Graduate School Mass, Spectra, Symmetry
Spring Block Course 2012
March 2012, Bad Schandau/Krippen

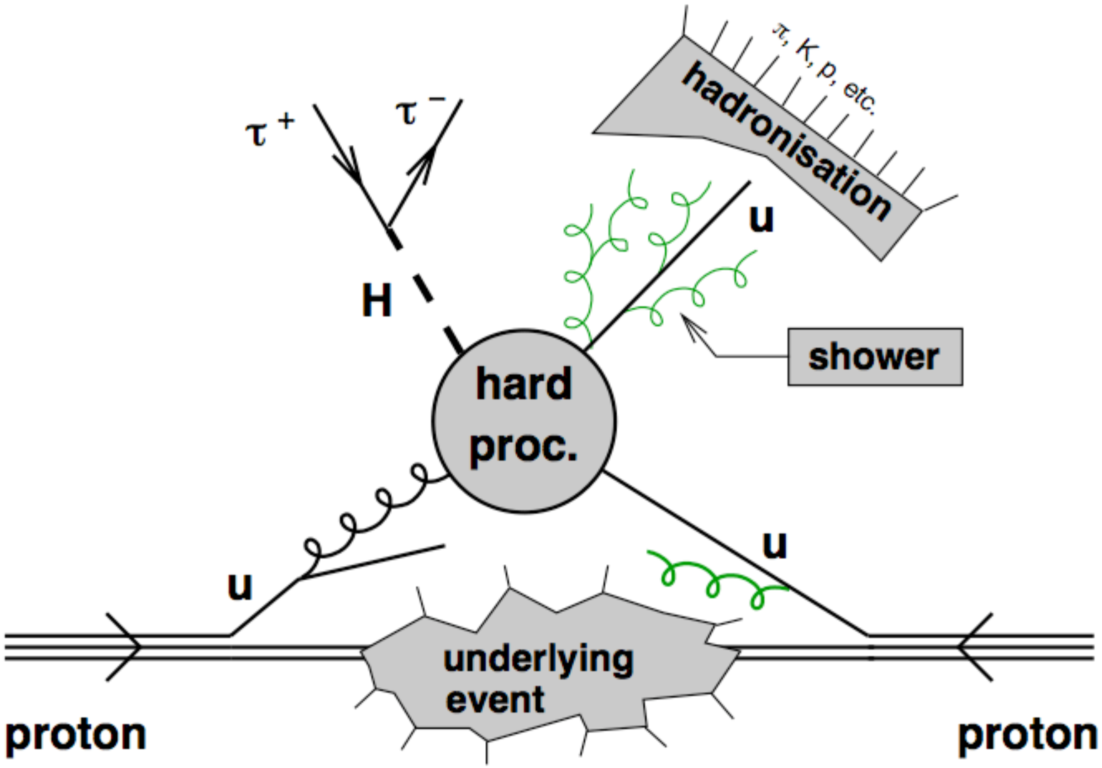
QCD for the LHC

Matteo Cacciari
LPTHE Paris

Part 3

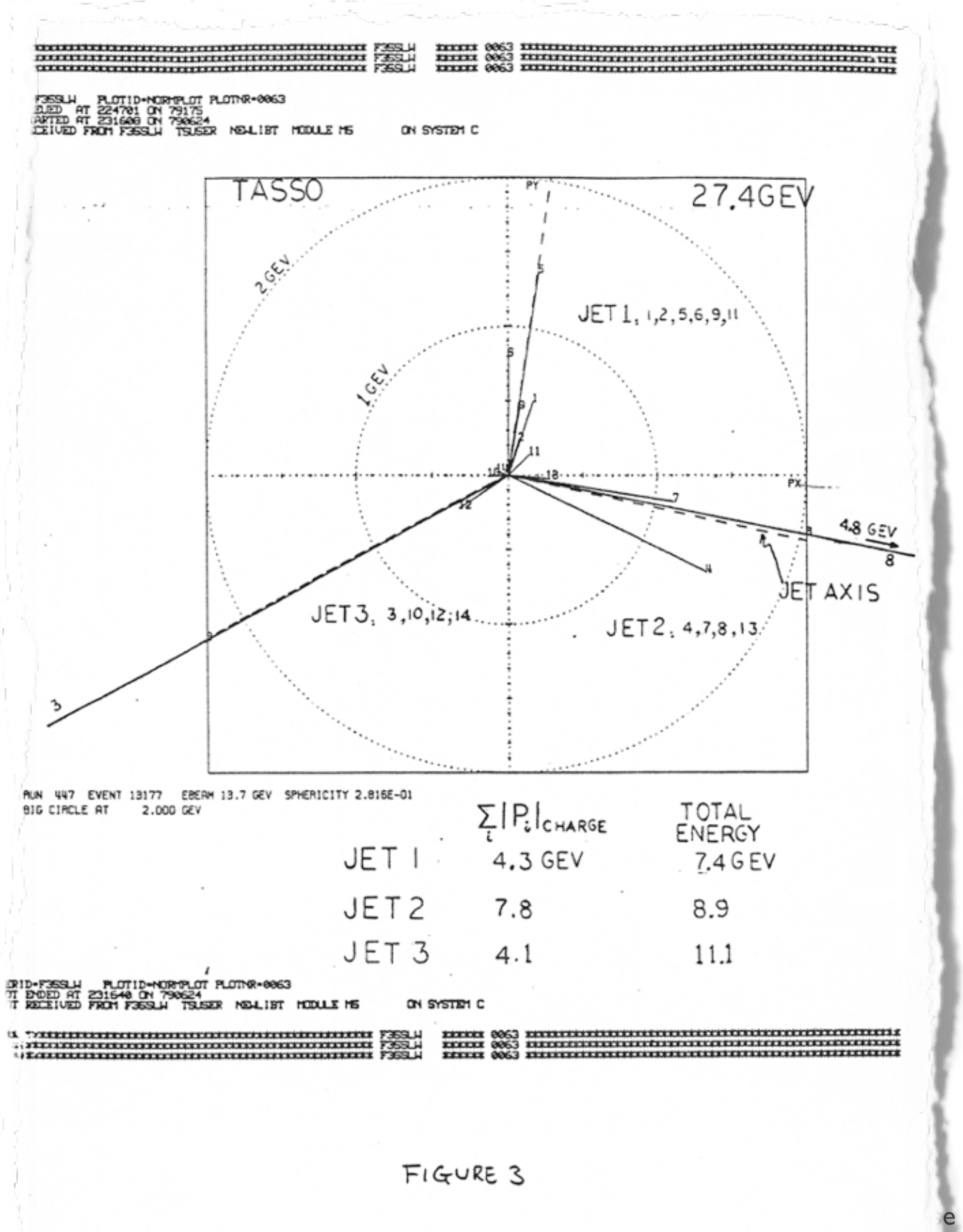
Many thanks to Guenther Dissertori, Rikkert Frederix, Fabio Maltoni, Paolo Nason, Gavin Salam, Maria Ubiali, and probably others, from whose talks/lectures I have drawn inspiration, as well as extracted many slides

Ingredients and tools



- PDFs
- Hard scattering
- Final state tools

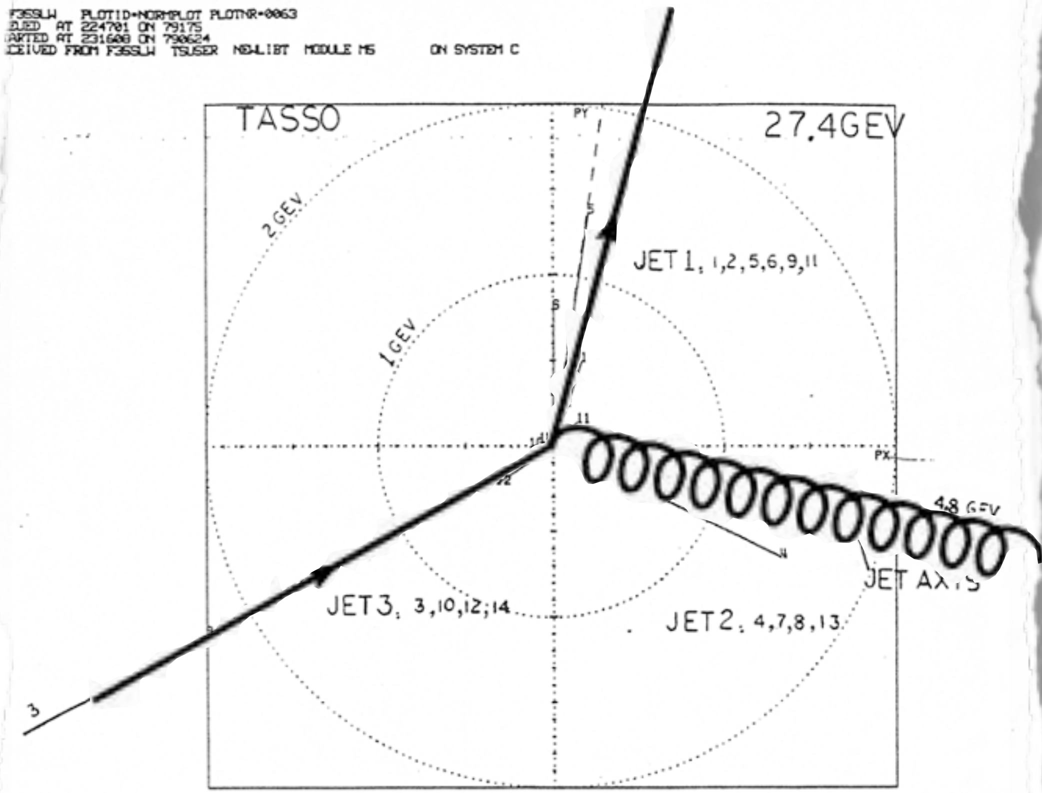
Gluon ‘discovery’



1979:

Three-jet events observed by TASSO, JADE, MARK J and PLUTO at PETRA in e^+e^- collisions at 27.4 GeV

Gluon 'discovery'



	$\sum_i \lvert P_i \rvert_{CHARGE}$	TOTAL ENERGY
JET 1	4.3 GEV	7.4 GEV
JET 2	7.8	8.9
JET 3	4.1	11.1

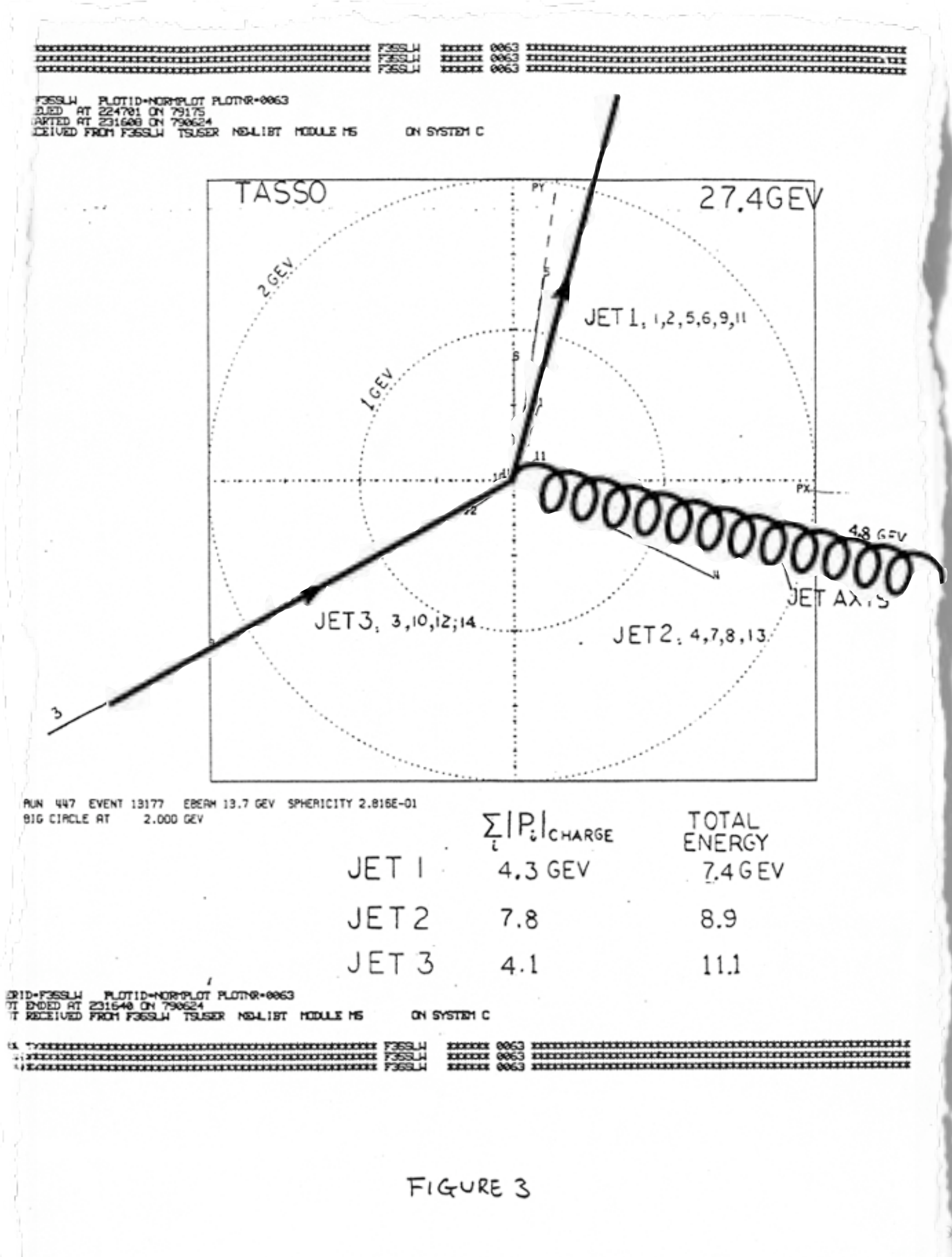
FIGURE 3

1979:

