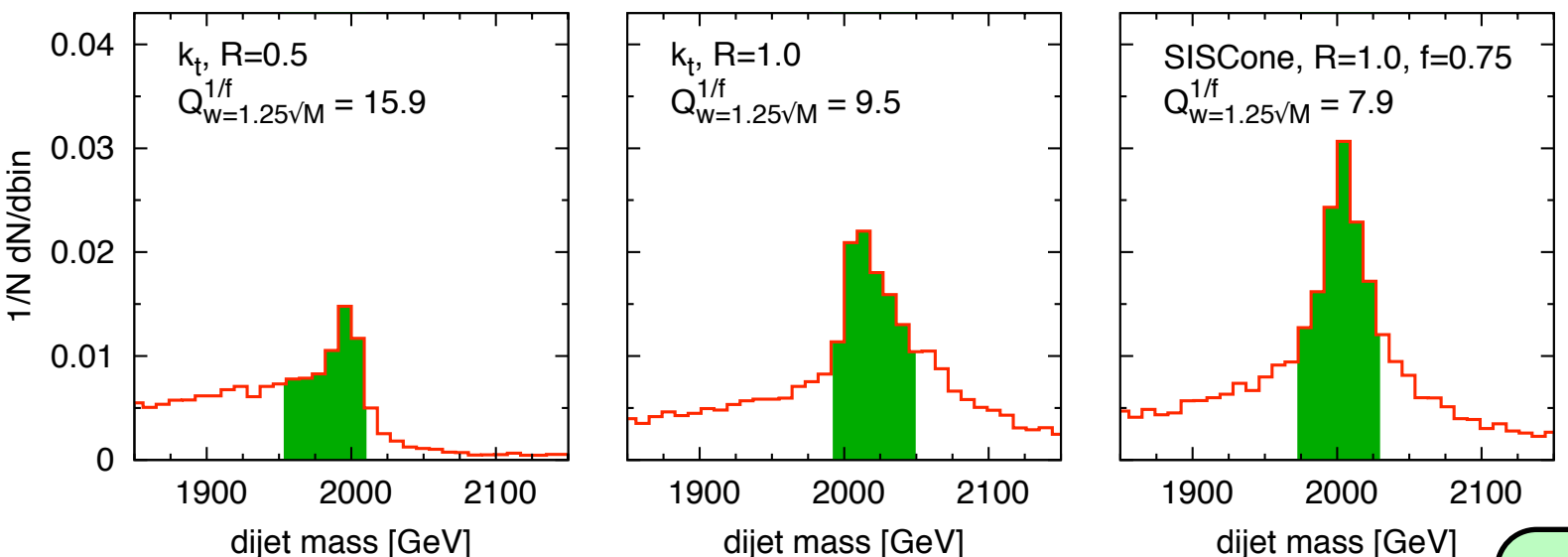


qq 100 GeV

qq jets at 100 GeV

R=0.5: BAD

R=1: BETTER



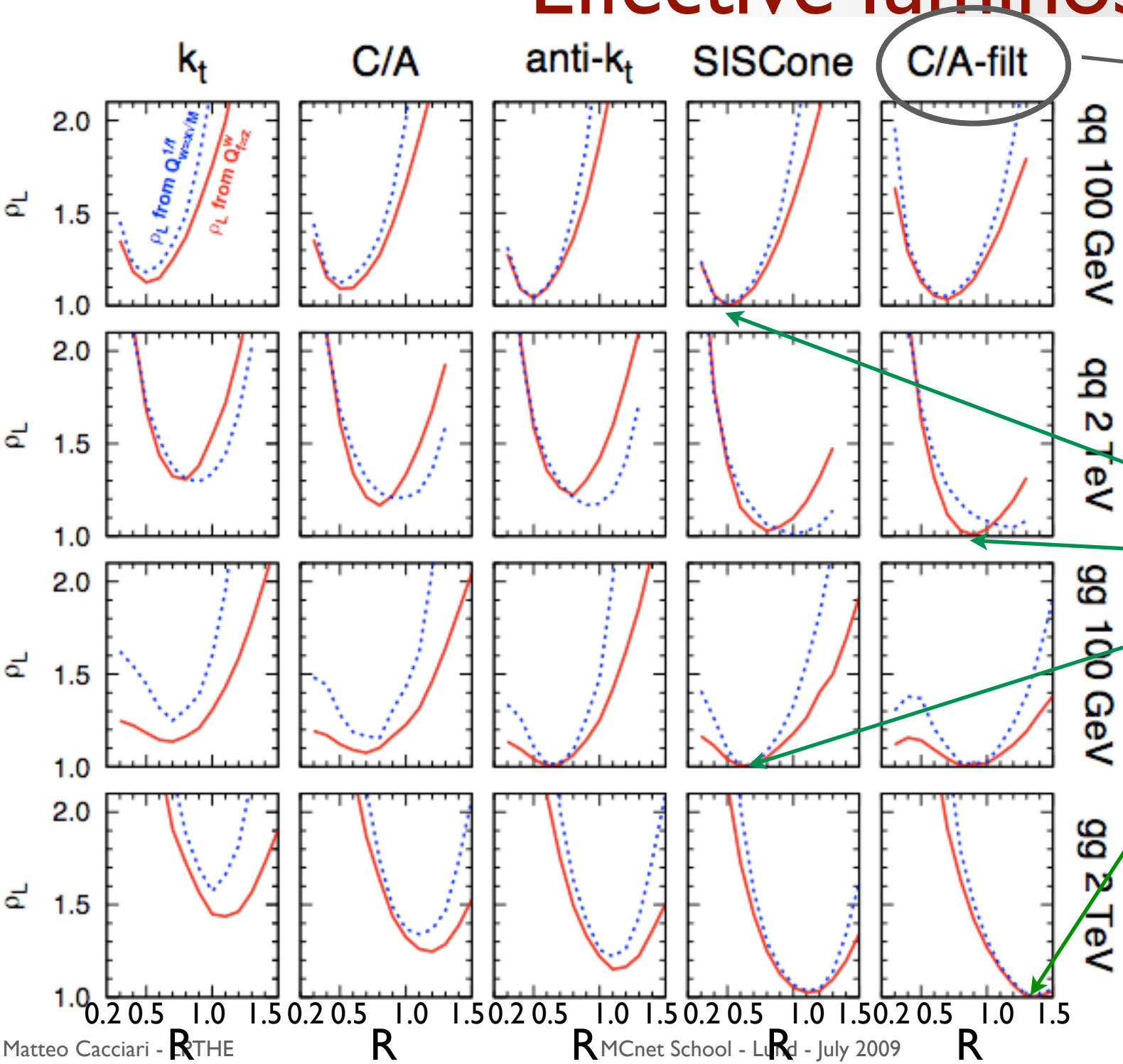
gg 2 TeV

gg jets at 2 TeV

Gluons (and heavy objects) prefer larger R

~ 100000 similar plots at http://quality.fastjet.fr

Effective luminosity ratios



Example of a 'third-generation' algorithm

Lower is better

Best

Different algs, different R, different performances

quality.fastjet.fr

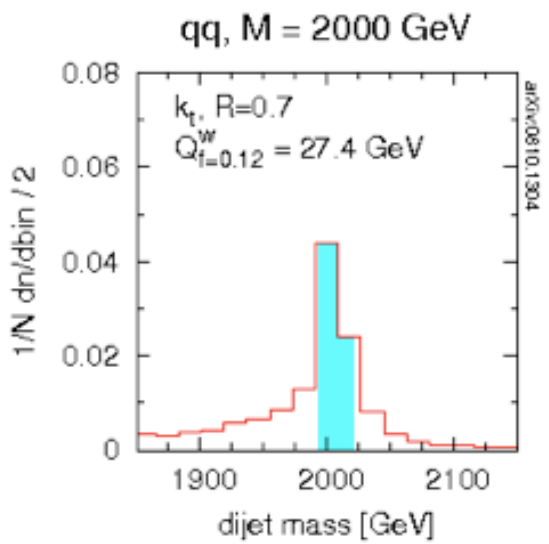
Do it yourself

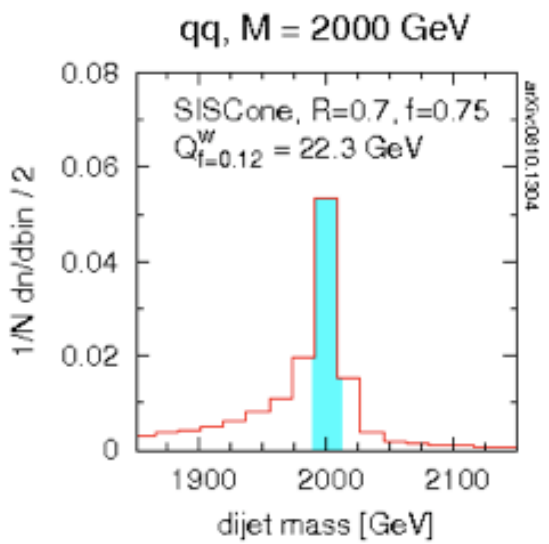
Testing jet definitions: qq & gg cases

http://quality.fastjet.fr

Testing jet definitions: qq & gg cases

