Jets and QCD — lecture 2

Matteo Cacciari[1,2] and Gavin Salam[3,4,1]

[1] LPTHE (CNRS/UPMC)
[2] Université Denis Diderot (Paris 7)
[3] CERN, PH-TH
[4] Princeton University

Focus Week at the GGI Workshop
High energy QCD after the start of the LHC
Florence, Italy, 12-16 September 2011

Soft stuff clusters with nearest neighbour

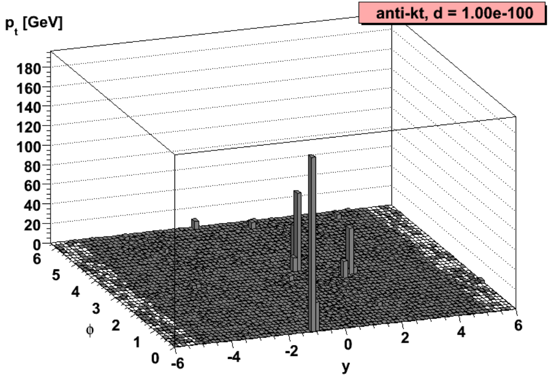
$$k_t\text{: } d_{ij} = \min(k_{ti}^2, k_{tj}^2)\Delta R_{ij}^2 \longrightarrow \textbf{anti-k}_\mathbf{t}\text{: } d_{ij} = \frac{\Delta R_{ij}^2}{\max(k_{ti}^2, k_{tj}^2)}$$

Hard stuff clusters with nearest neighbour

Soft stuff clusters with nearest neighbour

$$k_t:\ d_{ij} = \min(k_{ti}^2, k_{tj}^2)\Delta R_{ij}^2 \longrightarrow \textbf{anti-k}_\mathbf{t}:\ d_{ij} = \frac{\Delta R_{ij}^2}{\max(k_{ti}^2, k_{tj}^2)}$$

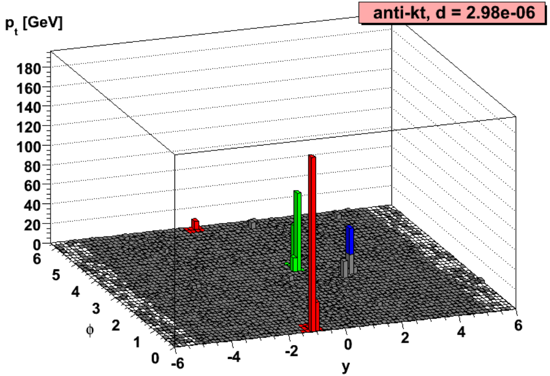
Hard stuff clusters with nearest neighbour



Soft stuff clusters with nearest neighbour

$$k_t:\ d_{ij} = \min(k_{ti}^2, k_{tj}^2)\Delta R_{ij}^2 \longrightarrow \textbf{anti-k}_\mathbf{t}:\ d_{ij} = \frac{\Delta R_{ij}^2}{\max(k_{ti}^2, k_{tj}^2)}$$

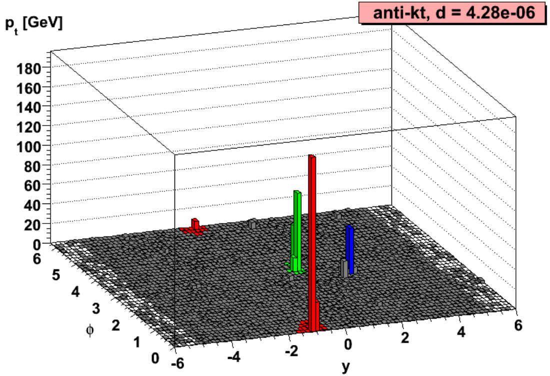
Hard stuff clusters with nearest neighbour



Soft stuff clusters with nearest neighbour

$$k_t:\ d_{ij} = \min(k_{ti}^2, k_{tj}^2)\Delta R_{ij}^2 \longrightarrow \textbf{anti-k}_\mathbf{t}:\ d_{ij} = \frac{\Delta R_{ij}^2}{\max(k_{ti}^2, k_{tj}^2)}$$

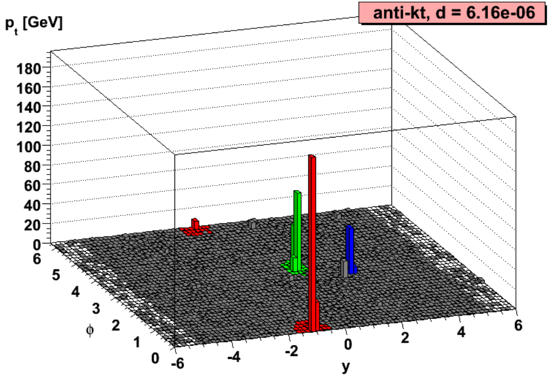
Hard stuff clusters with nearest neighbour



Soft stuff clusters with nearest neighbour

$$k_t\colon\ d_{ij} = \min(k_{ti}^2, k_{tj}^2)\Delta R_{ij}^2 \longrightarrow \textbf{anti-k}_\mathbf{t}\colon\ d_{ij} = \frac{\Delta R_{ij}^2}{\max(k_{ti}^2, k_{tj}^2)}$$

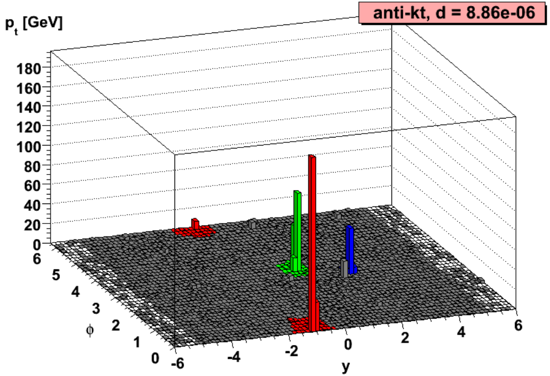
Hard stuff clusters with nearest neighbour



Soft stuff clusters with nearest neighbour

$$k_t:\ d_{ij} = \min(k_{ti}^2, k_{tj}^2)\Delta R_{ij}^2 \longrightarrow \textbf{anti-k}_\mathbf{t}:\ d_{ij} = \frac{\Delta R_{ij}^2}{\max(k_{ti}^2, k_{tj}^2)}$$

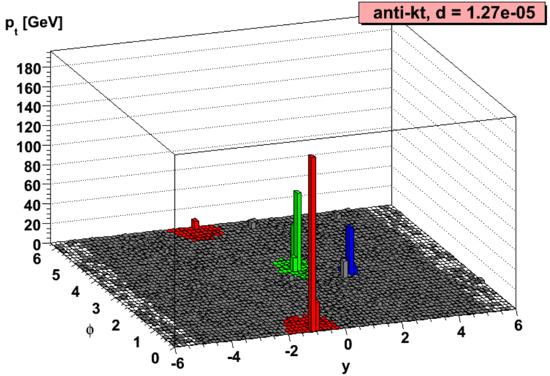
Hard stuff clusters with nearest neighbour



Soft stuff clusters with nearest neighbour

$$k_t\colon\ d_{ij} = \min(k_{ti}^2, k_{tj}^2)\Delta R_{ij}^2 \longrightarrow \textbf{anti-k}_\mathbf{t}\colon\ d_{ij} = \frac{\Delta R_{ij}^2}{\max(k_{ti}^2, k_{tj}^2)}$$

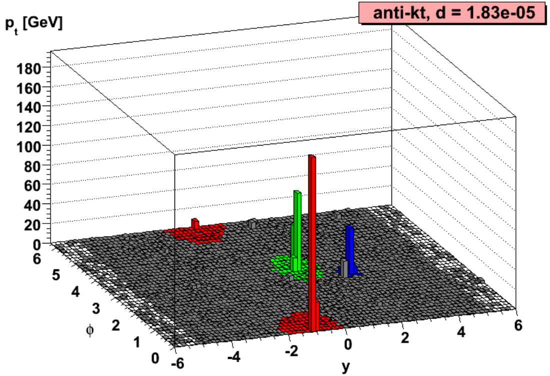
Hard stuff clusters with nearest neighbour



Soft stuff clusters with nearest neighbour

$$k_t\colon\ d_{ij} = \min(k_{ti}^2, k_{tj}^2)\Delta R_{ij}^2 \longrightarrow \textbf{anti-k}_\mathbf{t}\colon\ d_{ij} = \frac{\Delta R_{ij}^2}{\max(k_{ti}^2, k_{tj}^2)}$$

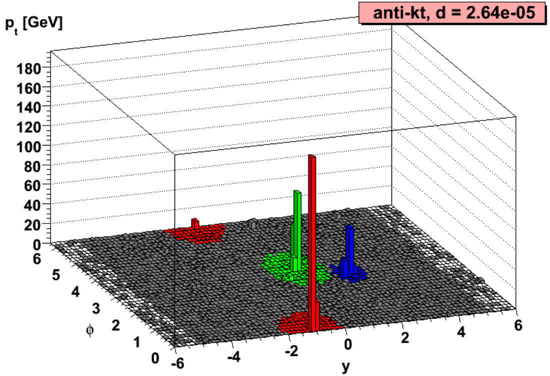
Hard stuff clusters with nearest neighbour



Soft stuff clusters with nearest neighbour

$$k_t\colon\ d_{ij} = \min(k_{ti}^2, k_{tj}^2)\Delta R_{ij}^2 \longrightarrow \textbf{anti-k}_\mathbf{t}\colon\ d_{ij} = \frac{\Delta R_{ij}^2}{\max(k_{ti}^2, k_{tj}^2)}$$

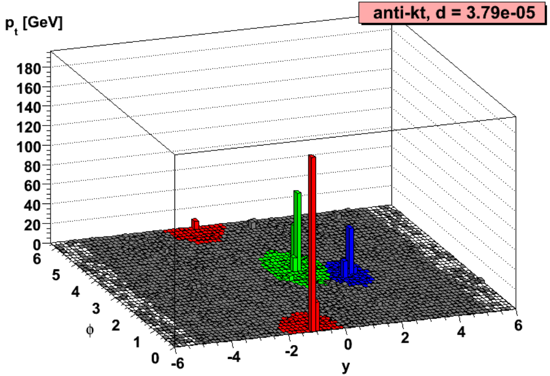
Hard stuff clusters with nearest neighbour



Soft stuff clusters with nearest neighbour

$$k_t\text{: } d_{ij} = \min(k_{ti}^2, k_{tj}^2)\Delta R_{ij}^2 \longrightarrow \textbf{anti-k}_\mathbf{t}\text{: } d_{ij} = \frac{\Delta R_{ij}^2}{\max(k_{ti}^2, k_{tj}^2)}$$