Three-jet events observed by TASSO, JADE, MARK J and PLUTO at PETRA in e^+e^- collisions at 27.4 GeV

Interpretation:
large angle emission of a hard gluon

Gluon ‘discovery’

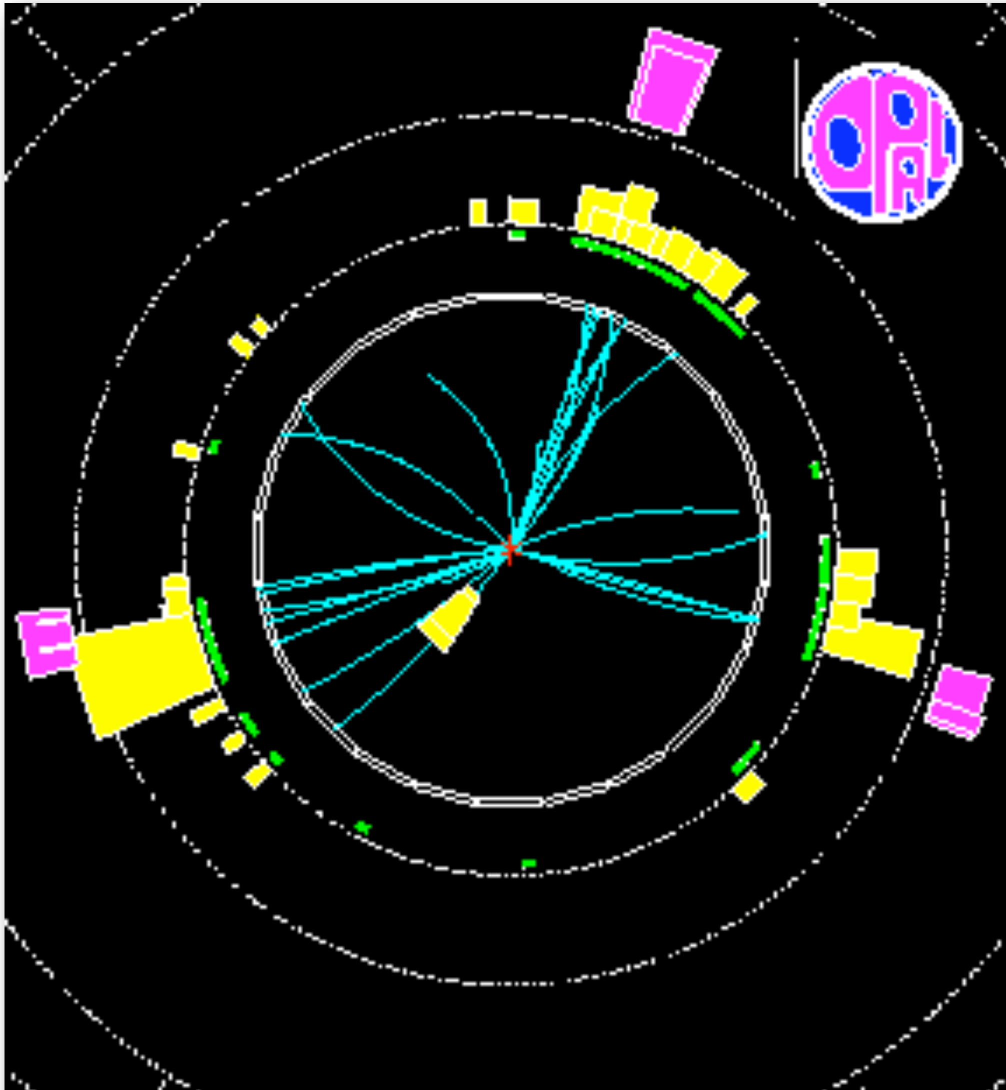


1979:

Three-jet events observed by TASSO, JADE, MARK J and PLUTO at PETRA in e^+e^- collisions at 27.4 GeV

Interpretation:
large angle emission of a hard gluon

Jets viewed as a proxy to the initial partons



From PETRA to LEP

A **jet** is something that happens in high energy events:

a collimated bunch of hadrons flying roughly in the same direction

(though, in the following, we'll extend this intuitive definition somewhat)

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

$$\{p_i\} \xrightarrow{\text{jet algorithm}} \{j_k\}$$

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

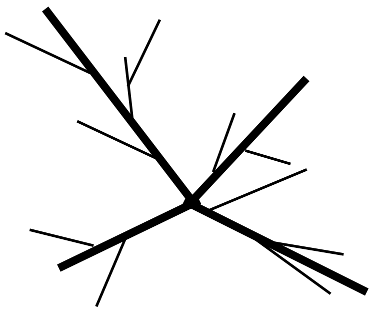
jets

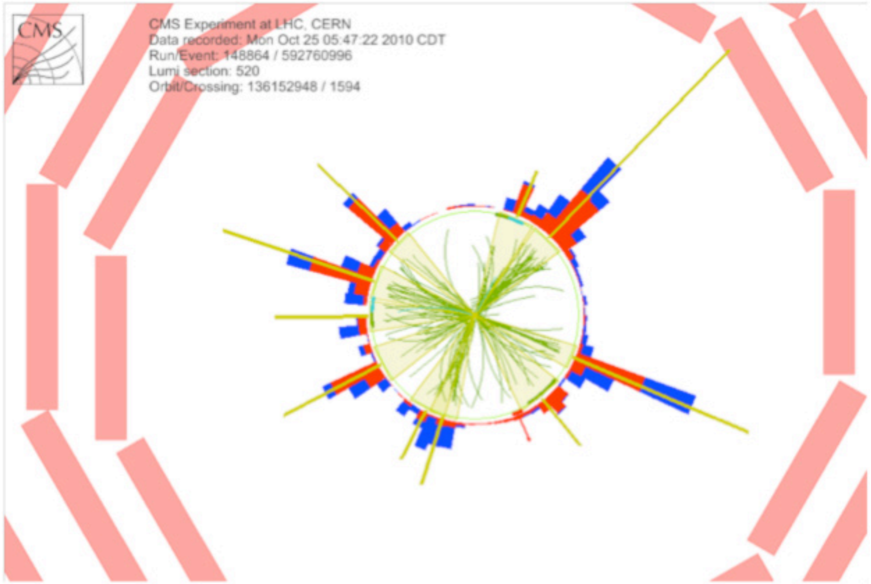
Most algorithms contain a resolution parameter, **R**, which controls the extension of the jet

(more about this later on)

Taming reality

Multileg + PS



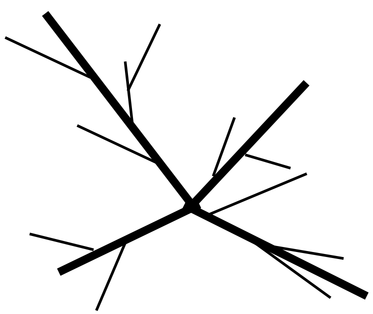


QCD predictions

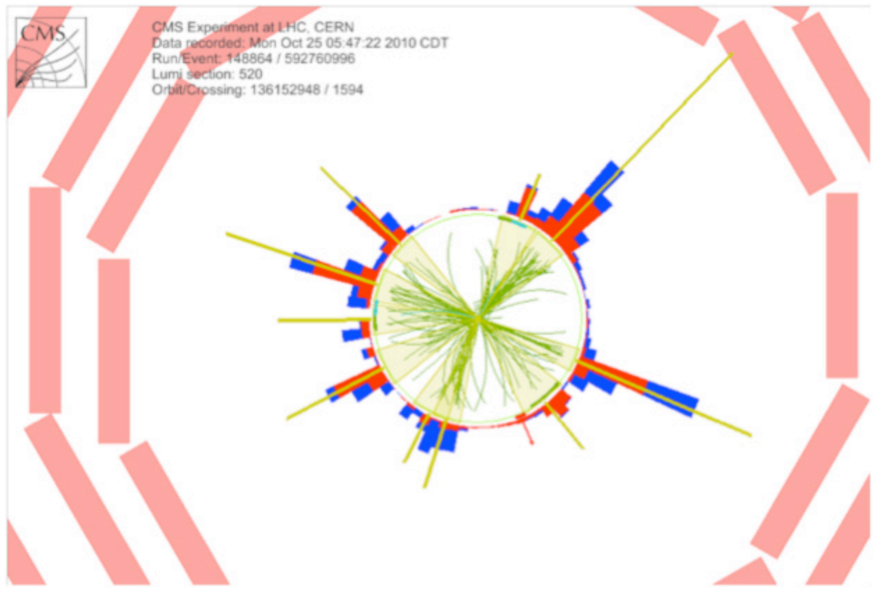
Real data

Taming reality

Multileg + PS



??



QCD predictions

Real data

Jets

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

Jets can serve two purposes

- They can be ***observables***, that one can measure and calculate
- They can be ***tools***, that one can employ to extract specific properties of the final state

Jet Definition

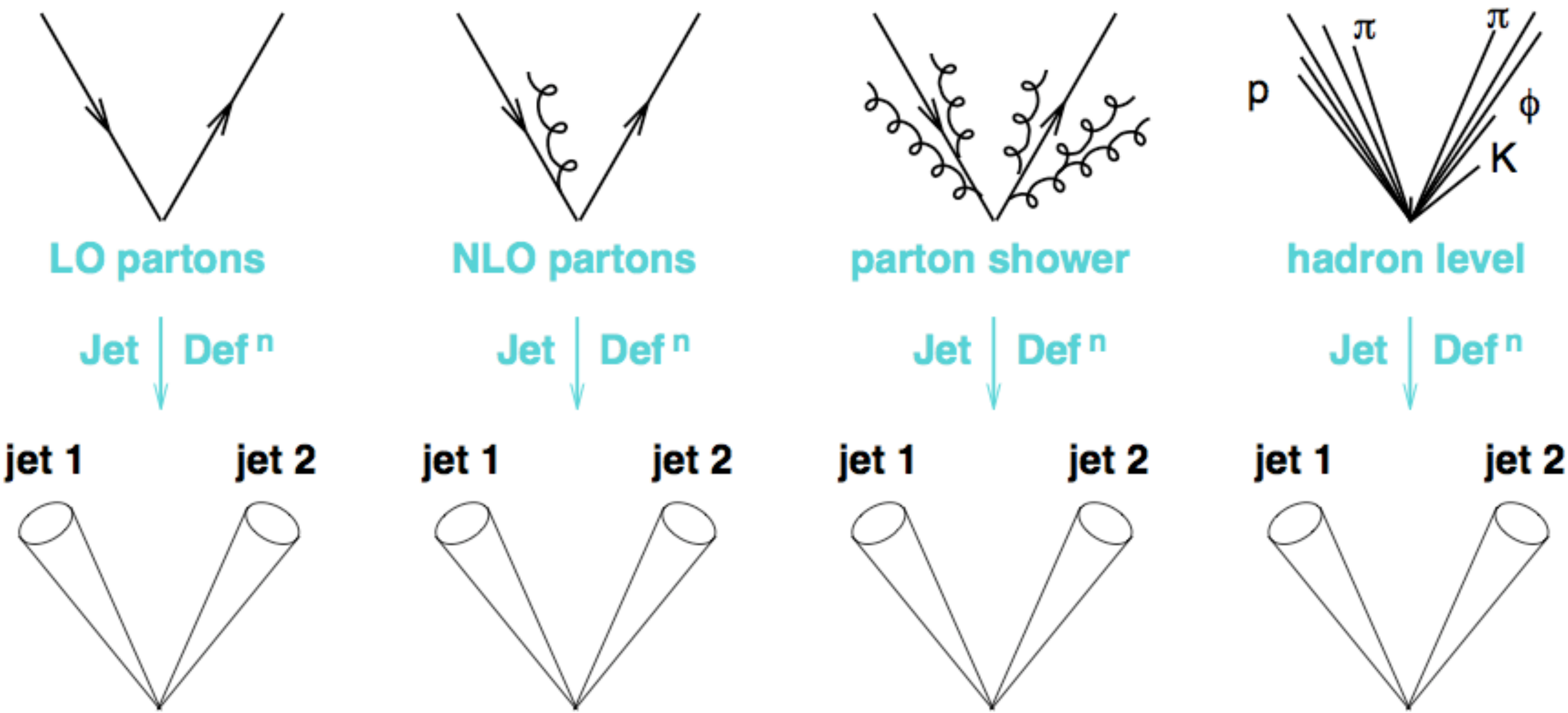
A jet algorithm
+
its parameters (e.g. R)
+
a recombination scheme
=
a **Jet Definition**

"Jet [definitions] are legal contracts between theorists and experimentalists"
-- MJ Tannenbaum

What makes a particular contract a **good** one?