by M. Cacciari, J. Rojo, G.P. Salam and G. Soyez, arXiv:0810.1304





This page is intended to help visualize how the choice of jet definition impacts a dijet invariant mass reconstruction at LHC.

The controls fall into 4 groups:

- *the jet definition*
- *the binning and quality measures*
- *the jet-type (quark, gluon) and mass scale*
- *pileup and subtraction*

The events were simulated with Pythia 6.4 (DWT tune) and reconstructed with FastJet 2.3.

*For more information, view and listen to the **flash demo**, or click on individual terms.*

This page has been tested with Firefox v2 and v3, IE7, Safari v3, Opera v9.5, Chrome 0.2.

Reset

k_t C/A anti-k_t SISCone C/A-filt	k_t C/A anti-k_t SISCone C/A-filt
− R = 0.7 + → all R	− R = 0.7 + → all R
$Q^w_{f=z}$ $Q^{1/f}_{w=x\sqrt{M}}$ x 2	$Q^w_{f=z}$ $Q^{1/f}_{w=x\sqrt{M}}$ x 2
− rebin = 2 +	− rebin = 2 +
qq gg	qq gg
− mass = 2000 +	− mass = 2000 +
pileup: none 0.05 0.25 mb^{-1}/ev	pileup: none 0.05 0.25 mb^{-1}/ev
subtraction:	subtraction:

Done

Jet areas: the link to physics

The definition of **active area** mimics the behaviour of the jet-clustering algorithms in the presence of a **large number of randomly distributed soft particles**

This is like underlying event or pileup!