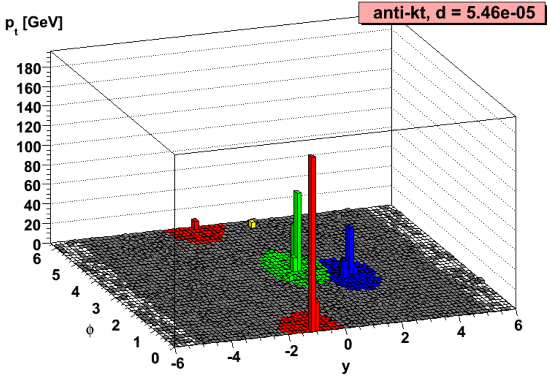
Hard stuff clusters with nearest neighbour



Soft stuff clusters with nearest neighbour

$$k_t\colon\ d_{ij} = \min(k_{ti}^2, k_{tj}^2)\Delta R_{ij}^2 \longrightarrow \textbf{anti-k}_\mathbf{t}\colon\ d_{ij} = \frac{\Delta R_{ij}^2}{\max(k_{ti}^2, k_{tj}^2)}$$

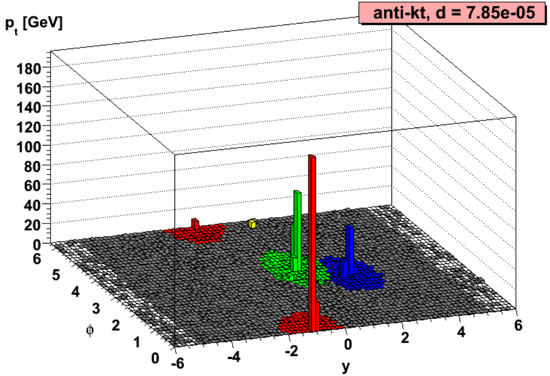
Hard stuff clusters with nearest neighbour



Soft stuff clusters with nearest neighbour

$$k_t\colon\ d_{ij} = \min(k_{ti}^2, k_{tj}^2)\Delta R_{ij}^2 \longrightarrow \textbf{anti-k}_\mathbf{t}\colon\ d_{ij} = \frac{\Delta R_{ij}^2}{\max(k_{ti}^2, k_{tj}^2)}$$

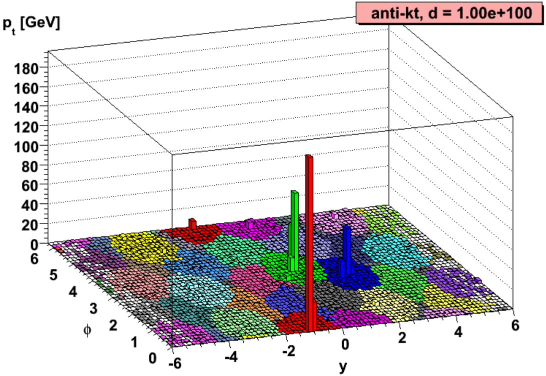
Hard stuff clusters with nearest neighbour



Soft stuff clusters with nearest neighbour

$$k_t\colon\ d_{ij} = \min(k_{ti}^2, k_{tj}^2)\Delta R_{ij}^2 \longrightarrow \textbf{anti-k}_\textbf{t}\colon\ d_{ij} = \frac{\Delta R_{ij}^2}{\max(k_{ti}^2, k_{tj}^2)}$$

Hard stuff clusters with nearest neighbour



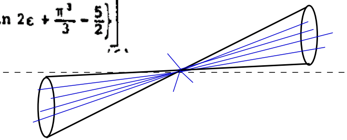
anti-k_t gives cone-like jets without using stable cones

Cone Origins

First ‘jet algorithm’ dates back to Sterman and Weinberg (1977) — the original infrared-safe cross section:

To study jets, we consider the partial cross section $\sigma(E,\theta,\Omega,\varepsilon,\delta)$ for e^+e^- hadron production events, in which all but a fraction $\varepsilon \ll 1$ 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-

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



Groundbreaking; good for 2 jets in e^+e^-; but never widely generalised

Cone algorithms today

Unifying idea: momentum flow within a cone only marginally modified by QCD branching

But cones come in many variants

Finding cones \ Processing	Progressive Removal	Split–Merge	Split–Drop
Seeded, Fixed (FC)	GetJet CellJet		
Seeded, Iterative (IC)	CMS Cone	JetClu(CDF)[†] ATLAS cone	
Seeded, It. + Midpoints (IC_{mp})		CDF MidPoint D0 Run II cone	PxCone
Seedless (SC)		SISCone	

[†] JetClu also has "ratcheting"

Cone algorithms today

Unifying idea: momentum flow within a cone only marginally modified by QCD branching

But cones come in many variants

Finding cones \ Processing	Progressive Removal	Split–Merge	Split–Drop
Seeded, Fixed (FC)	GetJet CellJet		
Seeded, Iterative (IC)	CMS Cone	JetClu(CDF)† ATLAS cone	
Seeded, It. + Midpoints (IC_{mp})		CDF MidPoint D0 Run II cone	PxCone
Seedless (SC)		SISCone	

† JetClu also has "ratcheting"

Cone algorithms today

Unifying idea: momentum flow within a cone only marginally modified by QCD branching

But cones come in many variants

Finding cones \ Processing	Progressive Removal	Split–Merge	Split–Drop
Seeded, Fixed (FC)	GetJet CellJet		
Seeded, Iterative (IC)	CMS Cone	JetClu(CDF)[†] ATLAS cone	
Seeded, It. + Midpoints (IC_{mp})		CDF MidPoint D0 Run II cone	PxCone
Seedless (SC)		SISCone	

[†] JetClu also has "ratcheting"

Common features in discussion of cones

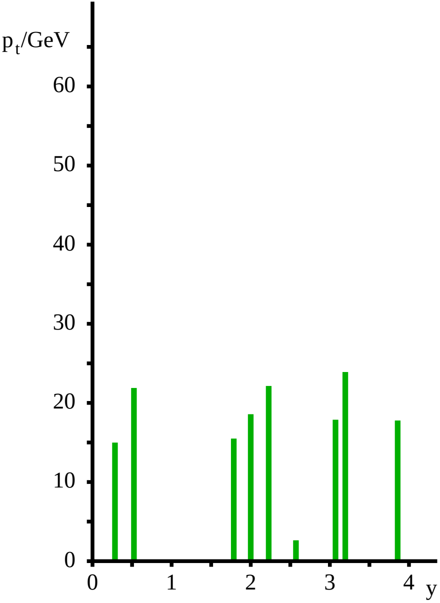
- Cones are always understood as circles in rapidity (y) and azimuth ϕ.
- A particle i is within the cone of radius R around the axis a if

$$\Delta R_{ia}^2 = (y_i - y_a)^2 + (\phi_i - \phi_a)^2 < R^2$$

The usual hadron collider variables

- We'll use $R = 0.7$ in the examples that follow
- And we'll use events all of whose particles are at $\phi = 0$, for simplicity

It. Cone with Split–Merge (IC-SM)



Avoid ordering seeds (coll. unsafe)

CDF JetClu[†] & ATLAS cones

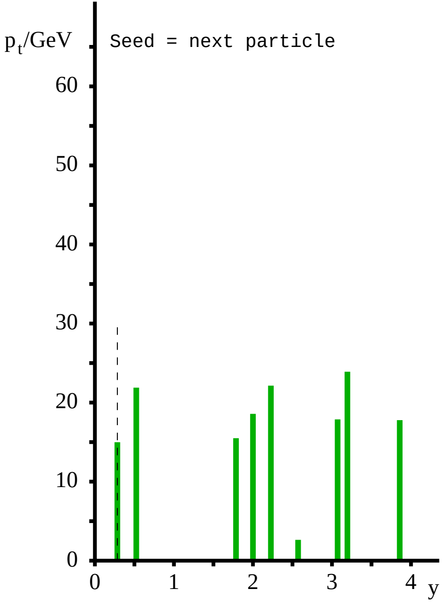
- use every particle as possible seed (no particular order)
- iterate until stable cone
- add the stable cone to the list of **protojets** unless it's already there
- until all seeds done

Note: protojets **overlap**. Certain particles appear in many protojets

protojet $\neq$ jet

Must resolve the overlaps.
Use a **split–merge** procedure.

It. Cone with Split–Merge (IC-SM)



Avoid ordering seeds (coll. unsafe)

CDF JetClu† & ATLAS cones

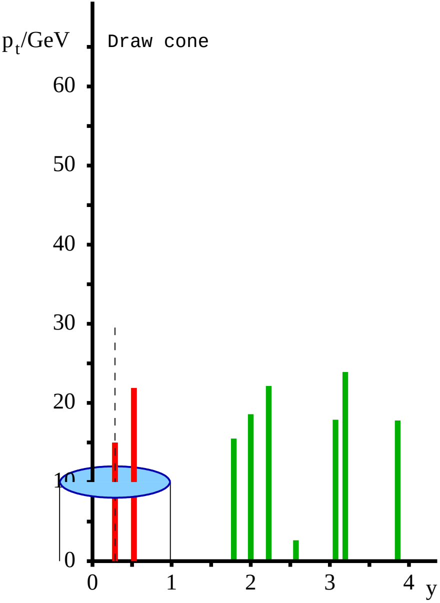
- use every particle as possible seed (no particular order)
- iterate until stable cone
- add the stable cone to the list of **protojets** unless it's already there
- until all seeds done

Note: protojets overlap. Certain particles appear in many protojets

protojet $\neq$ jet

Must resolve the overlaps.
Use a split–merge procedure.

It. Cone with Split–Merge (IC-SM)



Avoid ordering seeds (coll. unsafe)

CDF JetClu† & ATLAS cones

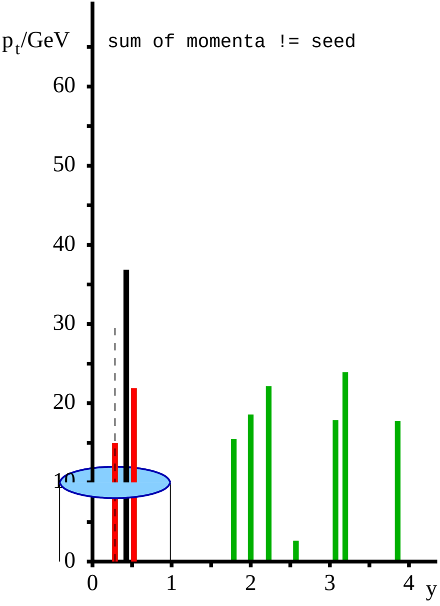
- use every particle as possible seed (no particular order)
- iterate until stable cone
- add the stable cone to the list of **protojets** unless it's already there
- until all seeds done

Note: protojets overlap. Certain particles appear in many protojets

protojet $\neq$ jet

Must resolve the overlaps.
Use a split–merge procedure.