Jets as proxies

A good jet definition should be resilient to QCD effects



NB. 'Resiliency' does not mean 'total insensitivity'

A 'hadron jet' is **not** a parton

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

Works 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)

Works because QCD only modifies energy flow on small scales

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

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)

FC-PR v. IC-PR

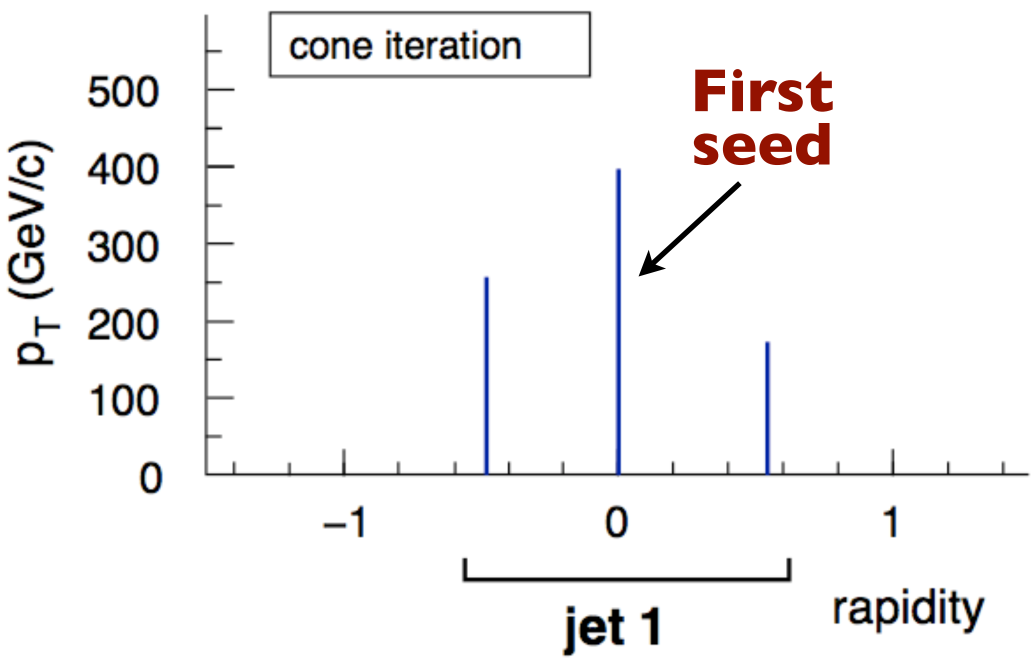
Iterative Cone with Progressive Removal (IC-PR)

(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

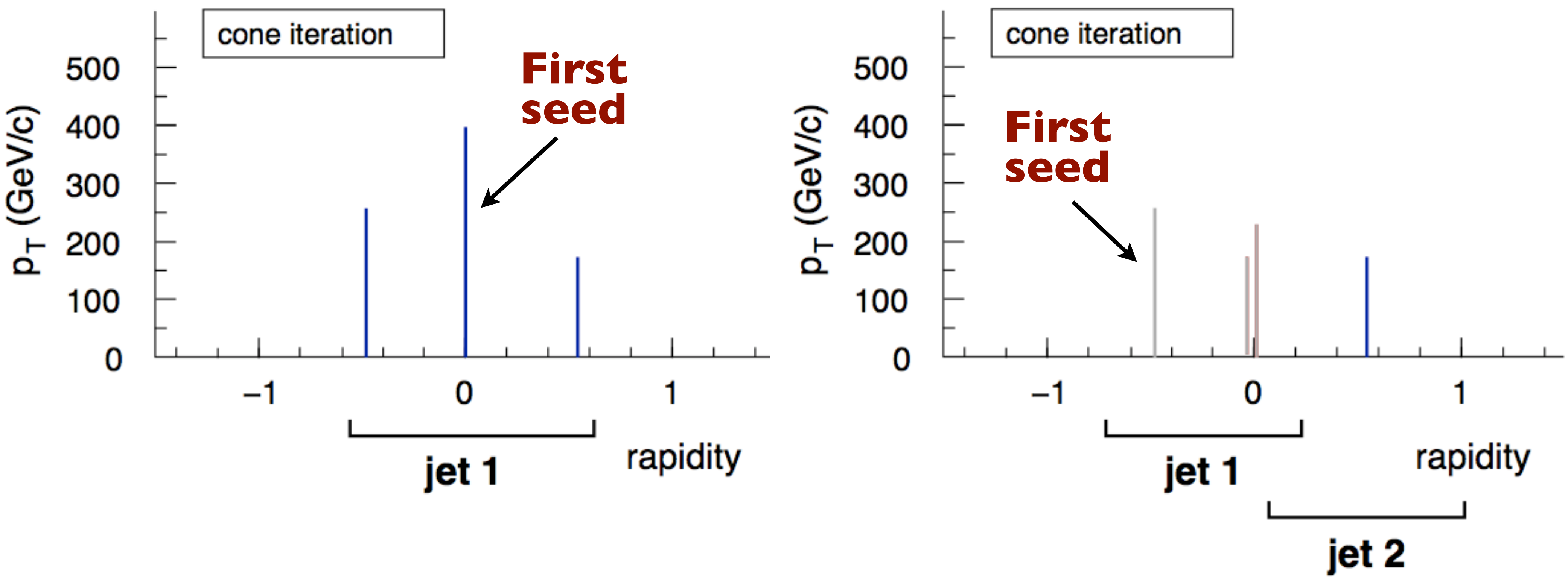
IC-PR cone collinear unsafety

A collinear splitting can change the final state



IC-PR cone collinear unsafety

A collinear splitting can change the final state

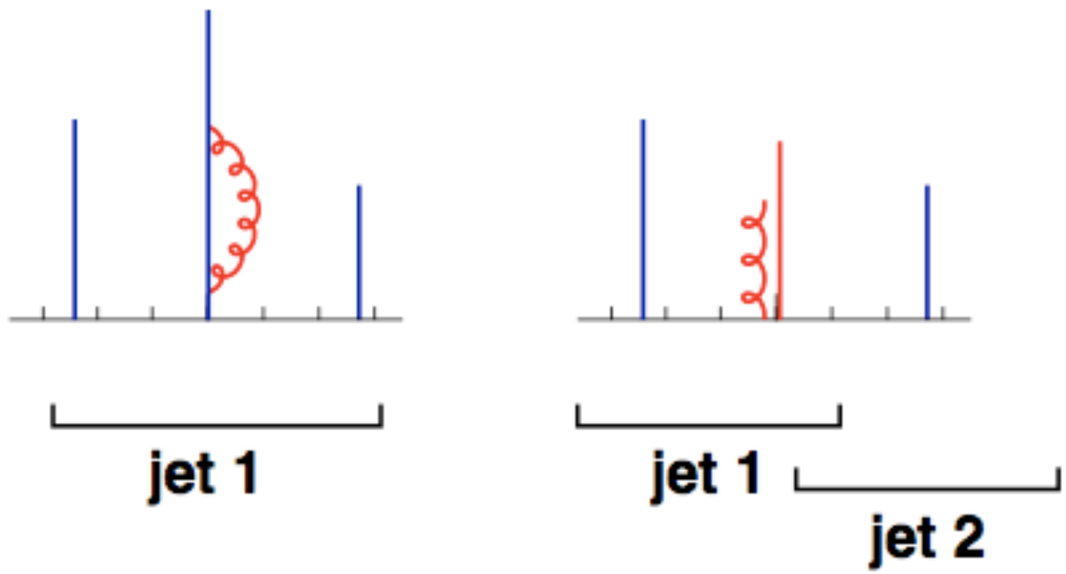


Splitting the hardest particle **collinearly** has changed the number of final jets

Consequences of collinear unsafety

In QCD perturbation theory, virtual and soft/collinear 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

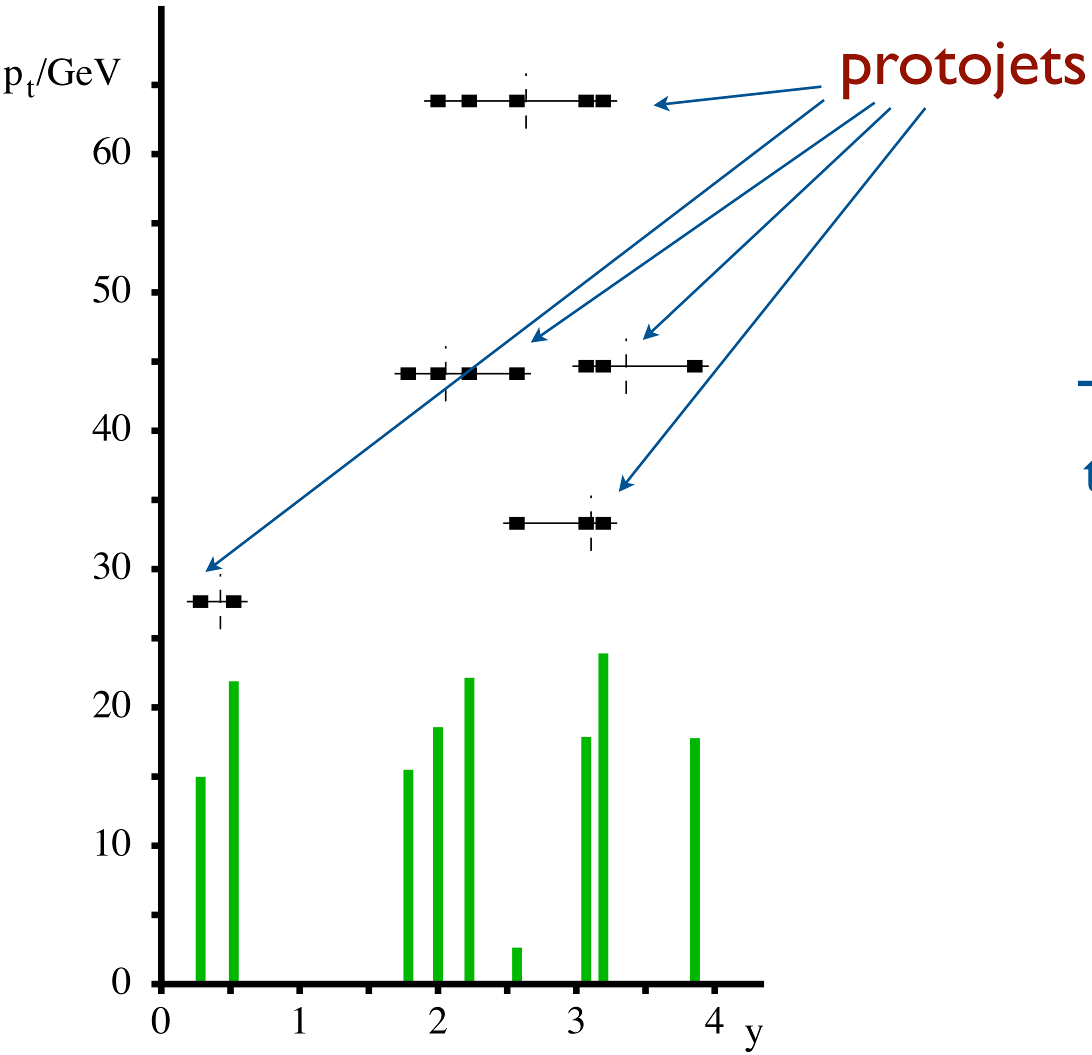
Iterative Cone with Split-Merge (IC-SM)

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



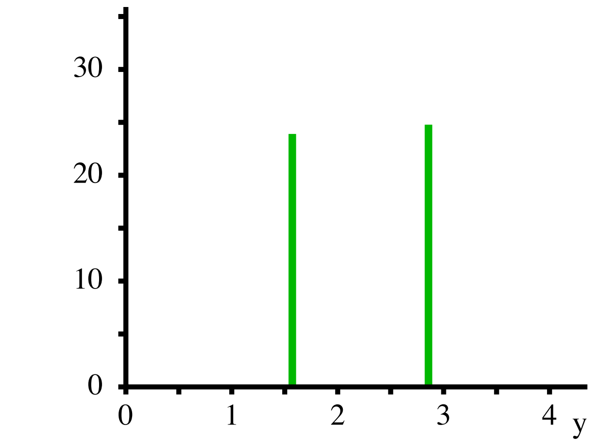
The lack of 'progressive removal' means that some protojets can be overlapping (i.e. contain the same particles). Must deal with this: **split-merge**

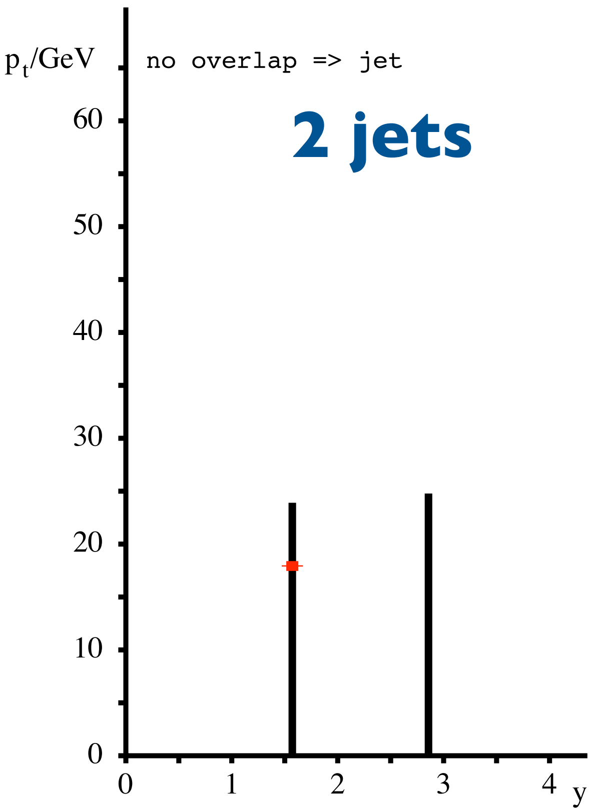
'Split-merge' is a further algorithm aimed at disentangling overlapping protojets.