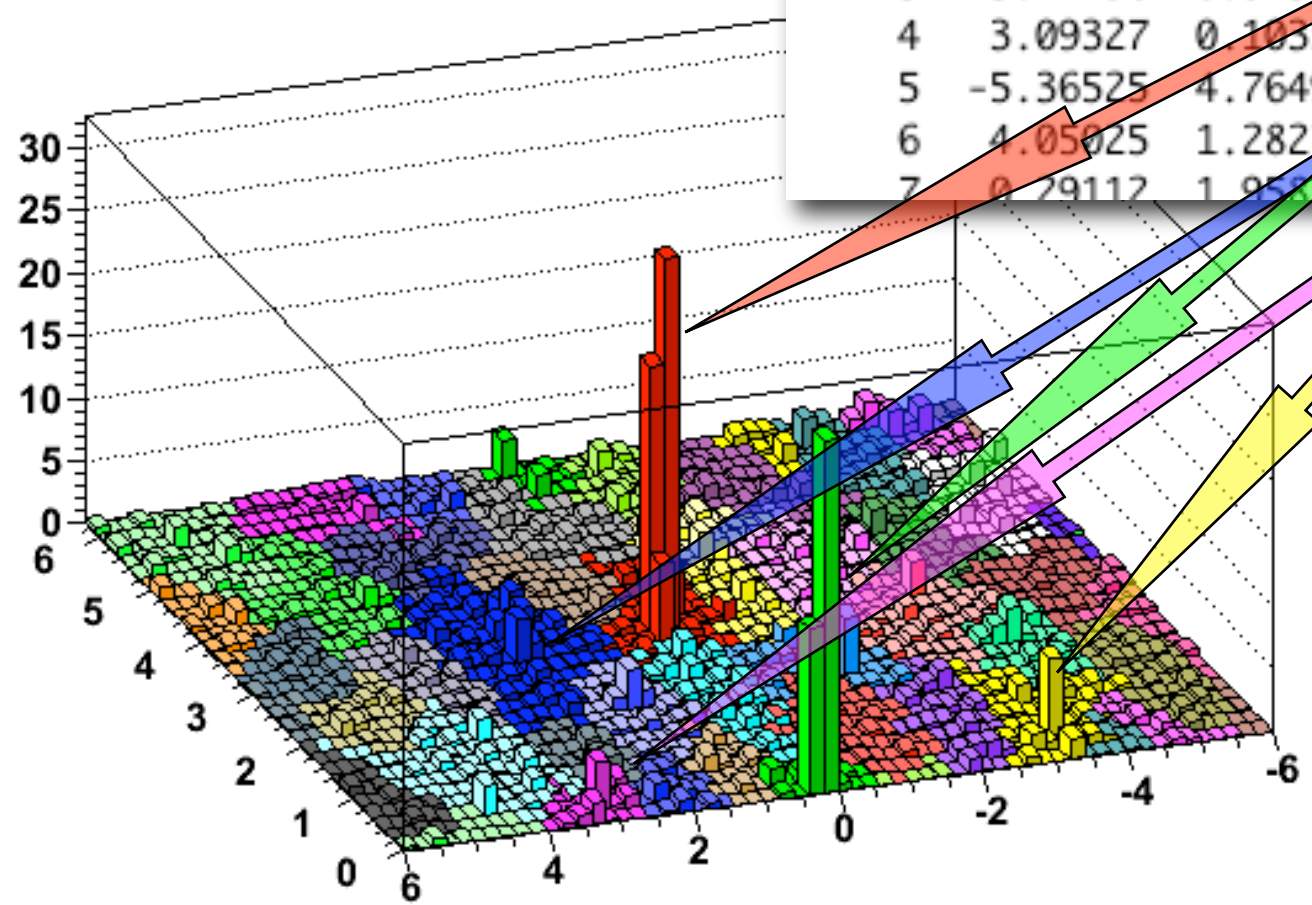
Tools needed to implement it:

1. An **infrared safe jet algorithm**
2. A reasonably **fast implementation**

Both are available

Jet active areas

```
iev 0 (irepeat 24): number of particles = 1428
strategy used =  NlnN
number of particles = 9051
Total area: 76.0265
Expected area: 76.0265
 ijet    eta       phi          Pt          area  +-    err
    0   0.15050  3.24498      69.970      2.625 +-  0.020
    1   0.18579  0.13150      59.133      1.896 +-  0.020
    2   2.33840  3.23960      31.976      4.749 +-  0.028
    3  -3.41796  0.52394      26.595      3.084 +-  0.021
    4   3.09327  0.10350      20.072      2.688 +-  0.023
    5  -5.36525  4.76491      19.598      2.780 +-  0.012
    6   4.05025  1.28270      15.361      3.592 +-  0.028
    7   0.29112  1.95635      14.566      2.114 +-  0.018
```