It. Cone with Split–Merge (IC-SM)



Avoid ordering seeds (coll. unsafe)

CDF JetClu† & ATLAS cones

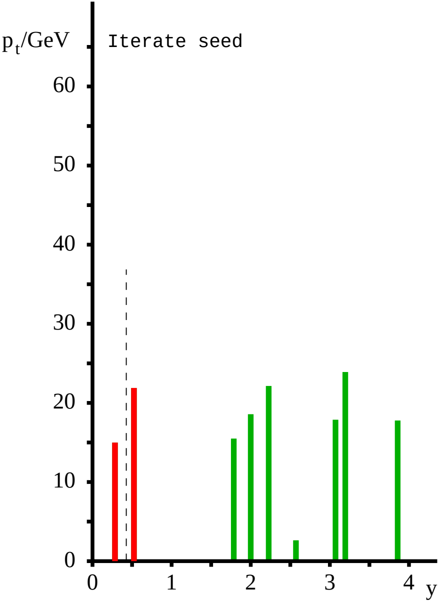
- use every particle as possible seed (no particular order)
- iterate until stable cone
- add the stable cone to the list of **protojets** unless it's already there
- until all seeds done

Note: protojets **overlap**. Certain particles appear in many protojets

protojet ≠ jet

Must resolve the overlaps.
Use a split–merge procedure.

It. Cone with Split–Merge (IC-SM)



Avoid ordering seeds (coll. unsafe)

CDF JetClu† & ATLAS cones

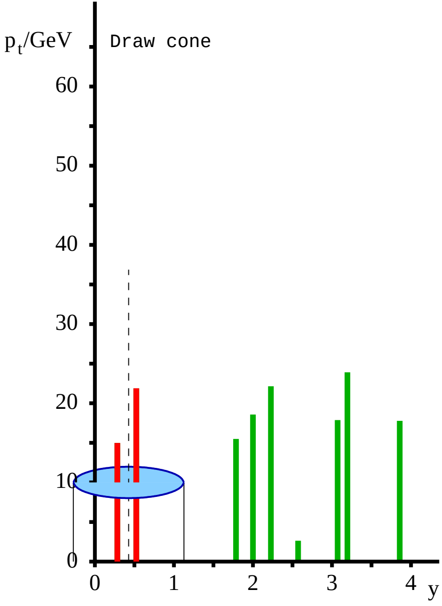
- use every particle as possible seed (no particular order)
- iterate until stable cone
- add the stable cone to the list of **protojets** unless it's already there
- until all seeds done

Note: protojets overlap. Certain particles appear in many protojets

protojet ≠ jet

Must resolve the overlaps.
Use a split–merge procedure.

It. Cone with Split–Merge (IC-SM)



Avoid ordering seeds (coll. unsafe)

CDF JetClu† & ATLAS cones

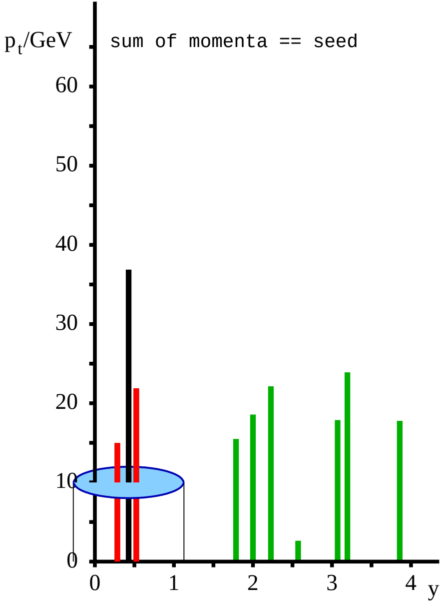
- use every particle as possible seed (no particular order)
- iterate until stable cone
- add the stable cone to the list of **protojets** unless it's already there
- until all seeds done

Note: protojets overlap. Certain particles appear in many protojets

protojet ≠ jet

Must resolve the overlaps.
Use a split–merge procedure.

It. Cone with Split–Merge (IC-SM)



Avoid ordering seeds (coll. unsafe)

CDF JetClu[†] & ATLAS cones

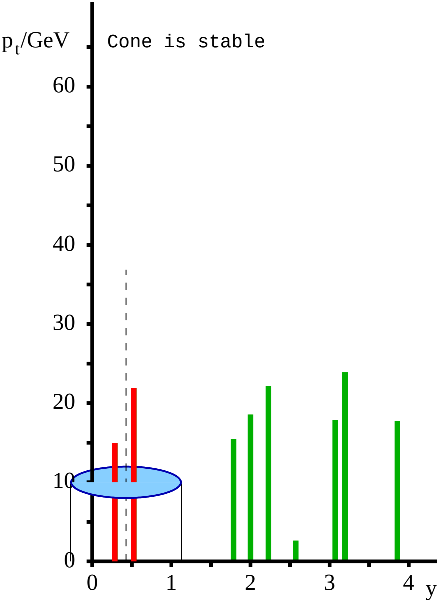
- use every particle as possible seed (no particular order)
- iterate until stable cone
- add the stable cone to the list of **protojets** unless it's already there
- until all seeds done

Note: protojets overlap. Certain particles appear in many protojets

protojet ≠ jet

Must resolve the overlaps.
Use a split–merge procedure.

It. Cone with Split–Merge (IC-SM)



Avoid ordering seeds (coll. unsafe)

CDF JetClu† & ATLAS cones

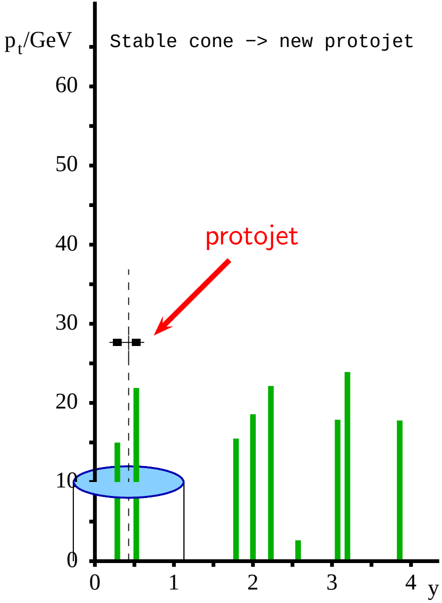
- use every particle as possible seed (no particular order)
- iterate until stable cone
- add the stable cone to the list of **protojets** unless it's already there
- until all seeds done

Note: protojets overlap. Certain particles appear in many protojets

protojet ≠ jet

Must resolve the overlaps.
Use a split–merge procedure.

It. Cone with Split–Merge (IC-SM)



Avoid ordering seeds (coll. unsafe)

CDF JetClu† & ATLAS cones

- use every particle as possible seed (no particular order)
- iterate until stable cone
- add the stable cone to the list of **protojets** unless it's already there
- until all seeds done

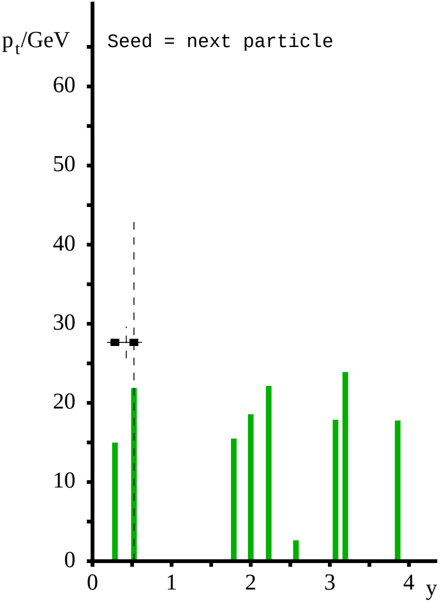
Note: protojets **overlap**. Certain particles appear in many protojets

protojet ≠ jet

Must resolve the overlaps.

Use a **split–merge** procedure.

It. Cone with Split–Merge (IC-SM)

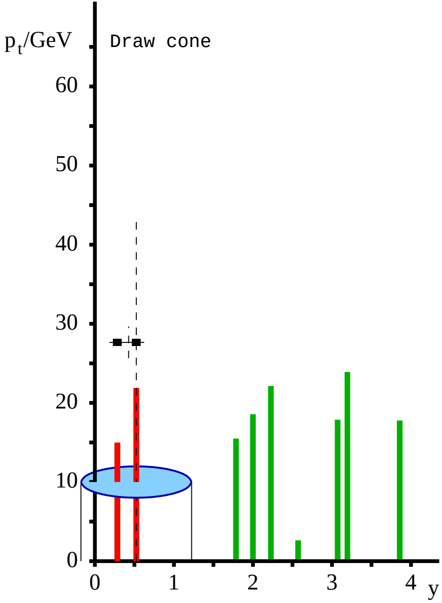


Avoid ordering seeds (coll. unsafe)

CDF JetClu† & ATLAS cones

- use every particle as possible seed (no particular order)
- iterate until stable cone
- add the stable cone to the list of **protojets** unless it's already there
- until all seeds done

It. Cone with Split–Merge (IC-SM)



Avoid ordering seeds (coll. unsafe)

CDF JetClu† & ATLAS cones

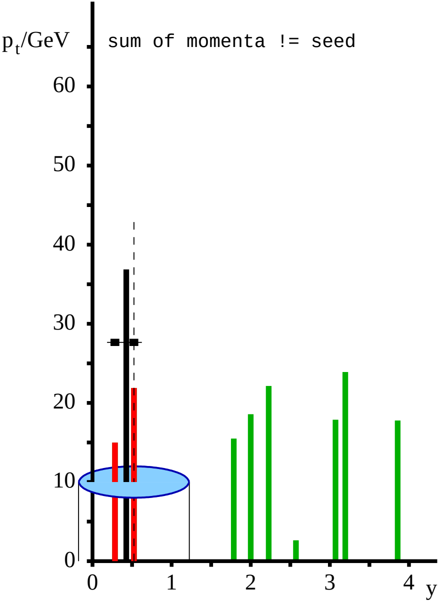
- use every particle as possible seed (no particular order)
- iterate until stable cone
- add the stable cone to the list of **protojets** unless it's already there
- until all seeds done

Note: protojets overlap. Certain particles appear in many protojets

protojet ≠ jet

Must resolve the overlaps.
Use a split–merge procedure.

It. Cone with Split–Merge (IC-SM)



Avoid ordering seeds (coll. unsafe)

CDF JetClu† & ATLAS cones

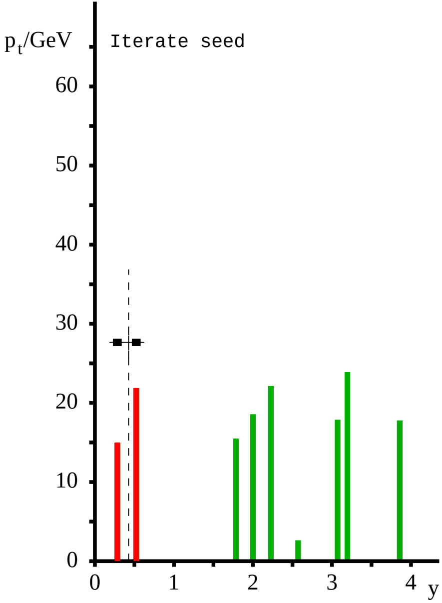
- use every particle as possible seed (no particular order)
- iterate until stable cone
- add the stable cone to the list of **protojets** unless it's already there
- until all seeds done

Note: protojets **overlap**. Certain particles appear in many protojets

protojet ≠ jet

Must resolve the overlaps.
Use a split–merge procedure.