The Tevatron Run II implementation goes like this:

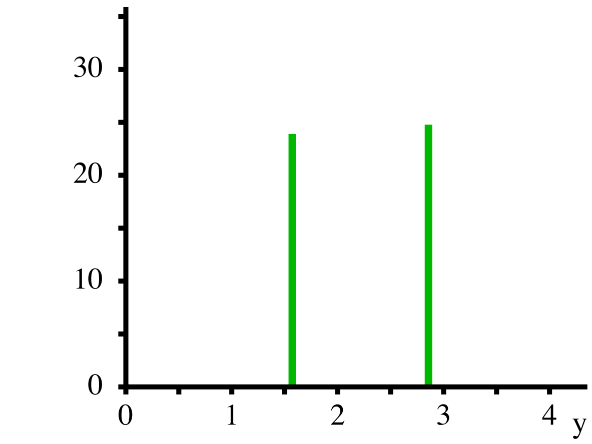
- Choose an **overlap threshold *f***
- Find hardest protojet
- Find hardest other protojet overlapping with it
- Merge is they share a fraction of momentum larger than *f*, split along axis at centre otherwise
- (Call protojet a jet if there are no overlapping protojets)

IC-SM infrared unsafety

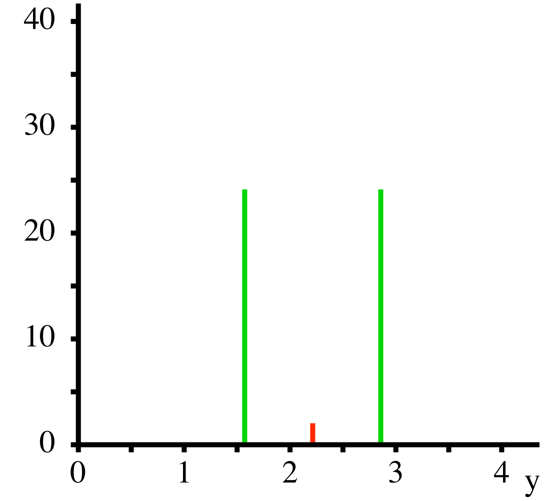


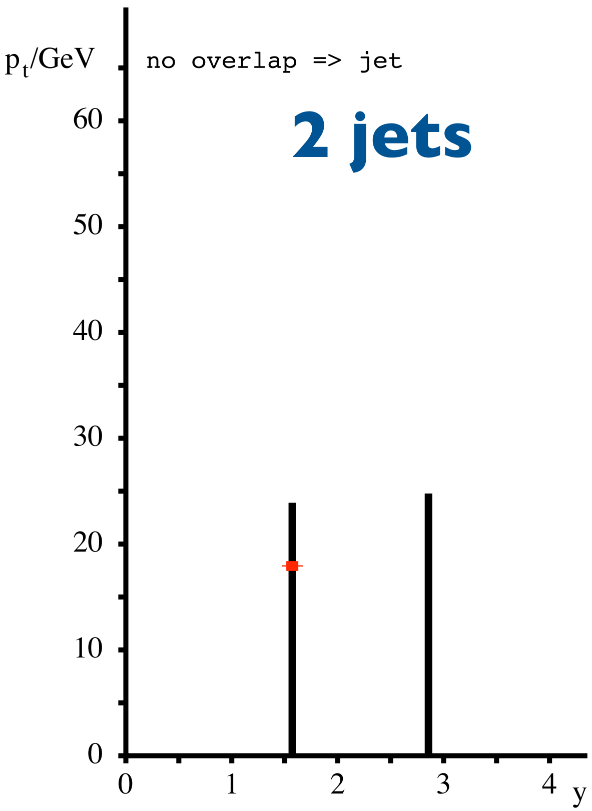


IC-SM infrared unsafety

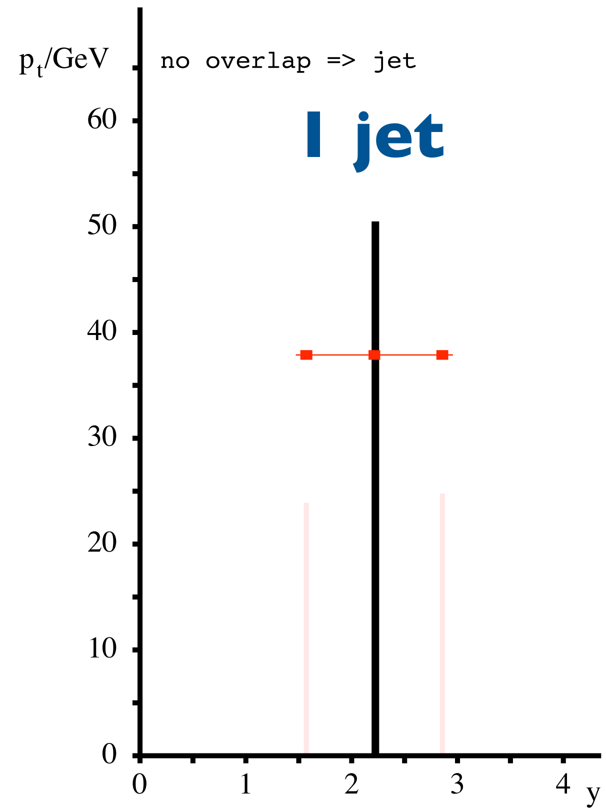


Add a
soft particle





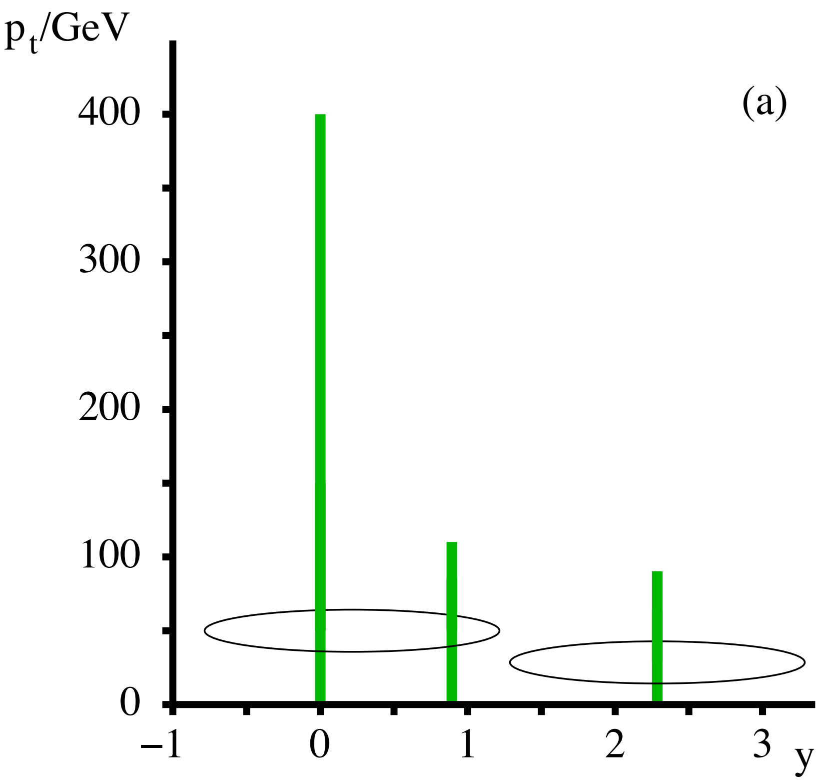
Final state jets differ



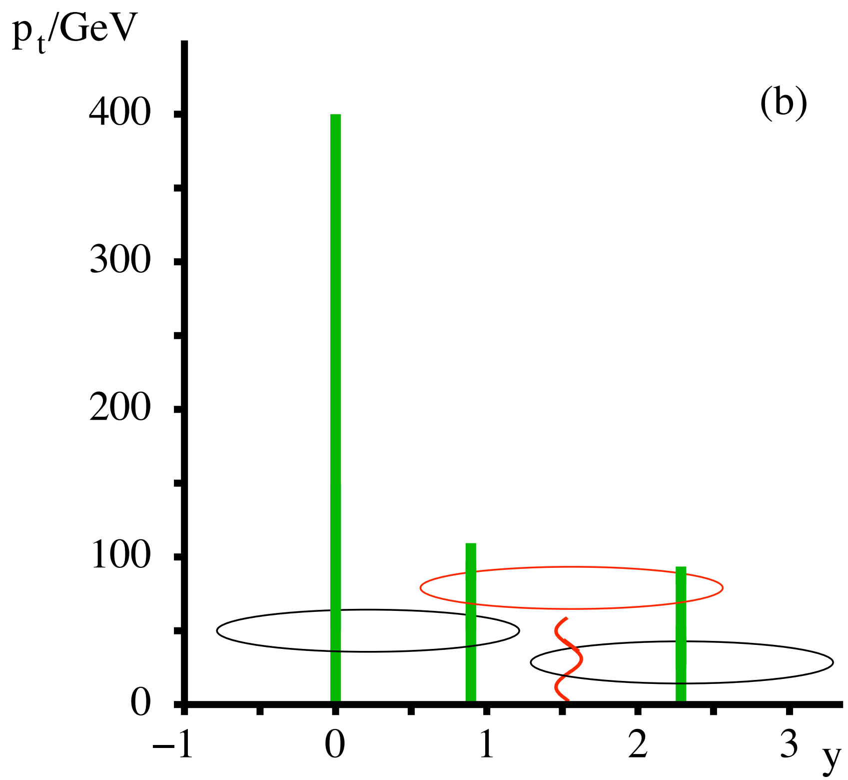
MidPoint (IC_{mp}-SM) infrared unsafety

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

Using unsafe jet tools essentially renders many QCD calculations useless

Good jet definitions become more and more important as event predictions have more and more substructure, as in higher order multileg calculations

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 theory/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

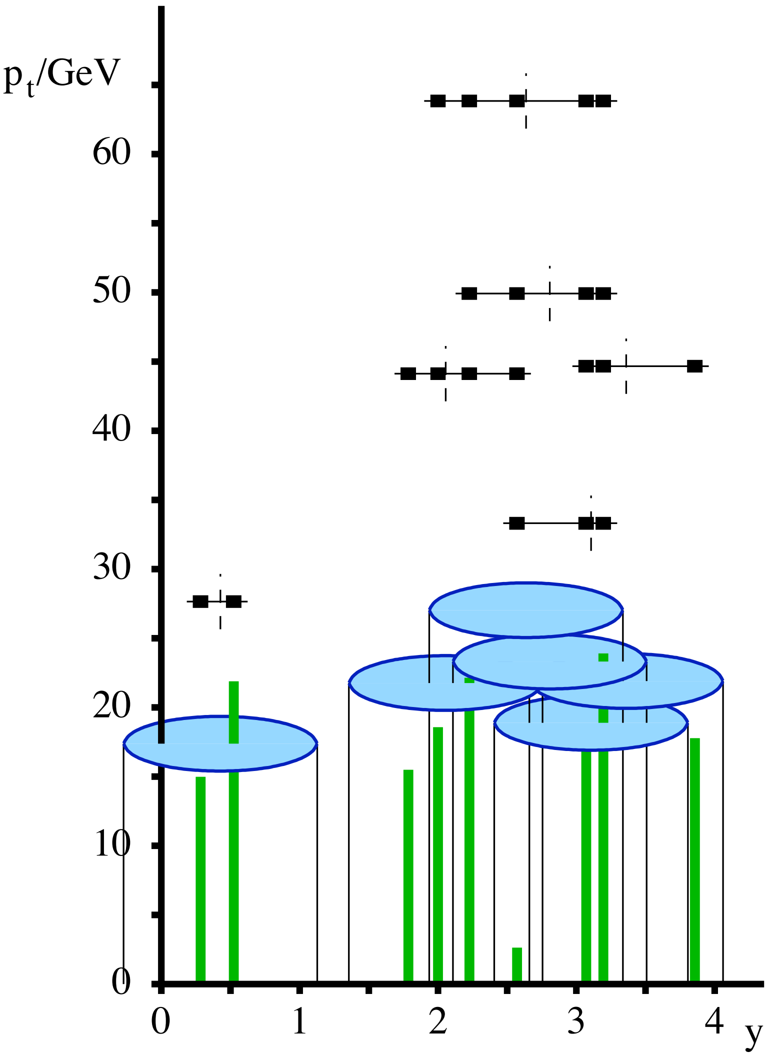
Unfortunately, this takes $N2^N$ operations:
the age of the universe for only 100 particles