The ghost can also give you a visual impression of the reach of each jet

Background subtraction

[MC, Salam, arXiv:0707.1378]

If the area of each jet, and the momentum per unit area of the background are known, one can correct the transverse momentum of the hard jets:

$$p_T^{\text{hard jet, corrected}} = p_T^{\text{hard jet, raw}} - \rho \times \text{Area}_{\text{hard jet}}$$

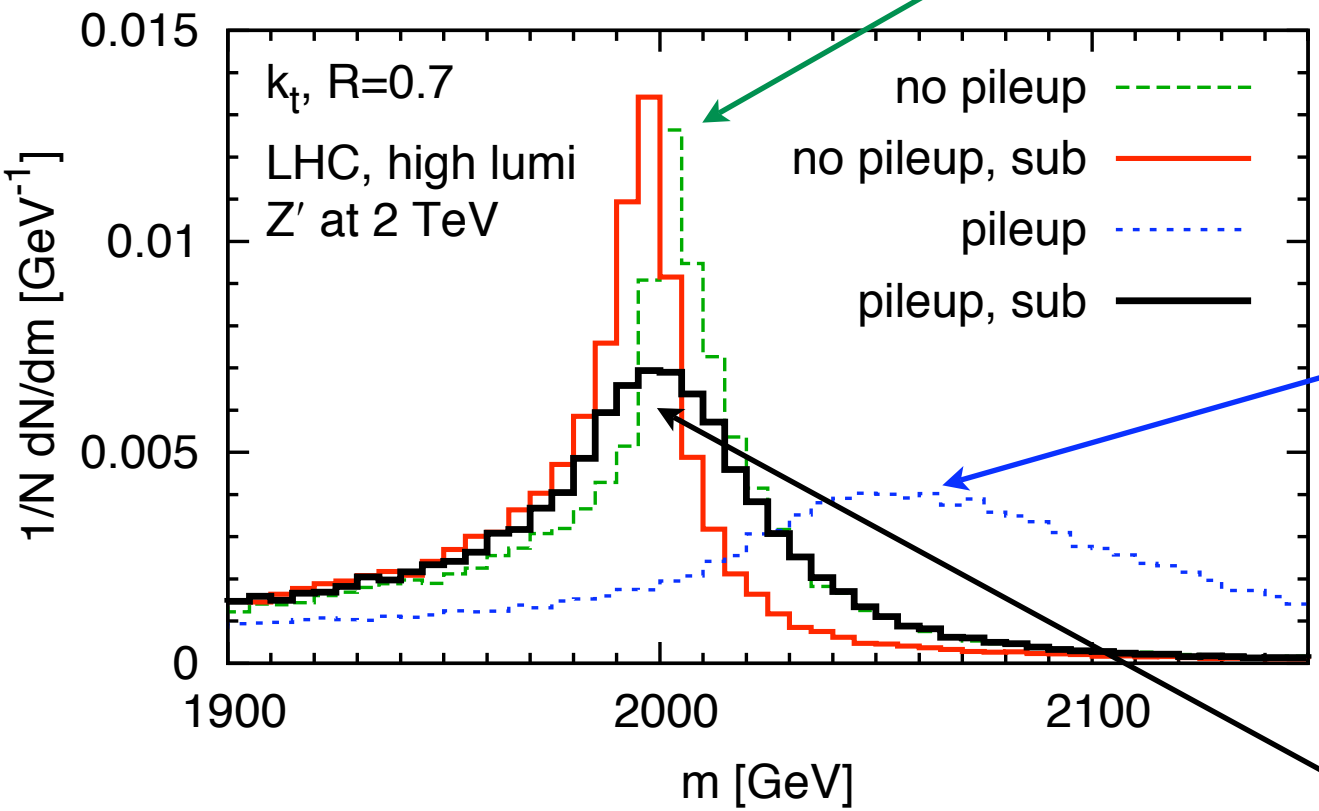
When ρ is calculated on an event-by-event basis, this procedure will generally improve the resolution of, say, a mass peak