It. Cone with Split–Merge (IC-SM)



Avoid ordering seeds (coll. unsafe)

CDF JetClu† & ATLAS cones

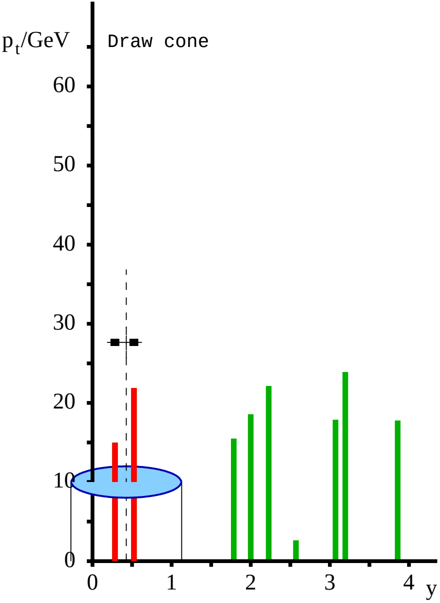
- use every particle as possible seed (no particular order)
- iterate until stable cone
- add the stable cone to the list of **protojets** unless it's already there
- until all seeds done

Note: protojets overlap. Certain particles appear in many protojets

protojet ≠ jet

Must resolve the overlaps.
Use a split–merge procedure.

It. Cone with Split–Merge (IC-SM)

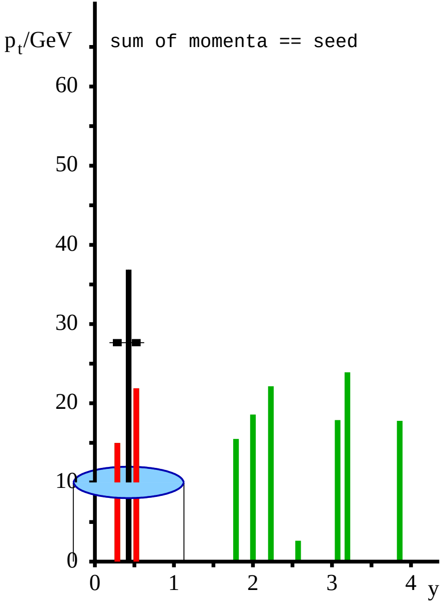


Avoid ordering seeds (coll. unsafe)

CDF JetClu† & ATLAS cones

- use every particle as possible seed (no particular order)
- iterate until stable cone
- add the stable cone to the list of **protojets** unless it's already there
- until all seeds done

It. Cone with Split–Merge (IC-SM)



Avoid ordering seeds (coll. unsafe)

CDF JetClu† & ATLAS cones

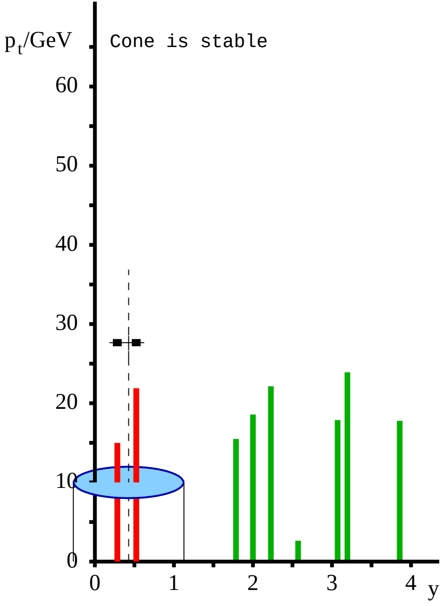
- use every particle as possible seed (no particular order)
- iterate until stable cone
- add the stable cone to the list of **protojets** unless it's already there
- until all seeds done

Note: protojets overlap. Certain particles appear in many protojets

protojet $\neq$ jet

Must resolve the overlaps.
Use a split–merge procedure.

It. Cone with Split–Merge (IC-SM)



Avoid ordering seeds (coll. unsafe)

CDF JetClu† & ATLAS cones

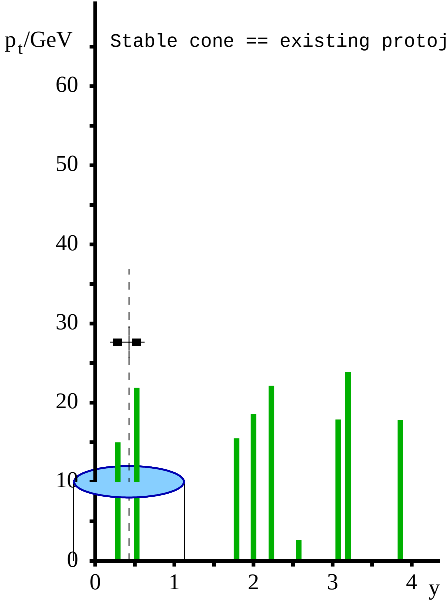
- use every particle as possible seed (no particular order)
- iterate until stable cone
- add the stable cone to the list of **protojets** unless it's already there
- until all seeds done

Note: protojets **overlap**. Certain particles appear in many protojets

protojet ≠ jet

Must resolve the overlaps.
Use a **split–merge** procedure.

It. Cone with Split–Merge (IC-SM)



Avoid ordering seeds (coll. unsafe)

CDF JetClu† & ATLAS cones

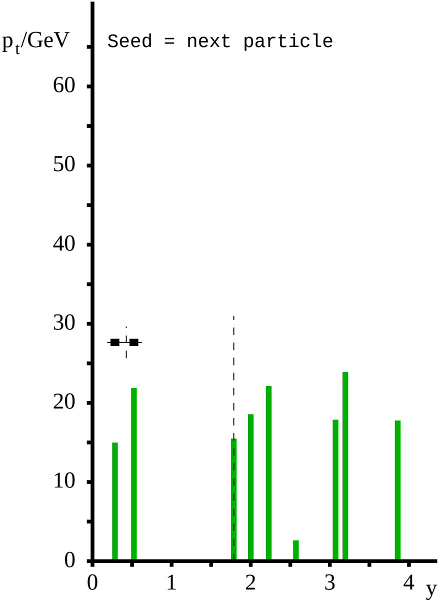
- use every particle as possible seed (no particular order)
- iterate until stable cone
- add the stable cone to the list of **protojets** unless it's already there
- until all seeds done

Note: protojets **overlap**. Certain particles appear in many protojets

protojet ≠ jet

Must resolve the overlaps.
Use a **split–merge** procedure.

It. Cone with Split–Merge (IC-SM)

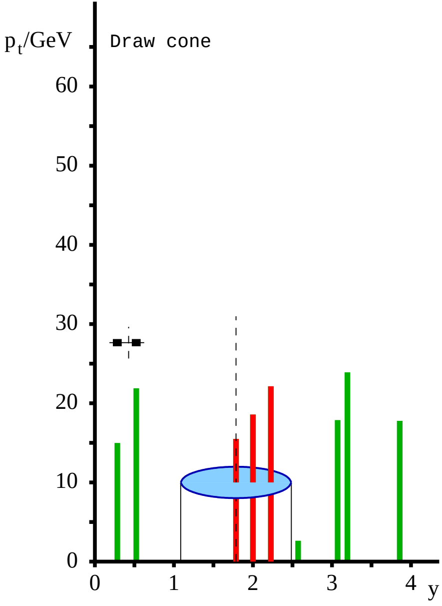


Avoid ordering seeds (coll. unsafe)

CDF JetClu† & ATLAS cones

- use every particle as possible seed (no particular order)
- iterate until stable cone
- add the stable cone to the list of **protojets** unless it's already there
- until all seeds done

It. Cone with Split–Merge (IC-SM)



Avoid ordering seeds (coll. unsafe)

CDF JetClu† & ATLAS cones

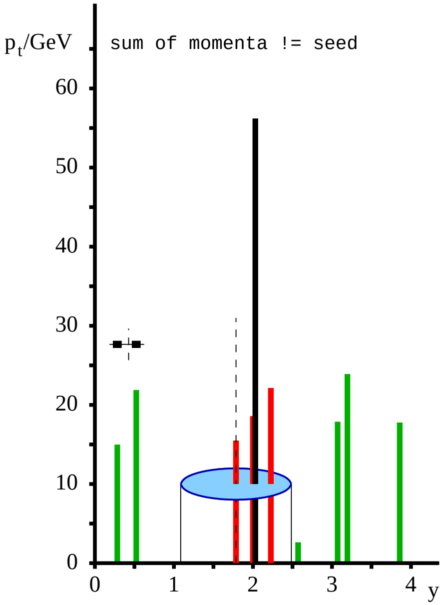
- use every particle as possible seed (no particular order)
- iterate until stable cone
- add the stable cone to the list of **protojets** unless it's already there
- until all seeds done

Note: protojets overlap. Certain particles appear in many protojets

protojet ≠ jet

Must resolve the overlaps.
Use a split–merge procedure.

It. Cone with Split–Merge (IC-SM)



Avoid ordering seeds (coll. unsafe)

CDF JetClu† & ATLAS cones

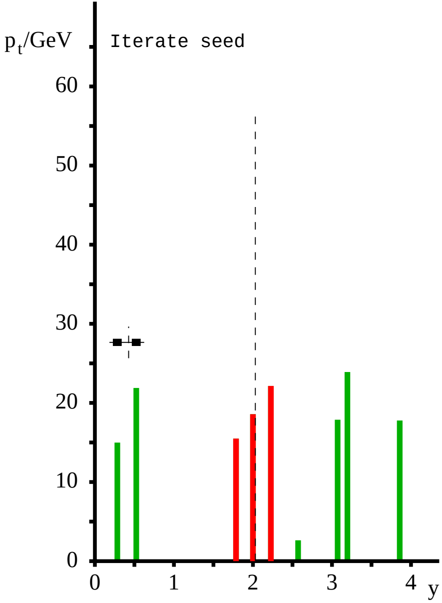
- use every particle as possible seed (no particular order)
- iterate until stable cone
- add the stable cone to the list of **protojets** unless it's already there
- until all seeds done

Note: protojets overlap. Certain particles appear in many protojets

protojet ≠ jet

Must resolve the overlaps.
Use a split–merge procedure.

It. Cone with Split–Merge (IC-SM)



Avoid ordering seeds (coll. unsafe)

CDF JetClu† & ATLAS cones

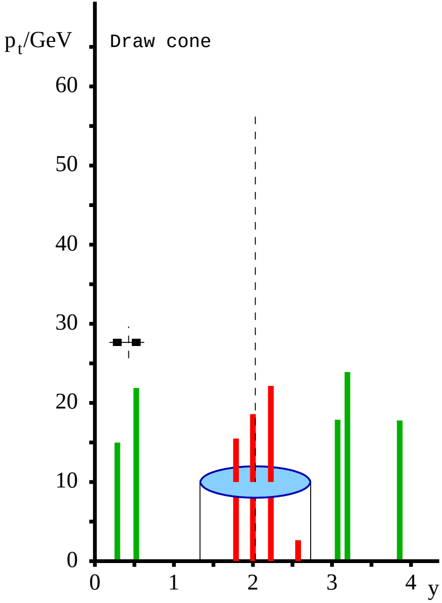
- use every particle as possible seed (no particular order)
- iterate until stable cone
- add the stable cone to the list of **protojets** unless it's already there
- until all seeds done

Note: protojets overlap. Certain particles appear in many protojets

protojet ≠ jet

Must resolve the overlaps.
Use a split–merge procedure.