Way out: a geometrical solution → SISCone

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

SISCone is guaranteed to find ALL the stable cones

These are **ALL** the stable cones



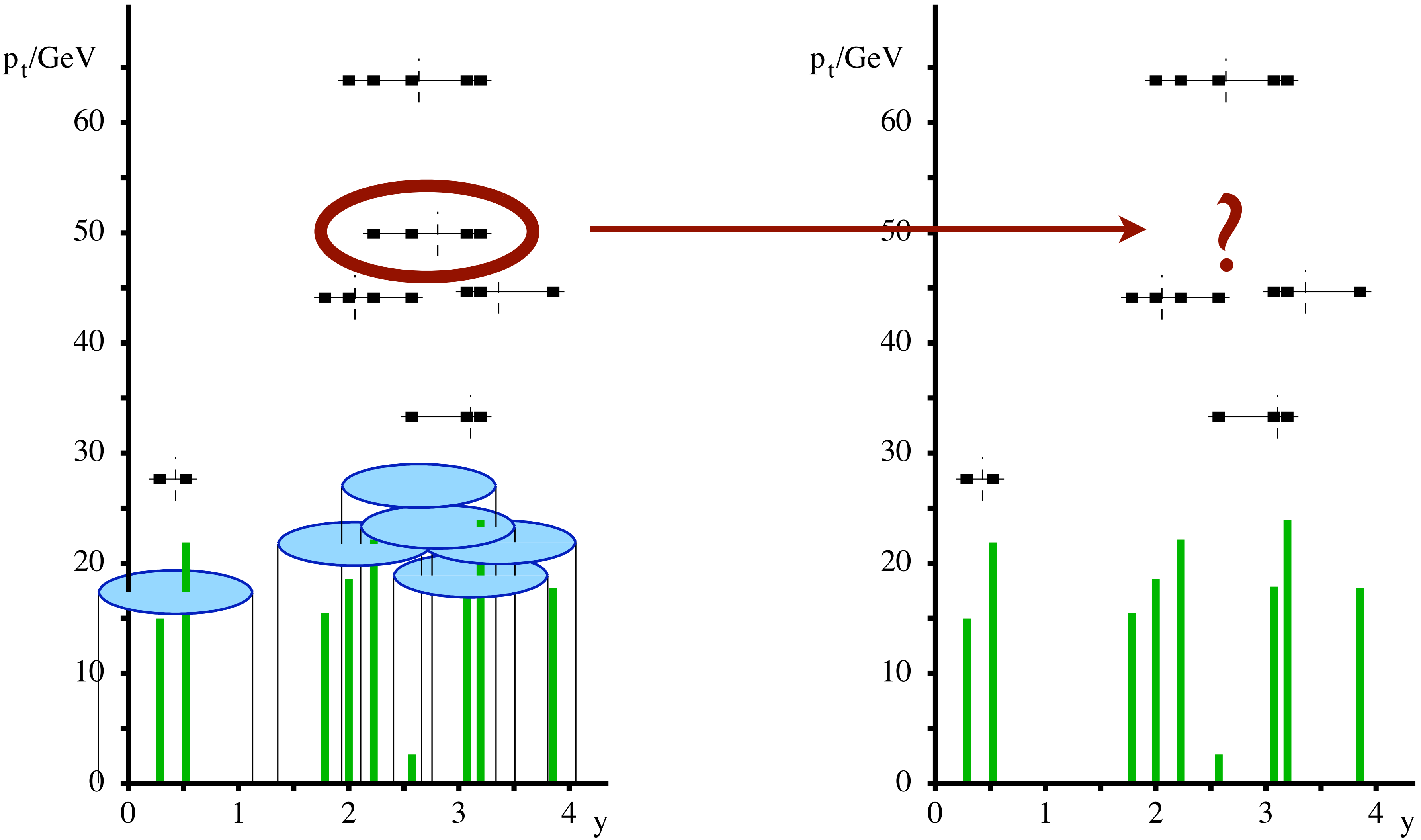
SISCone

IC-SM

SIScone v. IC-SM

These are **ALL** the stable cones

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



SISCone

IC-SM

Cones Infrared (un)safety

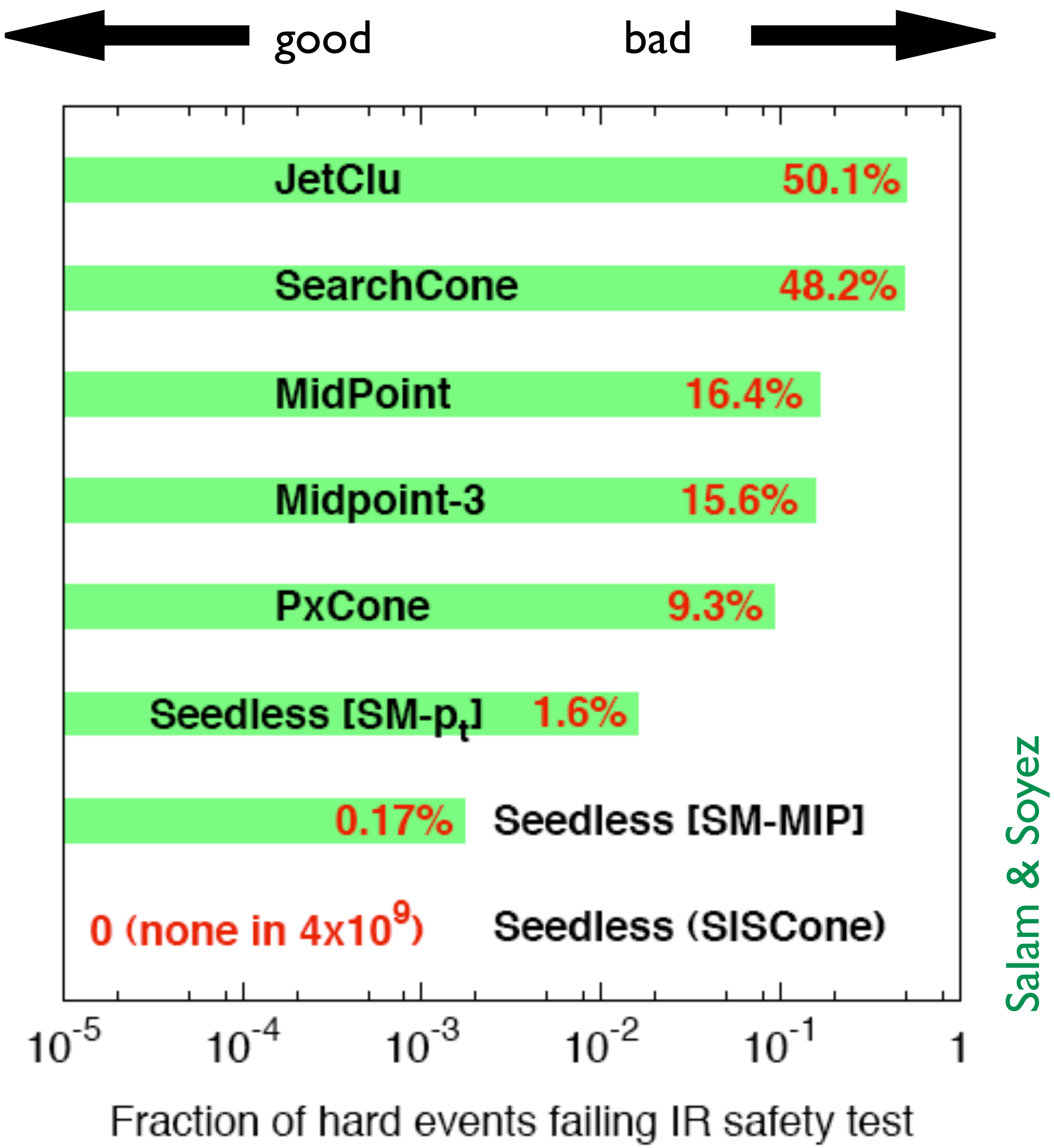
Q: How often are the hard jets changed by the addition of a soft particle?

- Generate event with $2 < N < 10$ hard particles, find jets
- Add $1 < N_{soft} < 5$ soft particles, find jets again [repeatedly]
- If the jets are different, algorithm is IR unsafe.

A:

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

- First introduced in e^+e^- collisions in the '80s
- Typically they work by calculating a **'distance'** between particles, and then recombine them pairwise according to a given order, until some condition is met (e.g. no particles are left, or the distance crosses a given threshold)

IRC safety can usually be seen to be trivially guaranteed

JADE algorithm

distance:

$$y_{ij} = \frac{2E_iE_j(1-\cos\theta_{ij})}{Q^2}$$

- Find the minimum y_{min} of all y_{ij}
- If y_{min} is below some jet resolution threshold y_{cut}, recombine i and j into a single new particle ('pseudojet'), and repeat
- If no $y_{min} < y_{cut}$ are left, all remaining particles are jets

Problem of this particular algorithm:
two soft particles emitted at large angle get easily recombined into a single jet:
counterintuitive and perturbatively troublesome

e^+e^- k_t (Durham) algorithm

[Catani, Dokshitzer, Olsson, Turnock, Webber '91]

Identical to JADE, but with distance:

$$y_{ij} = \frac{2\min(E_i^2, E_j^2)(1-\cos\theta_{ij})}{Q^2}$$

In the collinear limit, the numerator reduces to the **relative transverse momentum** (squared) of the two particles, hence the name of the algorithm

The use of the min() avoids the problem of recombination of back-to-back particles present in JADE: a soft and a hard particle close in angle are 'closer' than two soft ones at large angle

One key feature of the k_t algorithm is its relation to the structure of QCD divergences:

$$\frac{dP_{k\to ij}}{dE_i d\theta_{ij}} \sim \frac{\alpha_s}{\min(E_i, E_j)\theta_{ij}}$$

The k_t algorithm inverts the QCD branching sequence (the pair which is recombined first is the one with the largest probability to have branched)

k_t algorithm in hadron collisions

(Inclusive and longitudinally invariant version)

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

- Calculate the distances between the particles: $\mathbf{d_{ij}}$
- Calculate the beam distances: $\mathbf{d_{iB}}$
- Combine particles with **smallest distance** d_{ij} or, if d_{iB} is smallest, call it a jet
- Find again smallest distance and repeat procedure until no particles are left (this stopping criterion leads to the *inclusive* version of the k_t algorithm)

Given N particles this is, naively, an $O(N^3)$ algorithm: calculate N^2 distances, repeat for all N iterations

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

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	$N^2 \ln N$

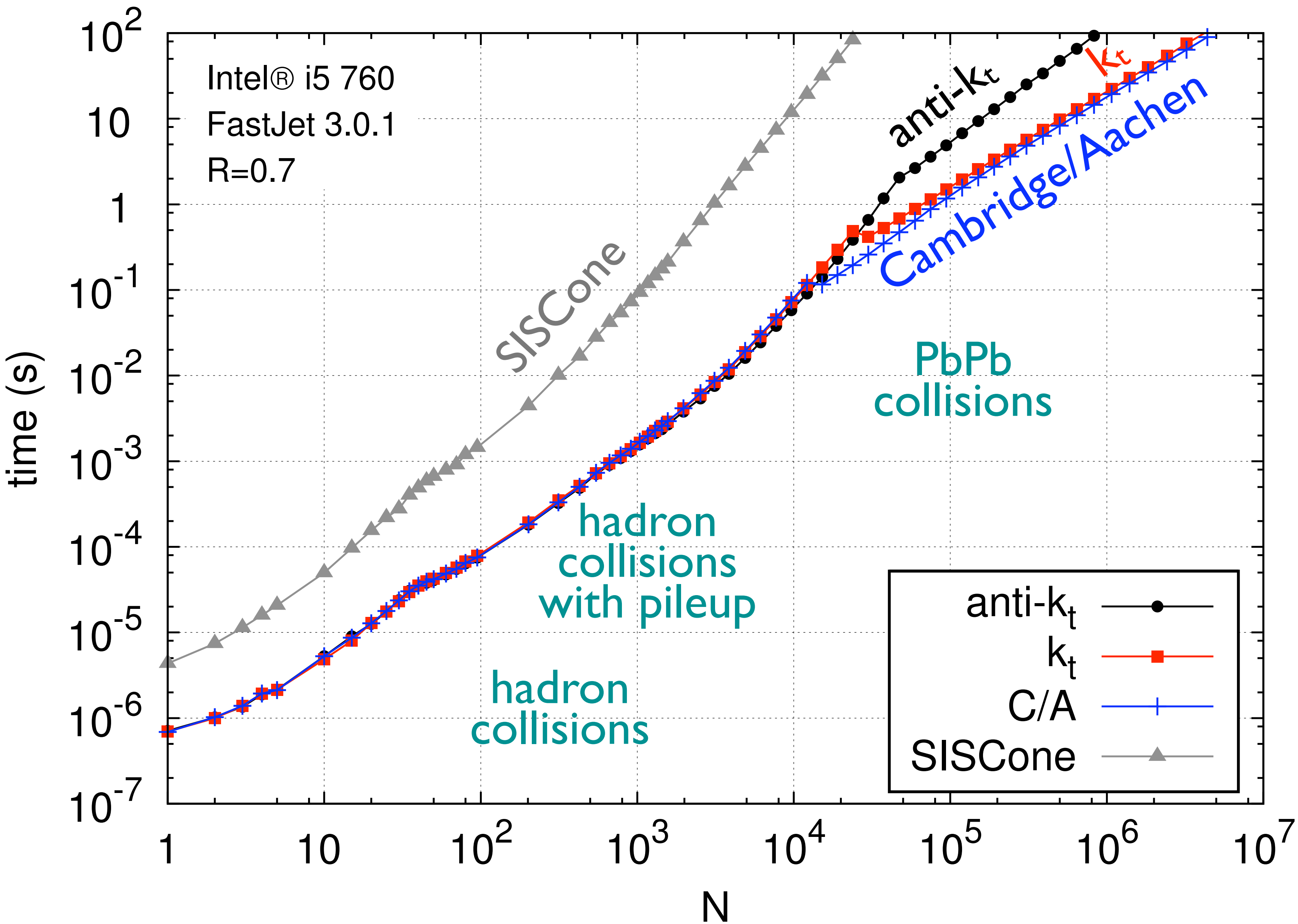
'second-generation' algorithms

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

(As well as many IRC unsafe ones)

FastJet speed test

Time needed to cluster an event with N particles



Jets’ reach

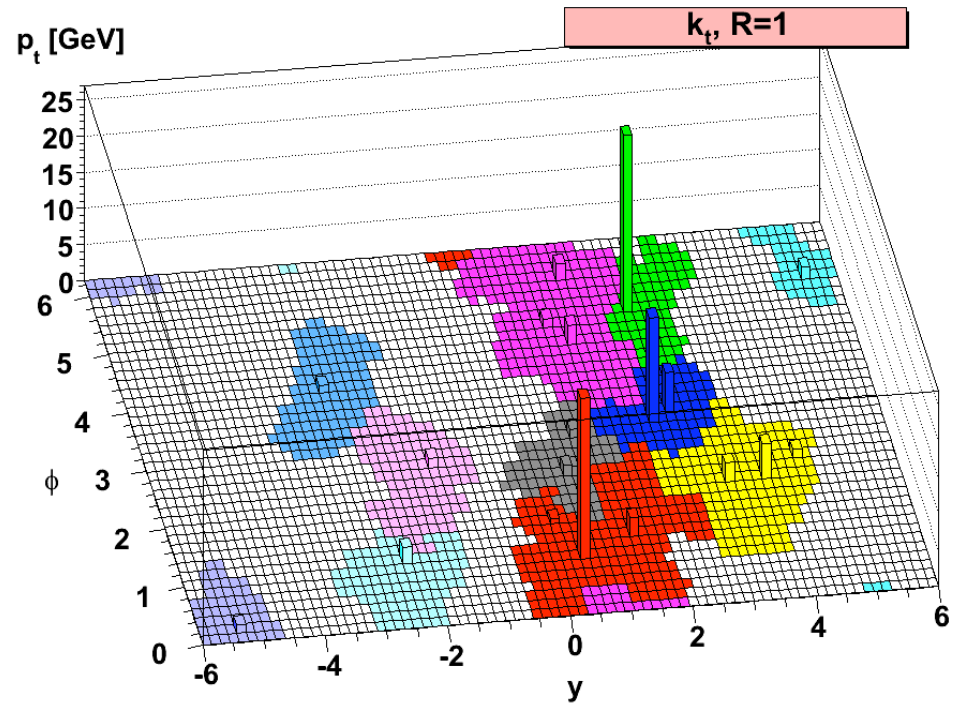
Algorithmically, a jet is simply a collection of particles

For a number of reasons, it is however useful to consider its **spatial extent**, i.e. given the position of its axis, up to where does it collect particles? What is its shape?