NB. Also be(a)ware of *backreaction*
(immersing a hard jet in a soft background may cause some particles belonging to the hard event to be lost from (backreaction loss) or added to (backreaction gain) the jet).
Small effect for UE, larger for pileup, can be very important for heavy ions.
Analytical understanding of this effect available (MC, Salam, Soyez, arXiv:0802:1188)

Example of pileup subtraction

Let's discover a leptophobic Z' and measure its mass:

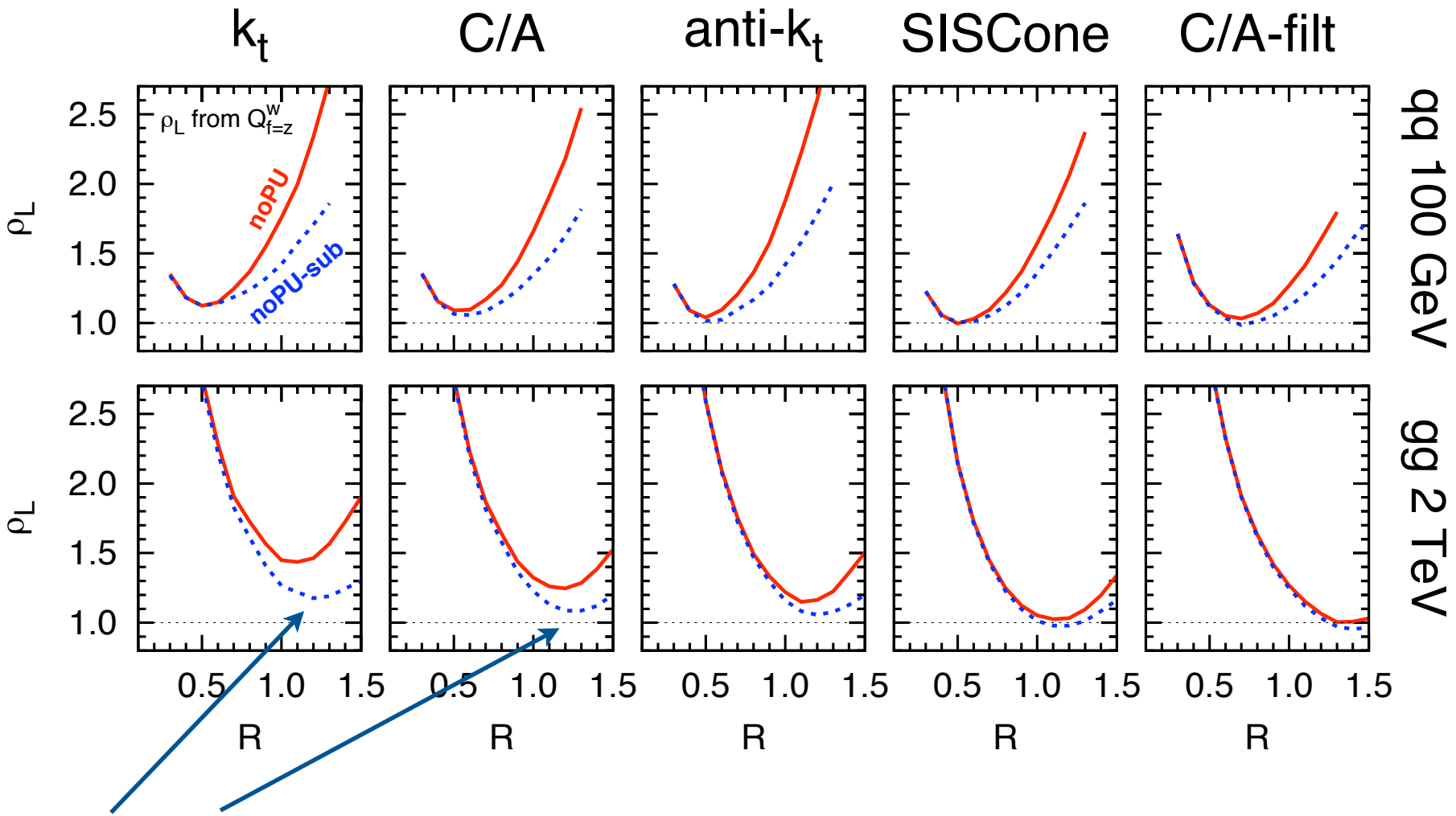
MC simulation:
m = 2000 GeV, width ~ 10 GeV