It. Cone with Split–Merge (IC-SM)



Avoid ordering seeds (coll. unsafe)

CDF JetClu† & ATLAS cones

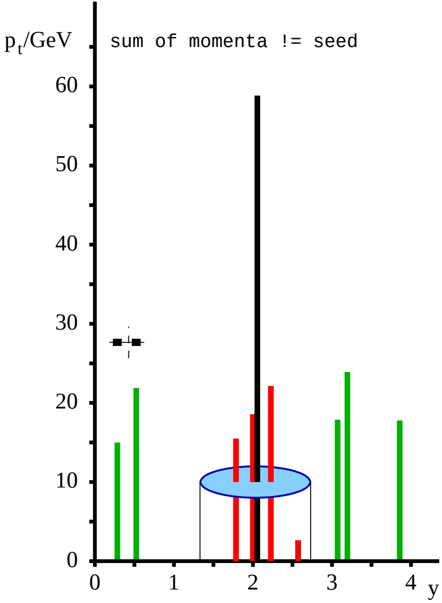
- use every particle as possible seed (no particular order)
- iterate until stable cone
- add the stable cone to the list of **protojets** unless it's already there
- until all seeds done

Note: protojets overlap. Certain particles appear in many protojets

protojet ≠ jet

Must resolve the overlaps.
Use a split–merge procedure.

It. Cone with Split–Merge (IC-SM)

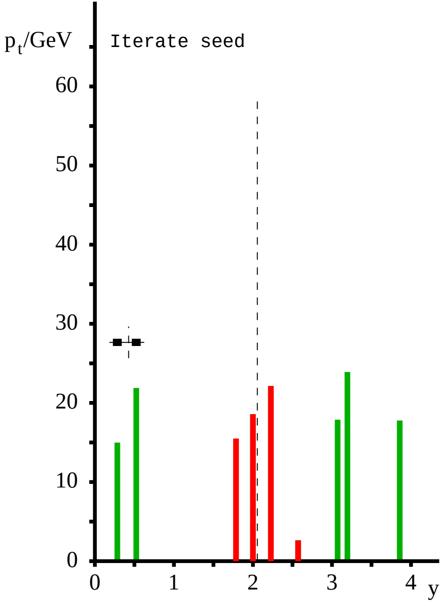


Avoid ordering seeds (coll. unsafe)

CDF JetClu† & ATLAS cones

- use every particle as possible seed (no particular order)
- iterate until stable cone
- add the stable cone to the list of **protojets** unless it's already there
- until all seeds done

It. Cone with Split–Merge (IC-SM)



Avoid ordering seeds (coll. unsafe)

CDF JetClu† & ATLAS cones

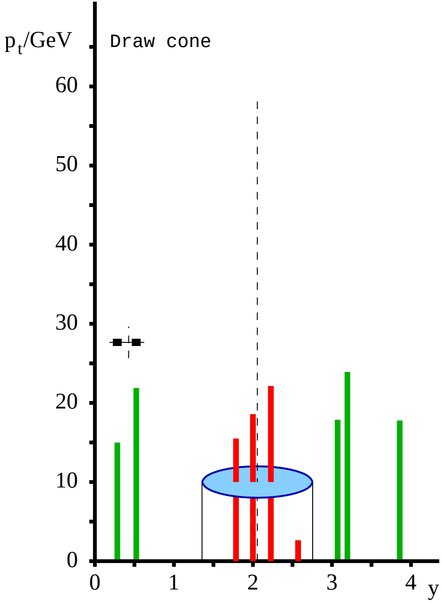
- use every particle as possible seed (no particular order)
- iterate until stable cone
- add the stable cone to the list of **protojets** unless it's already there
- until all seeds done

Note: protojets overlap. Certain particles appear in many protojets

protojet ≠ jet

Must resolve the overlaps.
Use a split–merge procedure.

It. Cone with Split–Merge (IC-SM)



Avoid ordering seeds (coll. unsafe)

CDF JetClu† & ATLAS cones

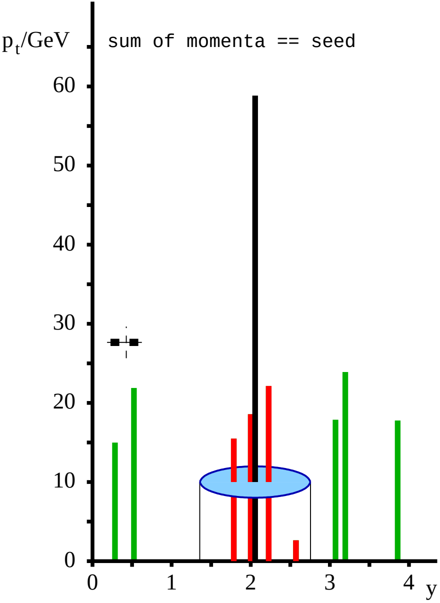
- use every particle as possible seed (no particular order)
- iterate until stable cone
- add the stable cone to the list of **protojets** unless it's already there
- until all seeds done

Note: protojets overlap. Certain particles appear in many protojets

protojet ≠ jet

Must resolve the overlaps.
Use a split–merge procedure.

It. Cone with Split–Merge (IC-SM)



Avoid ordering seeds (coll. unsafe)

CDF JetClu† & ATLAS cones

- use every particle as possible seed (no particular order)
- iterate until stable cone
- add the stable cone to the list of **protojets** unless it's already there
- until all seeds done

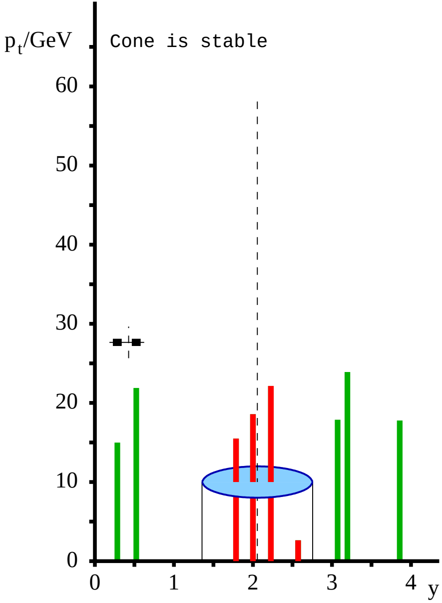
Note: protojets overlap. Certain particles appear in many protojets

protojet ≠ jet

Must resolve the overlaps.

Use a split–merge procedure.

It. Cone with Split–Merge (IC-SM)



Avoid ordering seeds (coll. unsafe)

CDF JetClu† & ATLAS cones

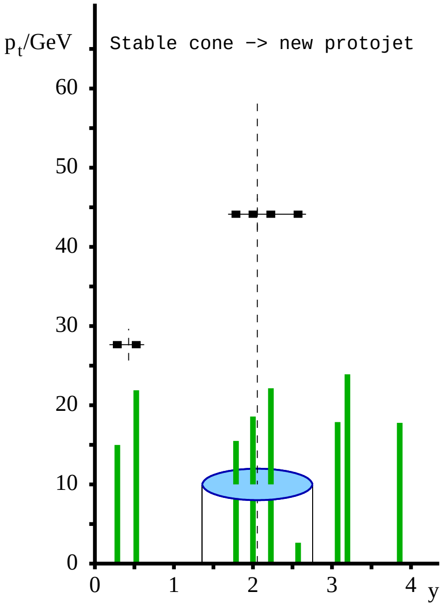
- use every particle as possible seed (no particular order)
- iterate until stable cone
- add the stable cone to the list of **protojets** unless it's already there
- until all seeds done

Note: protojets **overlap**. Certain particles appear in many protojets

protojet ≠ jet

Must resolve the overlaps.
Use a **split–merge** procedure.

It. Cone with Split–Merge (IC-SM)



Avoid ordering seeds (coll. unsafe)

CDF JetClu† & ATLAS cones

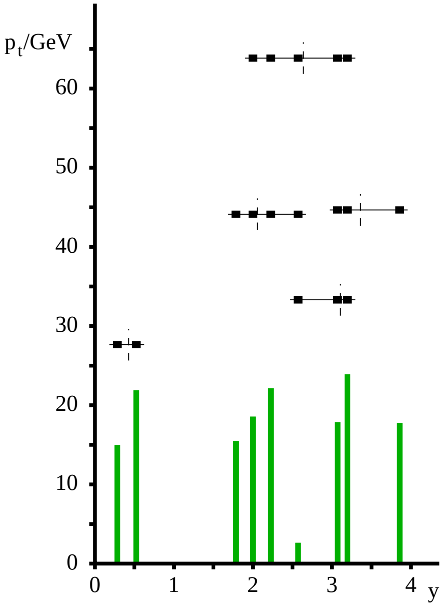
- use every particle as possible seed (no particular order)
- iterate until stable cone
- add the stable cone to the list of **protojets** unless it's already there
- until all seeds done

Note: protojets **overlap**. Certain particles appear in many protojets

protojet $\neq$ jet

Must resolve the overlaps.
Use a **split–merge** procedure.

It. Cone with Split–Merge (IC-SM)



Avoid ordering seeds (coll. unsafe)

CDF JetClu† & ATLAS cones

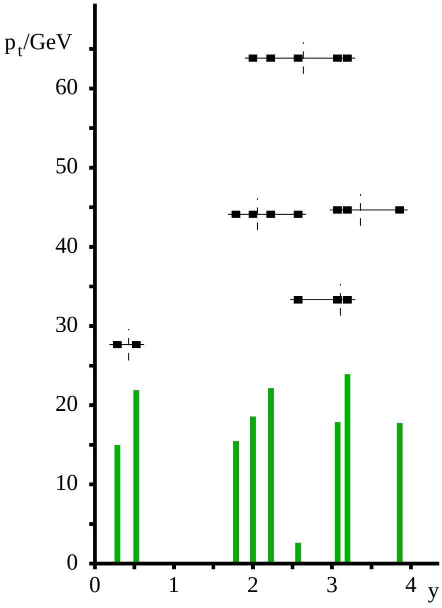
- use every particle as possible seed (no particular order)
- iterate until stable cone
- add the stable cone to the list of **protojets** unless it's already there
- until all seeds done

Note: protojets **overlap**. Certain particles appear in many protojets

protojet $\neq$ jet

Must resolve the overlaps.
Use a **split–merge** procedure.