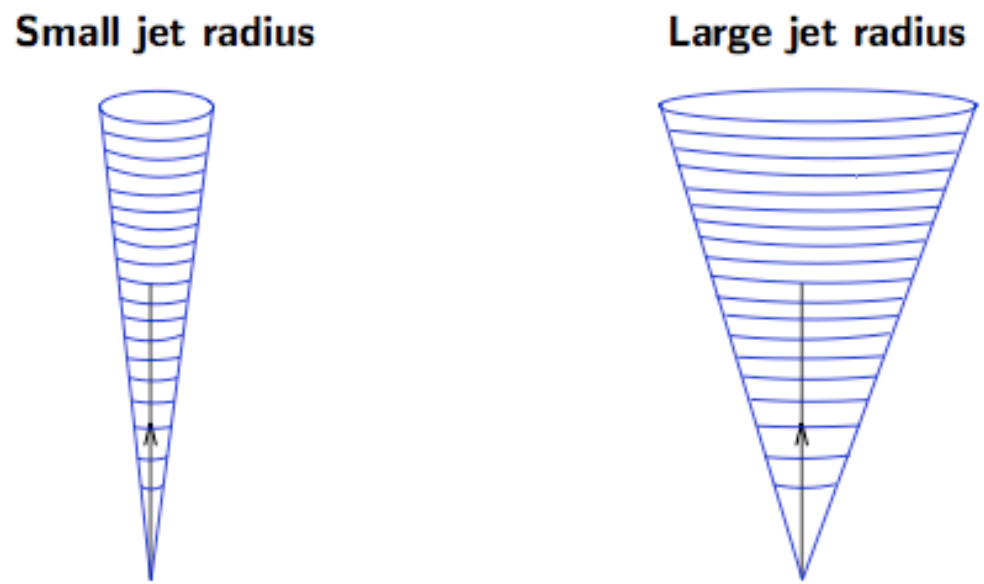
These details are important for a number of corrections of various origin: perturbative, non-perturbative (hadronisation), detector related, etc

Note that the intuitive picture of a jet being a cone (of radius R) is **wrong.** This is what k_t jets can look like:

(more later about what this plot really means)

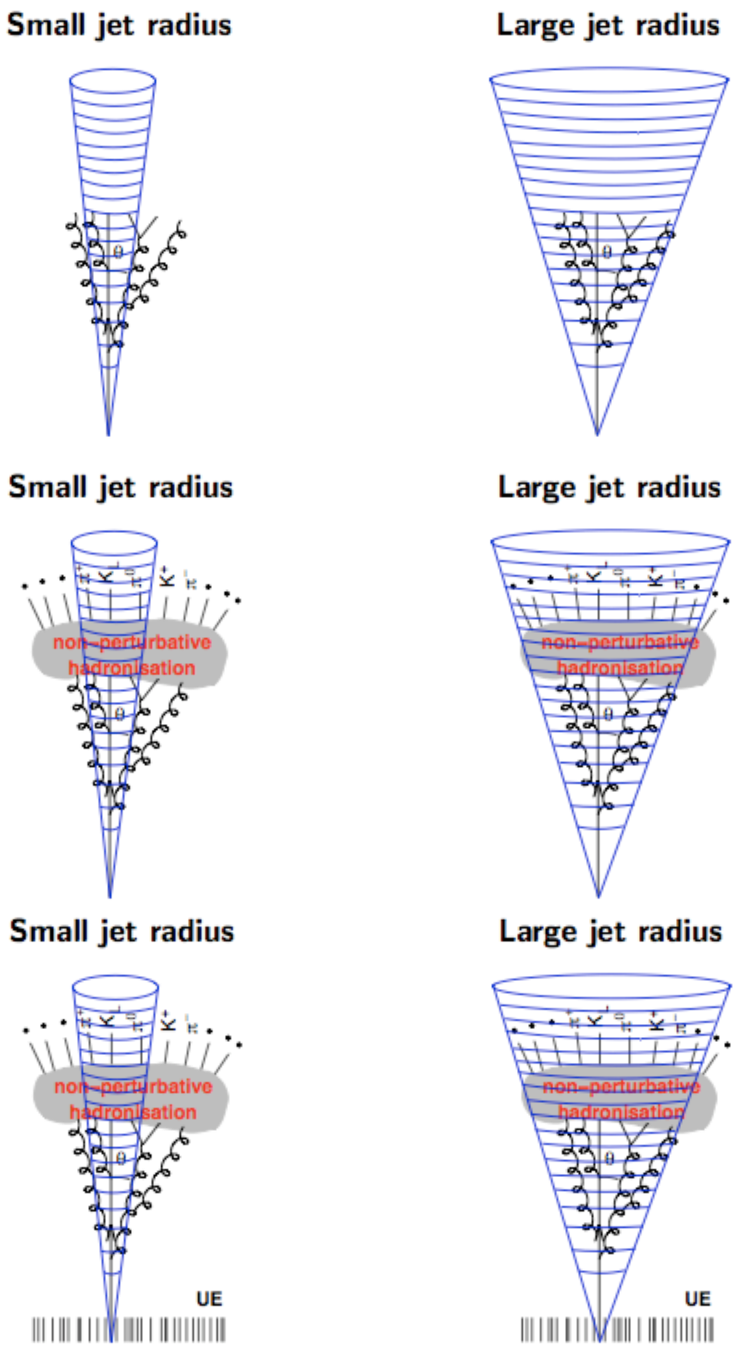


Effects of jet ‘radius’



Irrelevant for a single-particle jet

Effects of jet ‘radius’



perturbative radiation:
large radius better (lose less)

non-perturbative hadronisation:
large radius better (lose less)

underlying event:
large radius worse (capture more)

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: $$\Delta p_t \simeq \frac{R^2}{2} \times (2.5 — 15 \text{ GeV})$$

Tevatron LHC

(small-R limit results)

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

From jet 'reach' to jet areas

Not one, but three **definitions** of a jet's size:

MC, Salam, Soyez, arXiv:0802.1188

- **Passive** area

  Place a single soft particle in the event, measure the extent of the region where it gets clustered within a given jet

  Reach of jet for **pointlike** radiation

- **Active** area

  Fill the events with many soft particles, cluster them together with the hard ones, see how many get clustered within a given jet

  Reach of jet for **diffuse** radiation

- **Voronoi** area

  Sum of areas of intersections of Voronoi cells of jet constituents with circle of radius R centred on each constituent

  Coincides with passive area for k_t algorithm

(In the large number of particles limit all areas converge to the same value)

Jet active area

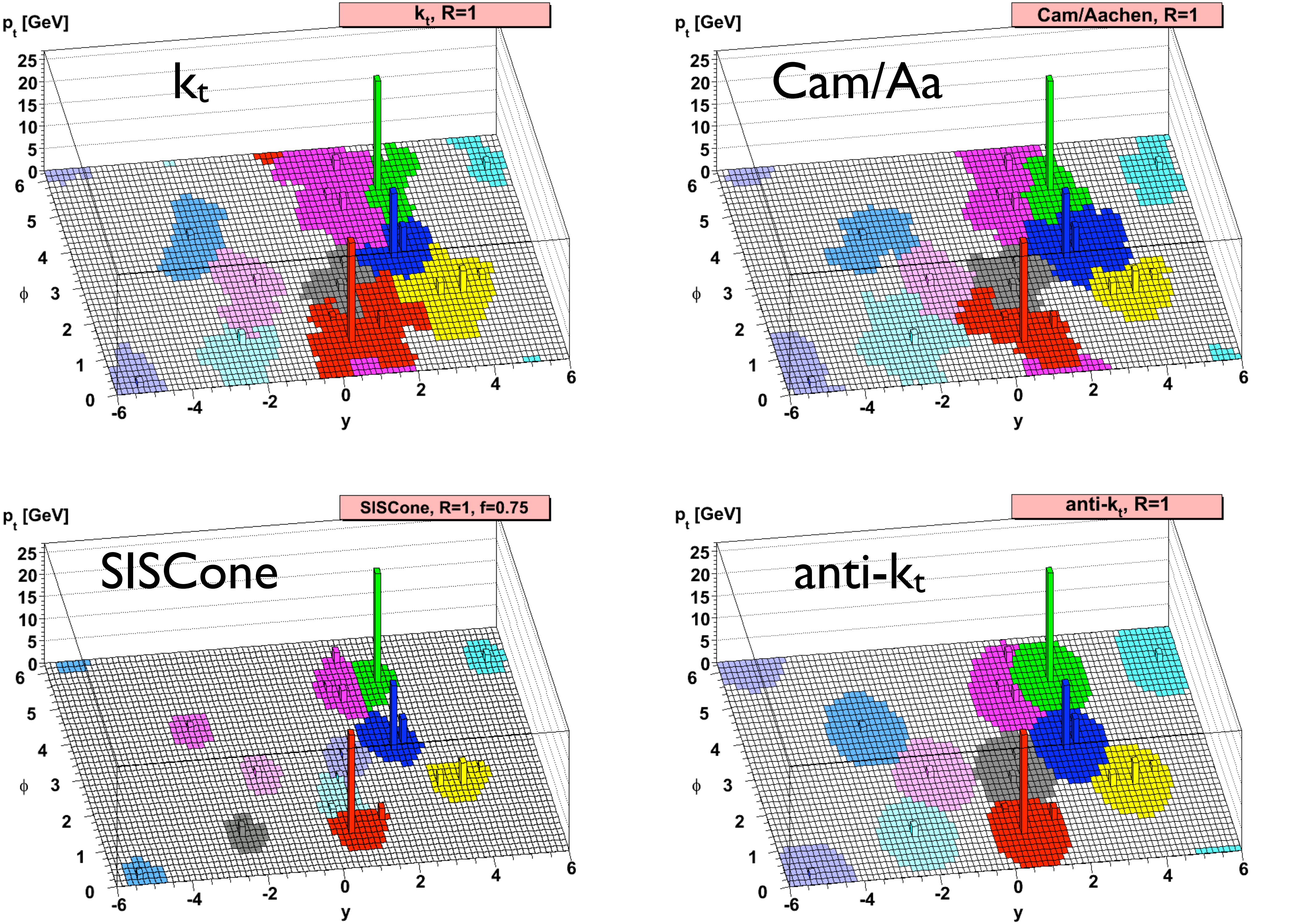
The definition of **active area** mimics the behaviour of the jet-clustering algorithms in the presence of a **large number of randomly distributed soft particles,** like those due to **pileup or underlying event**

Tools needed to implement it

1. An **infrared safe jet algorithm** (the ghosts should not change the jets)
2. A reasonably **fast implementation** (we are adding thousands of ghosts)