Naive measurement with PU:
m ~ 2050 GeV, width ~ 60 GeV

Measurement after subtraction:
m ~ 2000 GeV, width ~ 25 GeV

Effect of UE subtraction



Effect of UE subtraction: bad jet definitions are improved.

Gain in effective luminosity* about 20-30%

(i.e. they were 'bad' due to a large extent to their behaviour with respect to the UE)

* ρ_L = factor of luminosity needed to obtain equivalent signal/background significance

Summary

Summary

- There are many cone-type jet algorithms, but probably a single IRC-safe one

Summary

- There are many cone-type jet algorithms, but probably a single IRC-safe one
- There are also many recombination-type algorithms, usually IRC-safe

Summary

- There are many cone-type jet algorithms, but probably a single IRC-safe one
- There are also many recombination-type algorithms, usually IRC-safe
  - Their results will normally be 'similar', especially on inclusive quantities. However, if you plan to compare to higher order pQCD you **must** use an IRC-safe algorithm

Summary

- There are many cone-type jet algorithms, but probably a single IRC-safe one
- There are also many recombination-type algorithms, usually IRC-safe
  - Their results will normally be 'similar', especially on inclusive quantities. However, if you plan to compare to higher order pQCD you **must** use an IRC-safe algorithm
- A jet algorithm complemented by its parameters and the recombination scheme is called a **jet definition**

Summary

- There are many cone-type jet algorithms, but probably a single IRC-safe one
- There are also many recombination-type algorithms, usually IRC-safe
  - Their results will normally be 'similar', especially on inclusive quantities. However, if you plan to compare to higher order pQCD you **must** use an IRC-safe algorithm
- A jet algorithm complemented by its parameters and the recombination scheme is called a **jet definition**
  - The proper choice of the parameters of a jet definition can considerably improve the sensitivity of an analysis

Summary

- There are many cone-type jet algorithms, but probably a single IRC-safe one
- There are also many recombination-type algorithms, usually IRC-safe
  - Their results will normally be 'similar', especially on inclusive quantities. However, if you plan to compare to higher order pQCD you **must** use an IRC-safe algorithm
- A jet algorithm complemented by its parameters and the recombination scheme is called a **jet definition**
  - The proper choice of the parameters of a jet definition can considerably improve the sensitivity of an analysis
- Third-generation algorithms (e.g. filtering) appear promising for analyses where the jet substraction plays a relevant role

Summary

- There are many cone-type jet algorithms, but probably a single IRC-safe one
- There are also many recombination-type algorithms, usually IRC-safe
  - Their results will normally be 'similar', especially on inclusive quantities. However, if you plan to compare to higher order pQCD you **must** use an IRC-safe algorithm
- A jet algorithm complemented by its parameters and the recombination scheme is called a **jet definition**
  - The proper choice of the parameters of a jet definition can considerably improve the sensitivity of an analysis
- Third-generation algorithms (e.g. filtering) appear promising for analyses where the jet substraction plays a relevant role
- Jet areas and subtraction are tools whose full potential has probably not yet been explored and exploited