It. Cone with Split–Merge (IC-SM)



Avoid ordering seeds (coll. unsafe)

CDF JetClu† & ATLAS cones

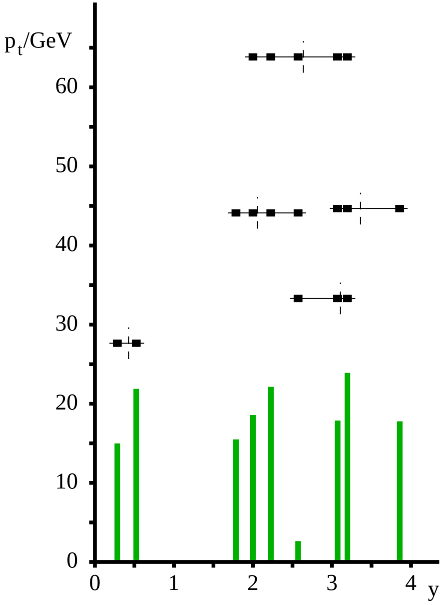
- use every particle as possible seed (no particular order)
- iterate until stable cone
- add the stable cone to the list of **protojets** unless it's already there
- until all seeds done

Note: protojets **overlap**. Certain particles appear in many protojets

protojet ≠ jet

Must resolve the overlaps.
Use a **split–merge** procedure.

IC-SM: split–merge part



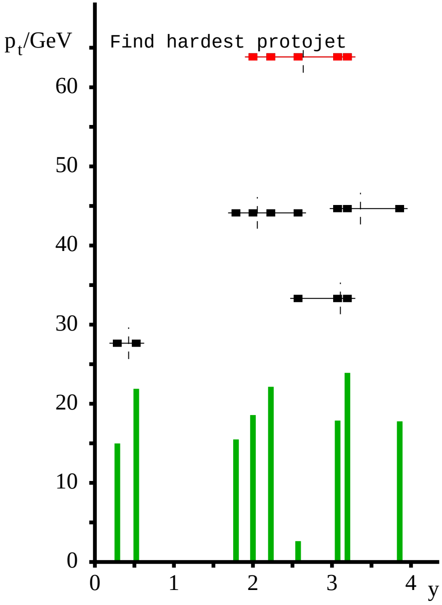
SM in Tevatron Run II formulation

but common to most xC-SM

Introduce overlap threshold f

- Identify hardest protojet (PJ), p_1
- Find hardest PJ that overlaps with it, p_2
- Calculated overlap, $O = p_{t,shared}/p_{t,2}$
  - if $O < f$, split along axis at center of two PJs
  - if $O > f$ merge the two PJs
- If there is no overlap, PJ $\to$ jet.
- repeat. . .

IC-SM: split–merge part



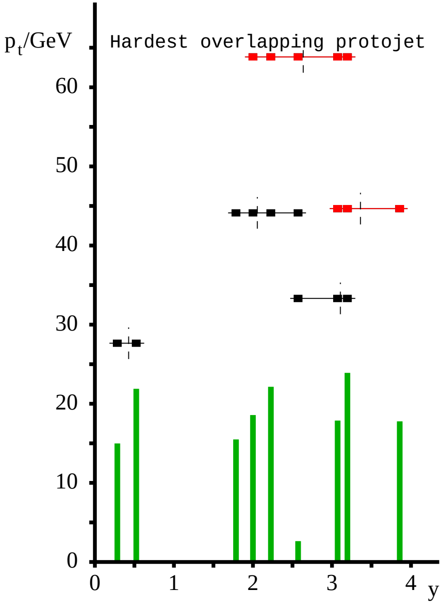
SM in Tevatron Run II formulation

but common to most xC-SM

Introduce overlap threshold f

- Identify hardest protojet (PJ), p_1
- Find hardest PJ that overlaps with it, p_2
- Calculated overlap, $O = p_{t,shared}/p_{t,2}$
  - if $O < f$, split along axis at center of two PJs
  - if $O > f$ merge the two PJs
- If there is no overlap, PJ $\rightarrow$ jet.
- repeat...

IC-SM: split–merge part



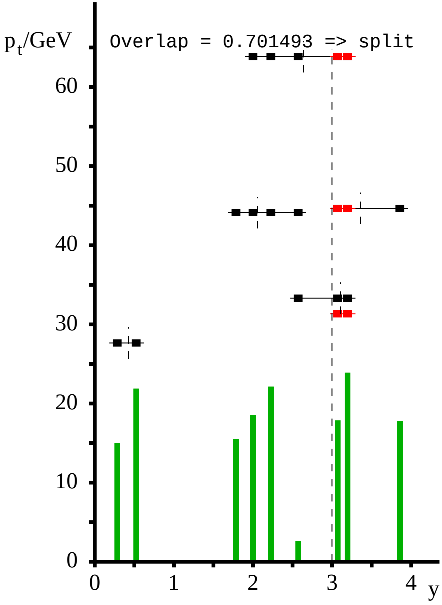
SM in Tevatron Run II formulation

but common to most xC-SM

Introduce overlap threshold f

- Identify hardest protojet (PJ), p_1
- Find hardest PJ that overlaps with it, p_2
- Calculated overlap, $O = p_{t,shared} / p_{t,2}$
  - if $O < f$, split along axis at center of two PJs
  - if $O > f$ merge the two PJs
- If there is no overlap, PJ $\to$ jet.
- repeat. . .

IC-SM: split–merge part



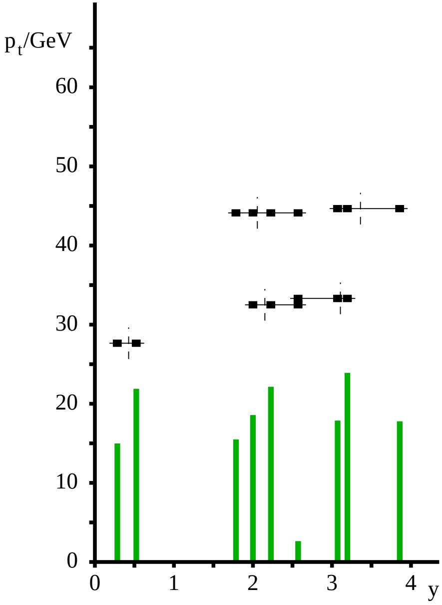
SM in Tevatron Run II formulation

but common to most xC-SM

Introduce overlap threshold f

- Identify hardest protojet (PJ), p_1
- Find hardest PJ that overlaps with it, p_2
- Calculated overlap, $O = p_{t,shared} / p_{t,2}$
  - if $O < f$, split along axis at center of two PJs
  - if $O > f$ merge the two PJs
- If there is no overlap, PJ $\rightarrow$ jet.
- repeat. . .

IC-SM: split–merge part



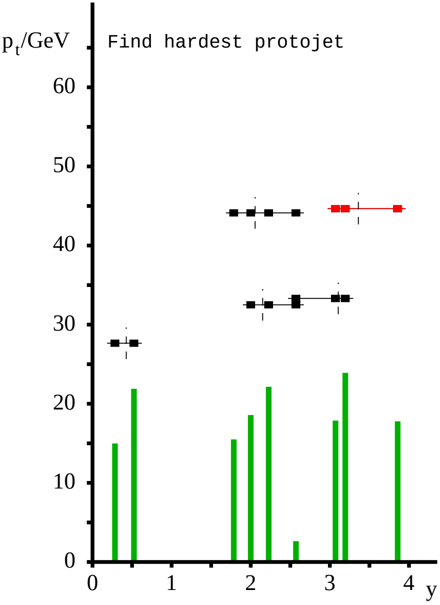
<u>SM in Tevatron Run II formulation</u>

but common to most xC-SM

Introduce overlap threshold f

- Identify hardest protojet (PJ), p_1
- Find hardest PJ that overlaps with it, p_2
- Calculated overlap, $O = p_{t,shared}/p_{t,2}$
  - if $O < f$, split along axis at center of two PJs
  - if $O > f$ merge the two PJs
- If there is no overlap, PJ $\to$ jet.
- repeat...

IC-SM: split–merge part



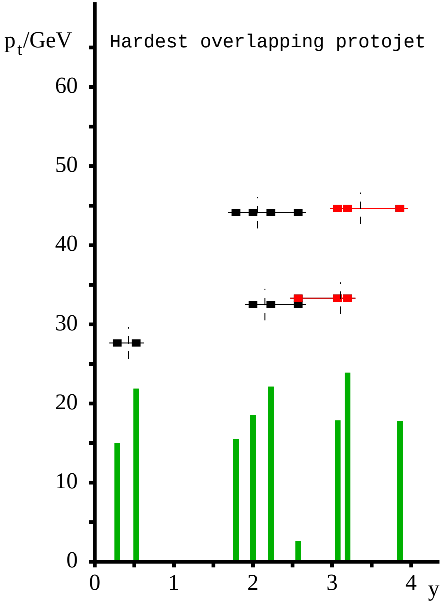
SM in Tevatron Run II formulation

but common to most xC-SM

Introduce overlap threshold f

- Identify hardest protojet (PJ), p_1
- Find hardest PJ that overlaps with it, p_2
- Calculated overlap, $O = p_{t,shared} / p_{t,2}$
  - if $O < f$, split along axis at center of two PJs
  - if $O > f$ merge the two PJs
- If there is no overlap, PJ $\to$ jet.
- repeat. . .

IC-SM: split–merge part



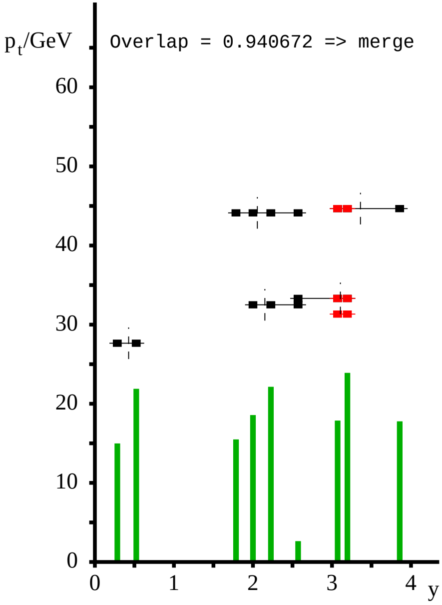
<u>SM in Tevatron Run II formulation</u>

but common to most xC-SM

Introduce overlap threshold f

- Identify hardest protojet (PJ), p_1
- Find hardest PJ that overlaps with it, p_2
- Calculated overlap, $O = p_{t,shared} / p_{t,2}$
  - if $O < f$, split along axis at center of two PJs
  - if $O > f$ merge the two PJs
- If there is no overlap, PJ $\rightarrow$ jet.
- repeat. . .

IC-SM: split–merge part



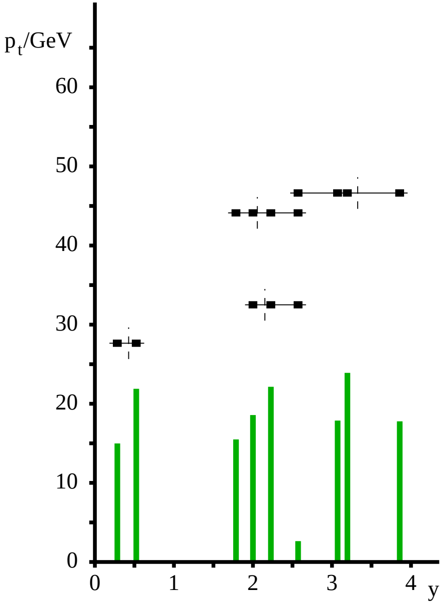
SM in Tevatron Run II formulation

but common to most xC-SM

Introduce overlap threshold f

- Identify hardest protojet (PJ), p_1
- Find hardest PJ that overlaps with it, p_2
- Calculated overlap, $O = p_{t,shared}/p_{t,2}$
  - if $O < f$, split along axis at center of two PJs
  - if $O > f$ merge the two PJs
- If there is no overlap, PJ $\rightarrow$ jet.
- repeat. . .