Both are available

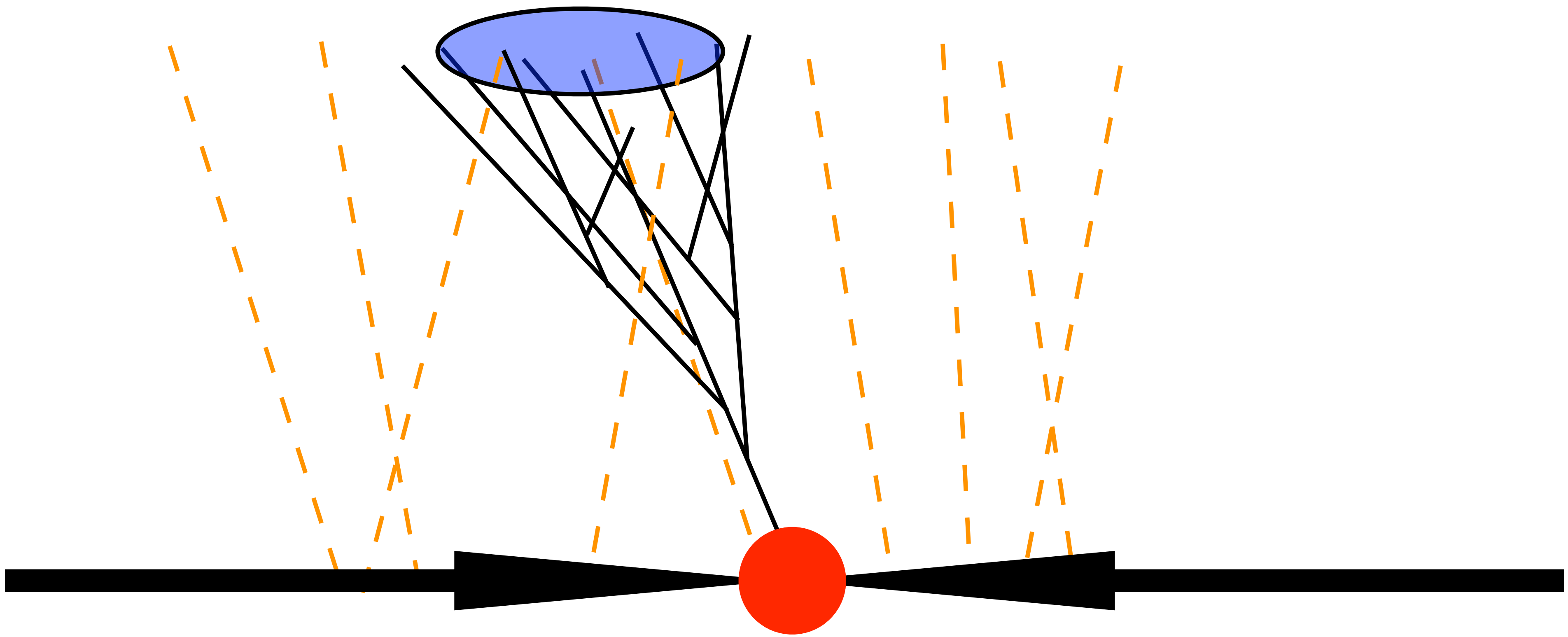
As a bonus, active areas also allow for a **visualisation** of a jet's reach


k$_t$, R=1
k$_t$
Cam/Aachen, R=1
Cam/Aa
SISCone, R=1, f=0.75
SISCone
anti-k$_t$, R=1
anti-k$_t$
p$_t$ [GeV]
ϕ
y

Jet area: summary

- Jets CAN have an area, but one must define it
- The jet (active) area expresses the susceptibility of a jet to contamination from a uniform background
- Different jet algorithms can have very different area properties:
  - Jet areas in many algorithms can fluctuate significantly from a jet to another. Isolated hard jets in anti-k_t are one exception
  - Jet areas can depend on a jet's p_t, driven by a (calculable) anomalous dimension that is specific to each jet algorithm. Anti-k_t jets are again an exception, in that the anomalous dimension is zero.

Hard jets and background

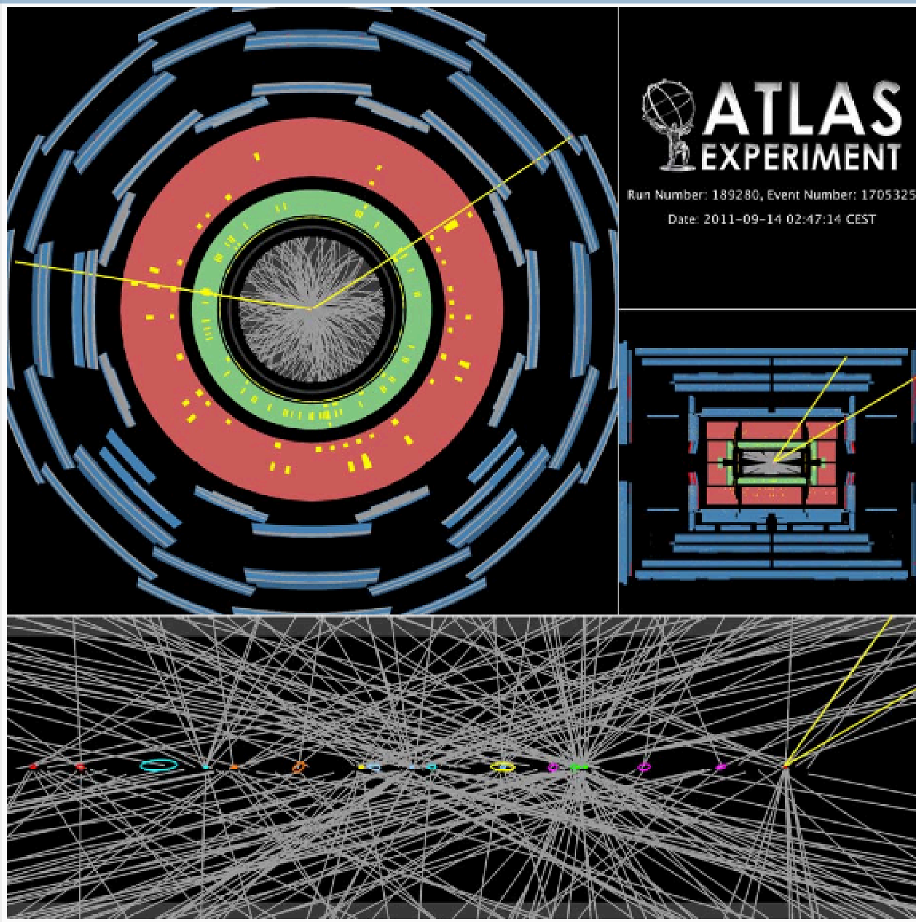


In a realistic set-up underlying event (UE) and pile-up (PU) from multiple collisions produce many soft particles which can 'contaminate' the hard jet

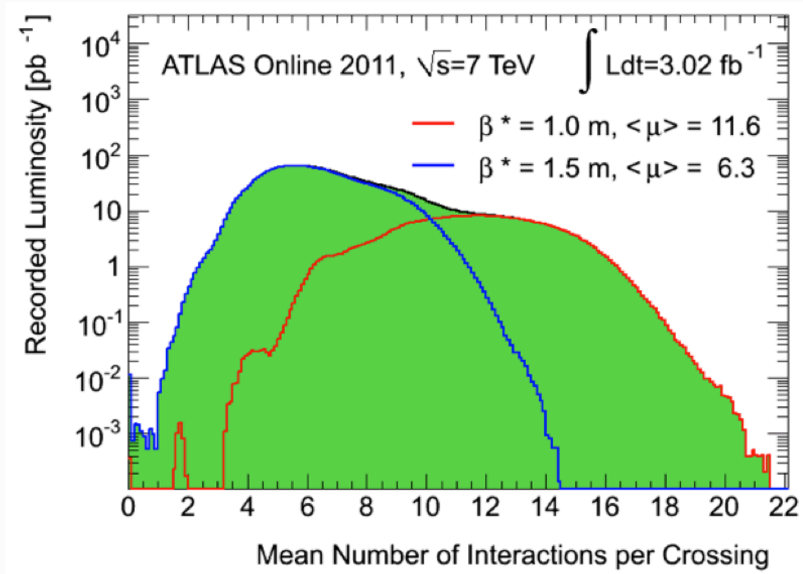
Slide from G. Dissertori

Pileup at LHC

Consequence of these beam parameters



$Z \rightarrow \mu\mu$ with $N_{vtx} = 20$



Very large Pile-Up: impact on trigger rates, computing/reconstruction time, reconstruction efficiencies (eg. isolation), jet energy reconstruction, ...

How are the hard jets modified by the background?

Susceptibility
(how much bkgd gets picked up)

Jet areas

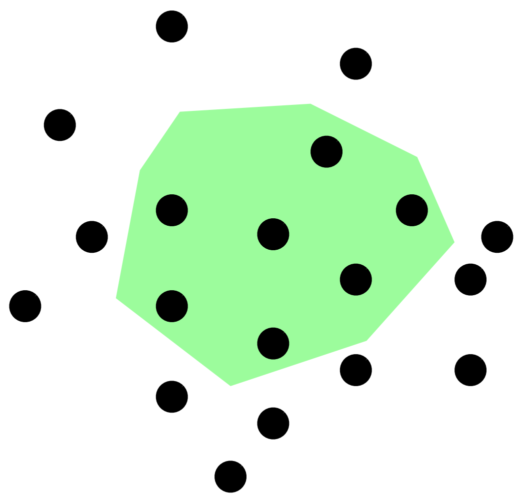
Resiliency
(how much the original jet changes)

Backreaction

Resiliency: backreaction

"How (much) a jet changes when immersed in a background"

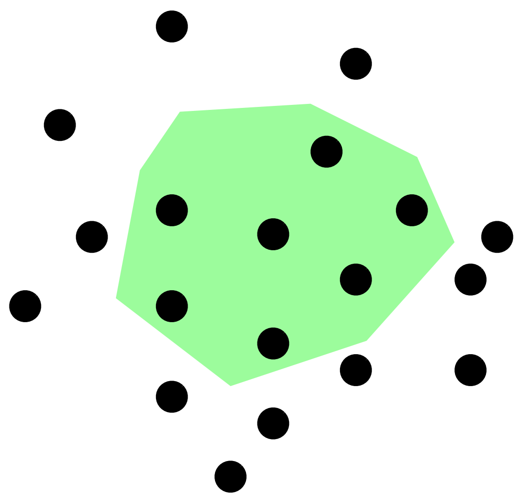
Without background

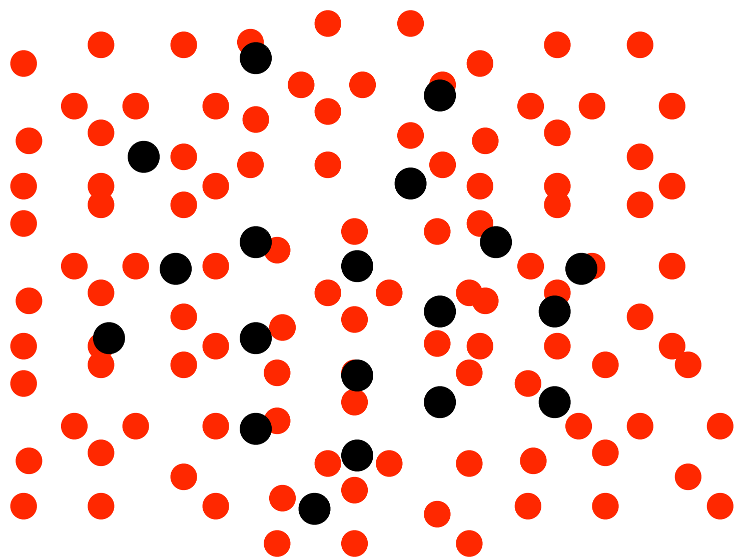


Resiliency: backreaction

"How (much) a jet changes when immersed in a background"

Without background

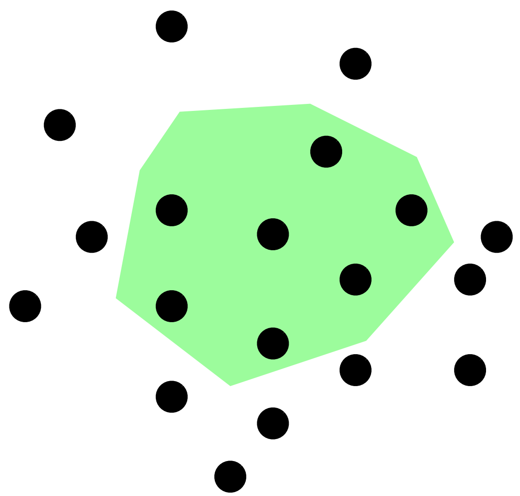




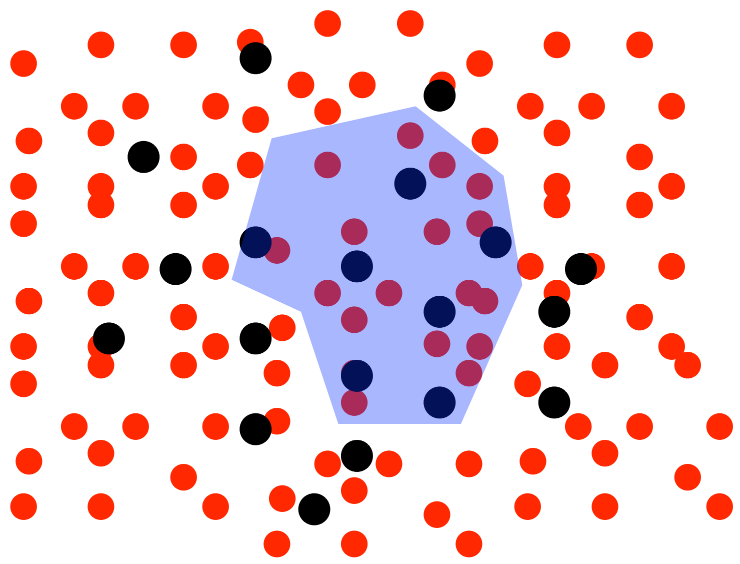
Resiliency: backreaction

"How (much) a jet changes when immersed in a background"