IC-SM: split–merge part



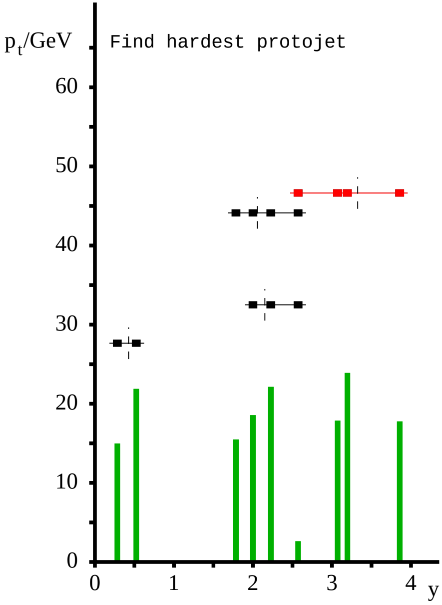
SM in Tevatron Run II formulation

but common to most xC-SM

Introduce overlap threshold f

- Identify hardest protojet (PJ), p_1
- Find hardest PJ that overlaps with it, p_2
- Calculated overlap, $O = p_{t,shared}/p_{t,2}$
  - if $O < f$, split along axis at center of two PJs
  - if $O > f$ merge the two PJs
- If there is no overlap, PJ $\rightarrow$ jet.
- repeat. . .

IC-SM: split–merge part



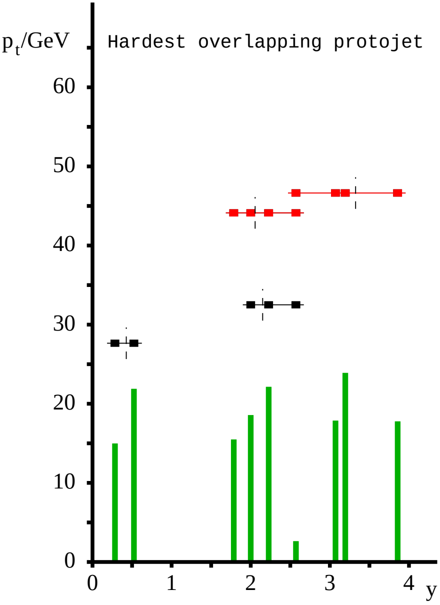
SM in Tevatron Run II formulation

but common to most xC-SM

Introduce overlap threshold f

- Identify hardest protojet (PJ), p_1
- Find hardest PJ that overlaps with it, p_2
- Calculated overlap, $O = p_{t,shared} / p_{t,2}$
  - if $O < f$, split along axis at center of two PJs
  - if $O > f$ merge the two PJs
- If there is no overlap, PJ $\rightarrow$ jet.
- repeat...

IC-SM: split–merge part



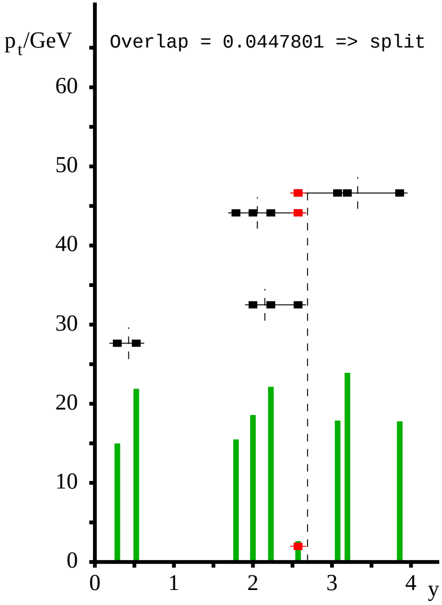
SM in Tevatron Run II formulation

but common to most xC-SM

Introduce overlap threshold f

- Identify hardest protojet (PJ), p_1
- Find hardest PJ that overlaps with it, p_2
- Calculated overlap, $O = p_{t,shared} / p_{t,2}$
  - if $O < f$, split along axis at center of two PJs
  - if $O > f$ merge the two PJs
- If there is no overlap, PJ $\to$ jet.
- repeat. . .

IC-SM: split–merge part



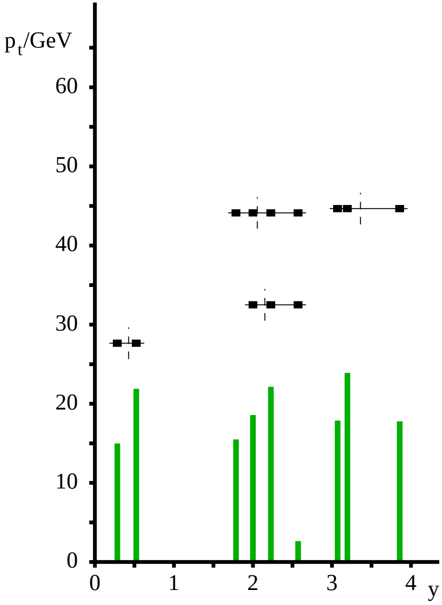
SM in Tevatron Run II formulation

but common to most xC-SM

Introduce overlap threshold f

- Identify hardest protojet (PJ), p_1
- Find hardest PJ that overlaps with it, p_2
- Calculated overlap, $O = p_{t,shared} / p_{t,2}$
  - if $O < f$, split along axis at center of two PJs
  - if $O > f$ merge the two PJs
- If there is no overlap, PJ $\rightarrow$ jet.
- repeat. . .

IC-SM: split–merge part



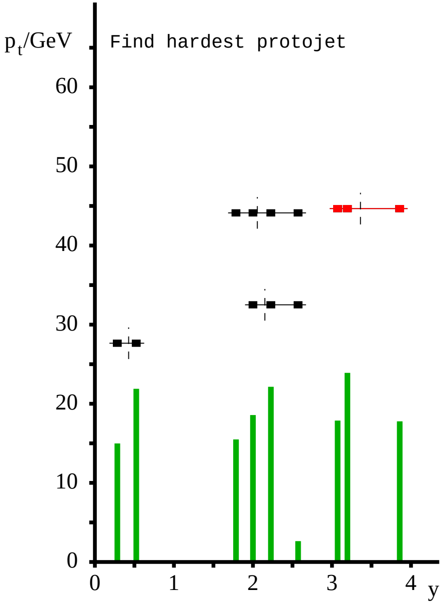
SM in Tevatron Run II formulation

but common to most xC-SM

Introduce overlap threshold f

- Identify hardest protojet (PJ), p_1
- Find hardest PJ that overlaps with it, p_2
- Calculated overlap, $O = p_{t,shared} / p_{t,2}$
  - if $O < f$, split along axis at center of two PJs
  - if $O > f$ merge the two PJs
- If there is no overlap, PJ $\to$ jet.
- repeat. . .

IC-SM: split–merge part



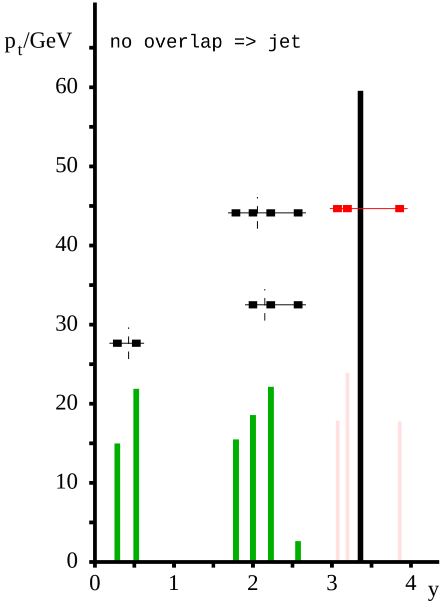
SM in Tevatron Run II formulation

but common to most xC-SM

Introduce overlap threshold f

- Identify hardest protojet (PJ), p_1
- Find hardest PJ that overlaps with it, p_2
- Calculated overlap, $O = p_{t,shared} / p_{t,2}$
  - if $O < f$, split along axis at center of two PJs
  - if $O > f$ merge the two PJs
- If there is no overlap, PJ $\to$ jet.
- repeat...

IC-SM: split–merge part



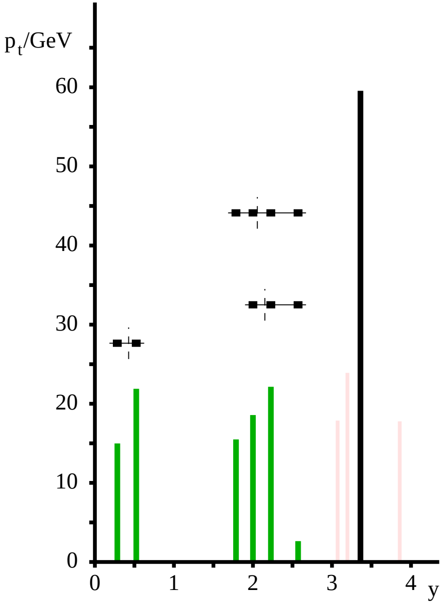
SM in Tevatron Run II formulation

but common to most xC-SM

Introduce overlap threshold f

- Identify hardest protojet (PJ), p_1
- Find hardest PJ that overlaps with it, p_2
- Calculated overlap, $O = p_{t,shared}/p_{t,2}$
  - if $O < f$, split along axis at center of two PJs
  - if $O > f$ merge the two PJs
- If there is no overlap, PJ $\rightarrow$ jet.
- repeat. . .

IC-SM: split–merge part



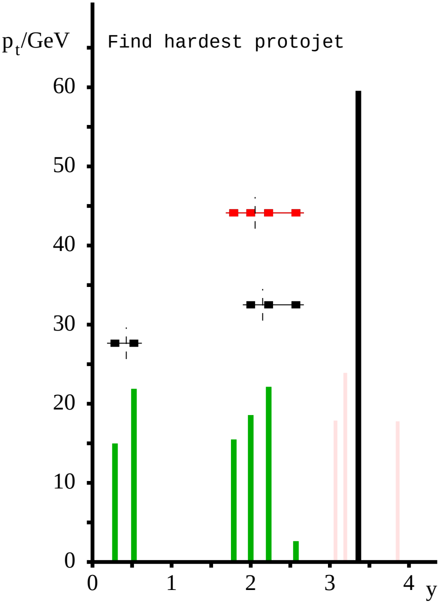
SM in Tevatron Run II formulation

but common to most xC-SM

Introduce overlap threshold f

- Identify hardest protojet (PJ), p_1
- Find hardest PJ that overlaps with it, p_2
- Calculated overlap, $O = p_{t,shared}/p_{t,2}$
  - if $O < f$, split along axis at center of two PJs
  - if $O > f$ merge the two PJs
- If there is no overlap, PJ $\rightarrow$ jet.
- repeat. . .