Without background



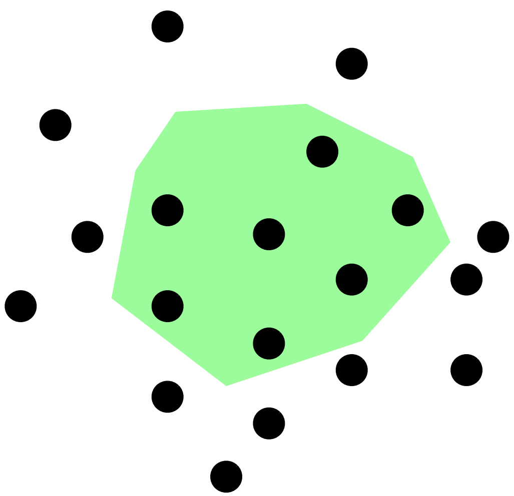
With background



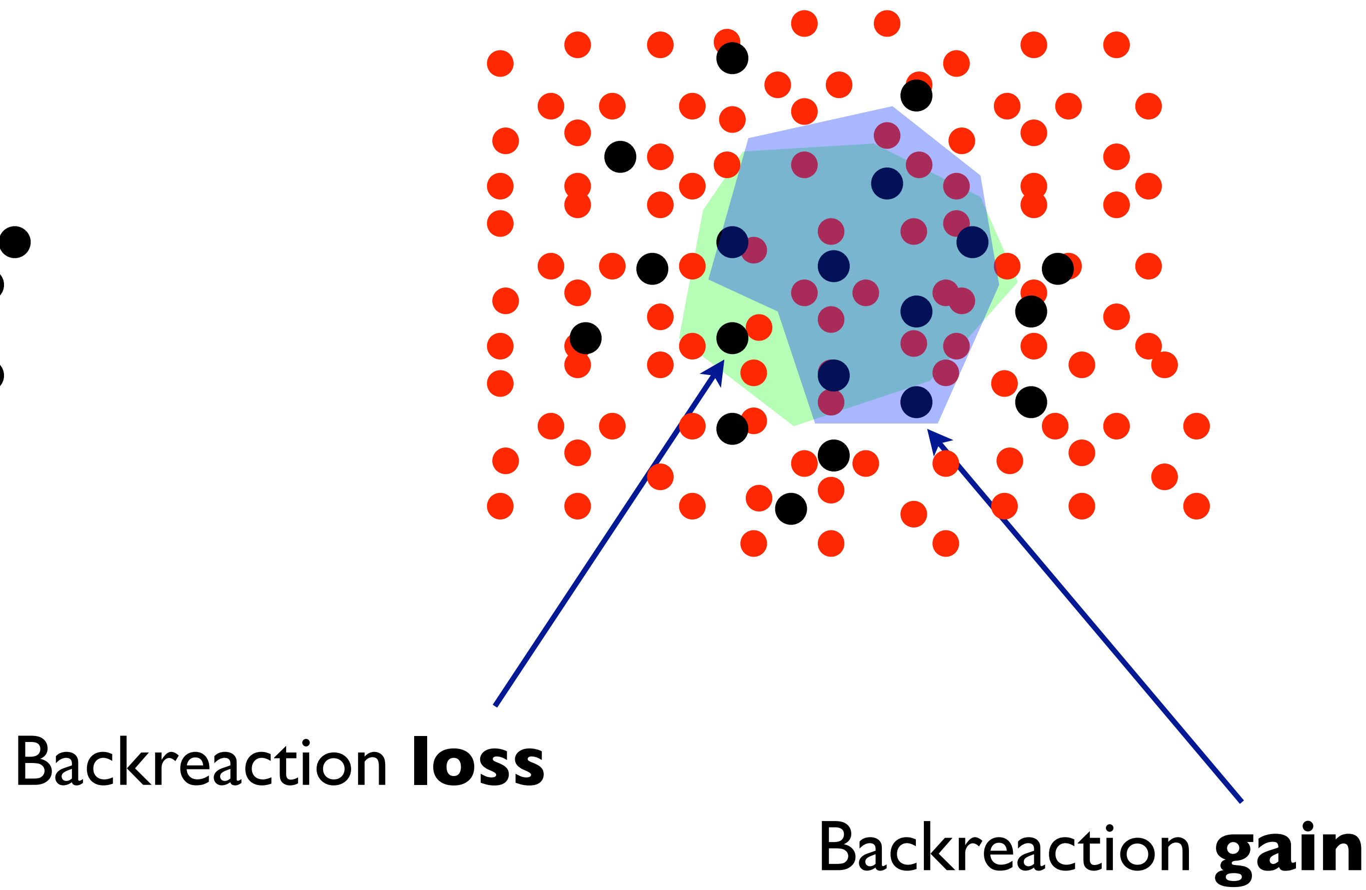
Resiliency: backreaction

“How (much) a jet changes when immersed in a background”

Without background

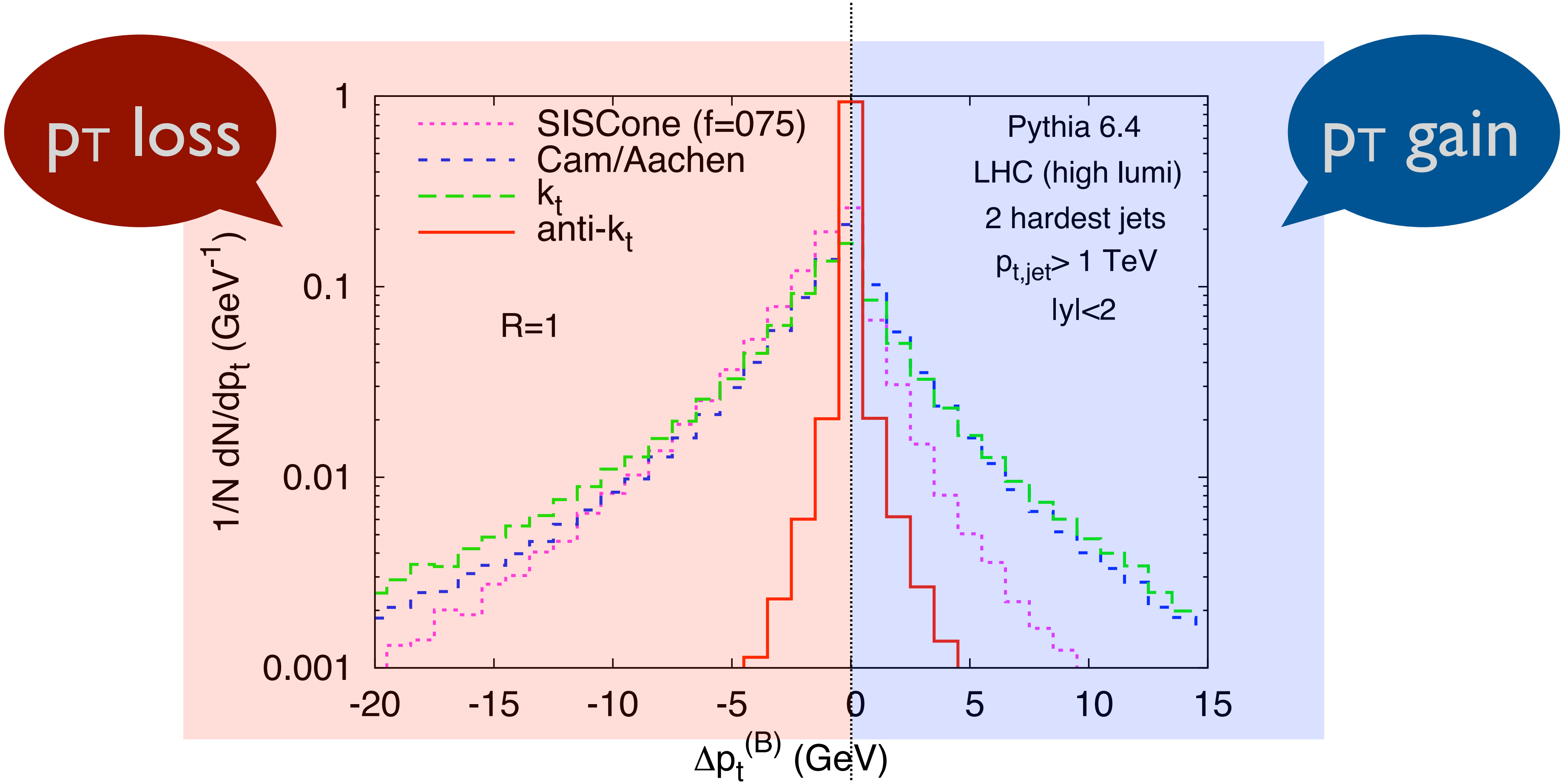


With background



Resiliency: backreaction

MC, Salam, Soyez, arXiv:0802.1188



Anti-k$_t$ jets are much more resilient to changes from background immersion

The IRC safe algorithms

	Speed	Regularity	UE contamination	Backreaction	Hierarchical substructure
k_t	☺ ☺ ☺	☂	☂☂	☁☁	☺ ☺
Cambridge /Aachen	☺ ☺ ☺	☂	☂	☁☁	☺ ☺ ☺
anti-k_t	☺ ☺ ☺	☺ ☺	☁/ ☺	☺ ☺	✘
SISCone	☺	☁	☺ ☺	☁	✘

Hard jets and background

MC, Salam, arXiv:0707.1378
MC, Salam, Soyez, arXiv:0802.1188

Modifications of the hard jet

$$\Delta p_t = \boxed{\rho A \pm (\sigma\sqrt{A} + \sigma_\rho A + \rho\sqrt{\langle A^2\rangle - \langle A\rangle^2})} \boxed{+ \Delta p_t^{BR}}$$

Background transverse momentum density (per unit area)

background

'susceptibility'

back-reaction

'resiliency'

Background determination

Jet algorithms like k_t or Cambridge/Aachen allow one to determine
on an event-by-event basis
the **“typical” level of transverse momentum density**
of a **roughly uniform background noise**:

$$\rho \equiv \underset{\text{(over a \underline{single} event)}}{\text{median}} \left[\left\{ \frac{p_t^{jet}}{\text{Area}_{jet}} \right\} \right]$$

MC, Salam, 2007

This ρ value can, in turn, be used to characterise the UE

Since this measurement is done with the jets, it is alternative/complementary to the usual analyses done using charged tracks (à la R. Field)

Background subtraction

Once ρ has been measured, it can be used to correct the transverse momentum of the hard jets:

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

ρ being measured on an event-by-event basis, and each jet subtracted individually, this procedure will remove many fluctuations and generally improve the resolution of, say, a mass peak

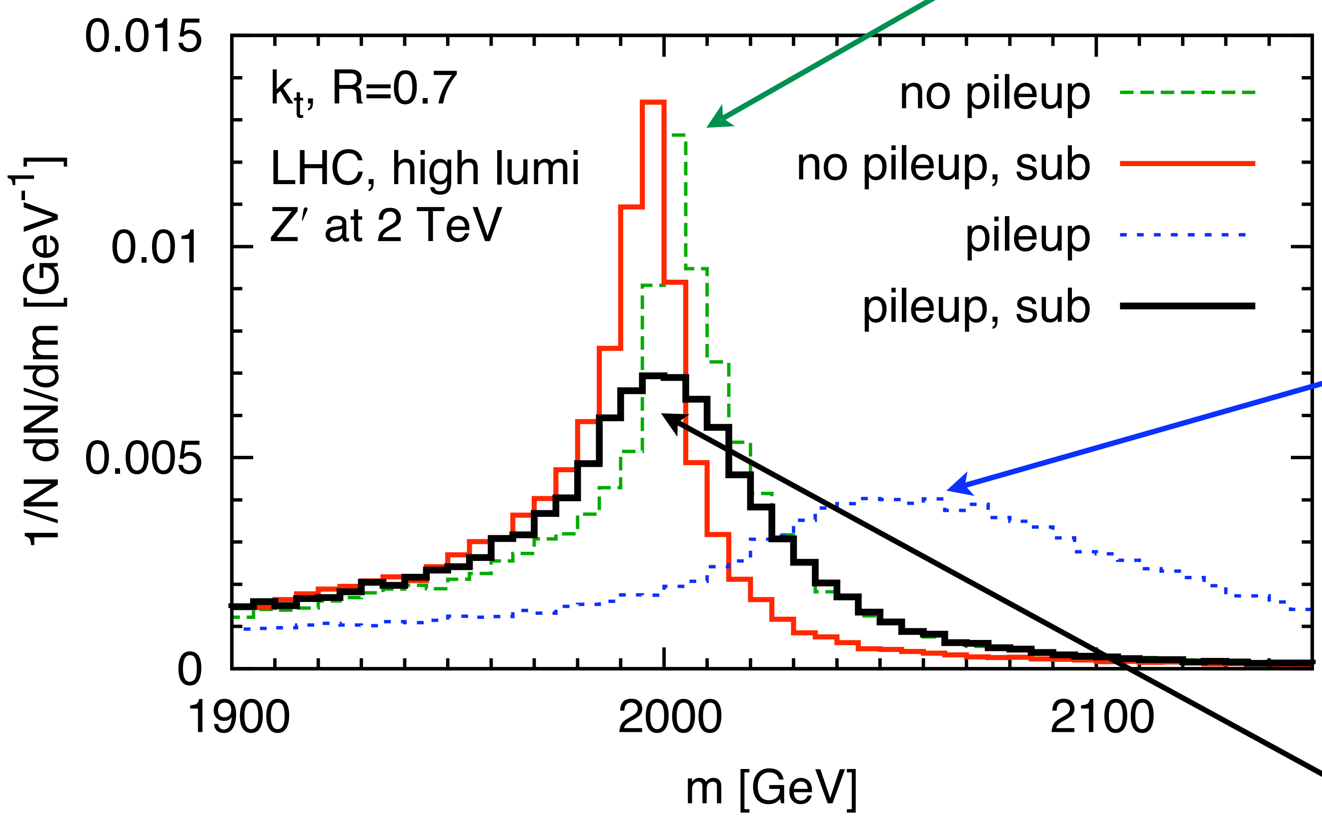
$$\Delta p_t = \rho A \pm (\sigma\sqrt{A} + \cancel{\sigma_\rho A} + \rho\sqrt{\cancel{\langle A^2\rangle - \langle A\rangle^2}}) + \Delta p_t^{BR}$$

NB. Also be(a)ware of ***backreaction***

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