IC-SM: split–merge part



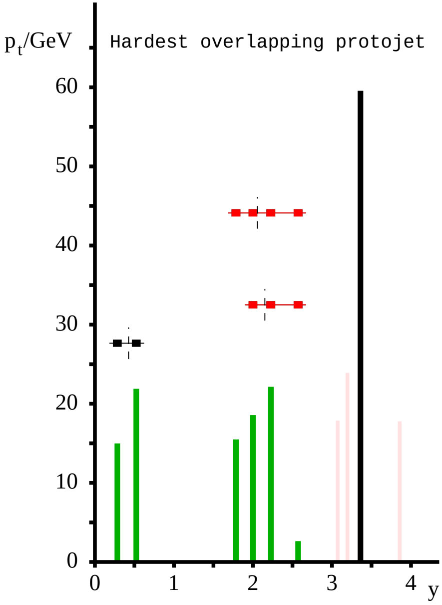
<u>SM in Tevatron Run II formulation</u>

but common to most xC-SM

Introduce overlap threshold f

- Identify hardest protojet (PJ), p_1
- Find hardest PJ that overlaps with it, p_2
- Calculated overlap, $O = p_{t,shared}/p_{t,2}$
  - if $O < f$, split along axis at center of two PJs
  - if $O > f$ merge the two PJs
- If there is no overlap, PJ $\rightarrow$ jet.
- repeat. . .

IC-SM: split–merge part



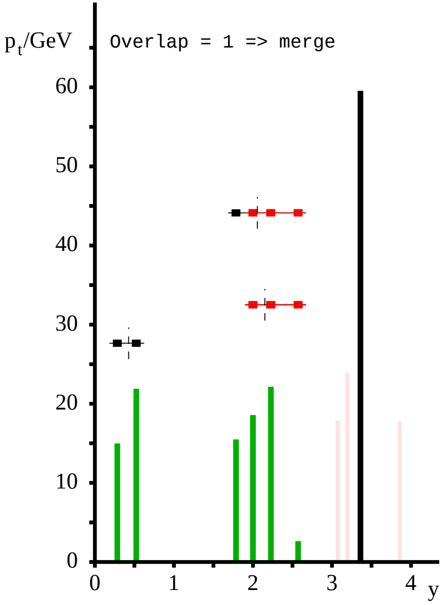
SM in Tevatron Run II formulation

but common to most xC-SM

Introduce overlap threshold f

- Identify hardest protojet (PJ), p_1
- Find hardest PJ that overlaps with it, p_2
- Calculated overlap, $O = p_{t,shared}/p_{t,2}$
  - if $O < f$, split along axis at center of two PJs
  - if $O > f$ merge the two PJs
- If there is no overlap, PJ $\to$ jet.
- repeat. . .

IC-SM: split–merge part



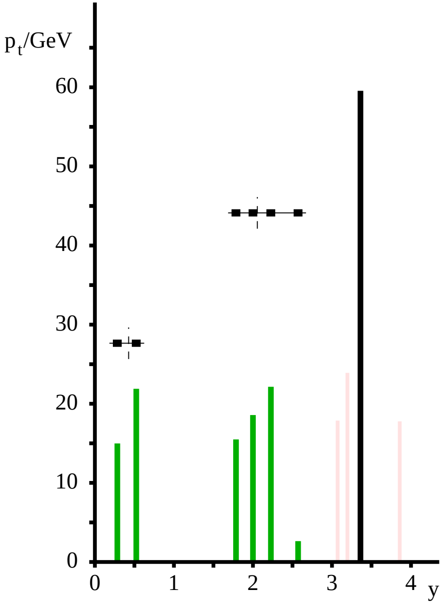
SM in Tevatron Run II formulation

but common to most xC-SM

Introduce overlap threshold f

- Identify hardest protojet (PJ), p_1
- Find hardest PJ that overlaps with it, p_2
- Calculated overlap, $O = p_{t,shared}/p_{t,2}$
  - if $O < f$, split along axis at center of two PJs
  - if $O > f$ merge the two PJs
- If there is no overlap, PJ $\rightarrow$ jet.
- repeat. . .

IC-SM: split–merge part



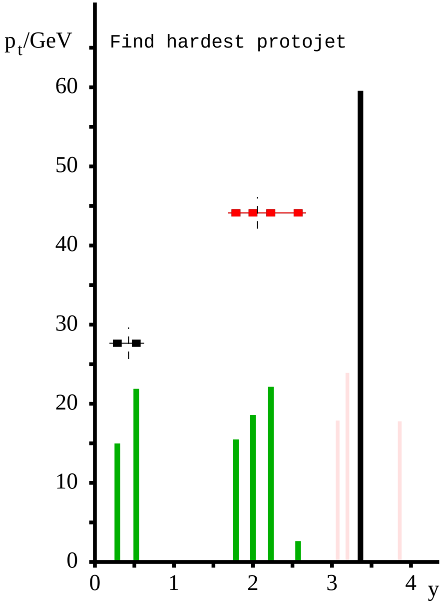
SM in Tevatron Run II formulation

but common to most xC-SM

Introduce overlap threshold f

- Identify hardest protojet (PJ), p_1
- Find hardest PJ that overlaps with it, p_2
- Calculated overlap, $O = p_{t,shared} / p_{t,2}$
  - if $O < f$, split along axis at center of two PJs
  - if $O > f$ merge the two PJs
- If there is no overlap, PJ $\rightarrow$ jet.
- repeat. . .

IC-SM: split–merge part



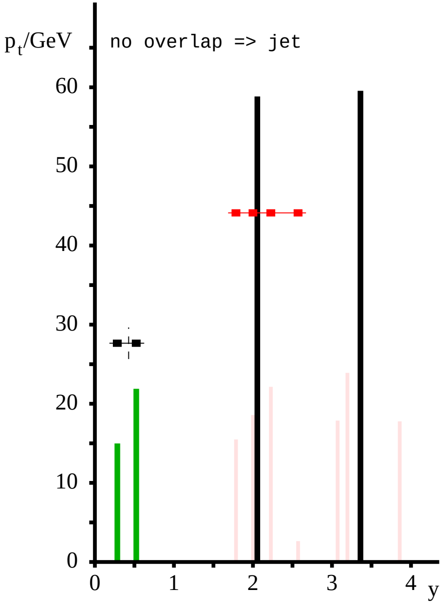
SM in Tevatron Run II formulation

but common to most xC-SM

Introduce overlap threshold f

- Identify hardest protojet (PJ), p_1
- Find hardest PJ that overlaps with it, p_2
- Calculated overlap, $O = p_{t,shared}/p_{t,2}$
  - if $O < f$, split along axis at center of two PJs
  - if $O > f$ merge the two PJs
- If there is no overlap, PJ $\rightarrow$ jet.
- repeat. . .

IC-SM: split–merge part



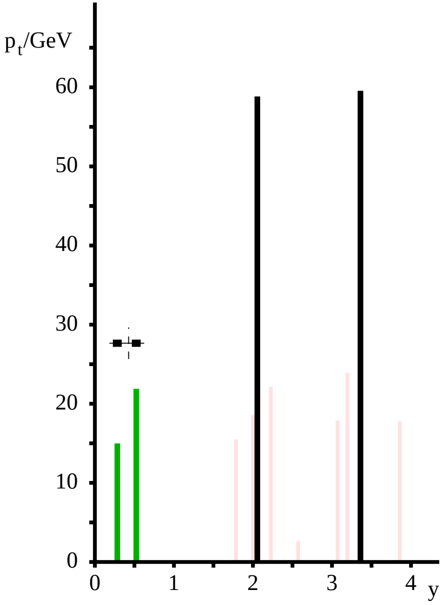
SM in Tevatron Run II formulation

but common to most xC-SM

Introduce overlap threshold f

- Identify hardest protojet (PJ), p_1
- Find hardest PJ that overlaps with it, p_2
- Calculated overlap, $O = p_{t,shared}/p_{t,2}$
  - if $O < f$, split along axis at center of two PJs
  - if $O > f$ merge the two PJs
- If there is no overlap, PJ $\rightarrow$ jet.
- repeat. . .

IC-SM: split–merge part



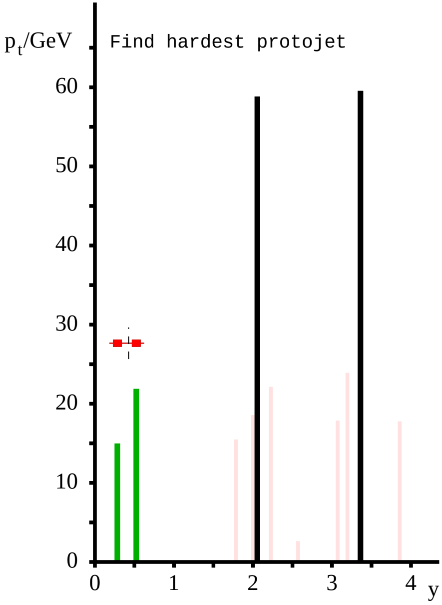
SM in Tevatron Run II formulation

but common to most xC-SM

Introduce overlap threshold f

- Identify hardest protojet (PJ), p_1
- Find hardest PJ that overlaps with it, p_2
- Calculated overlap, $O = p_{t,shared}/p_{t,2}$
  - if $O < f$, split along axis at center of two PJs
  - if $O > f$ merge the two PJs
- If there is no overlap, PJ $\to$ jet.
- repeat. . .

IC-SM: split–merge part



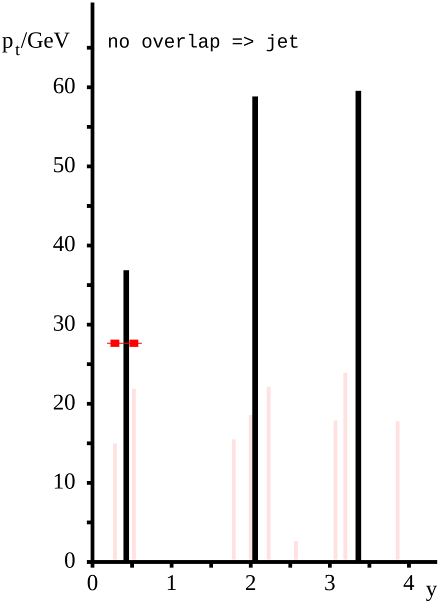
SM in Tevatron Run II formulation

but common to most xC-SM

Introduce overlap threshold f

- Identify hardest protojet (PJ), p_1
- Find hardest PJ that overlaps with it, p_2
- Calculated overlap, $O = p_{t,shared} / p_{t,2}$
  - if $O < f$, split along axis at center of two PJs
  - if $O > f$ merge the two PJs
- If there is no overlap, PJ $\to$ jet.
- repeat. . .

IC-SM: split–merge part



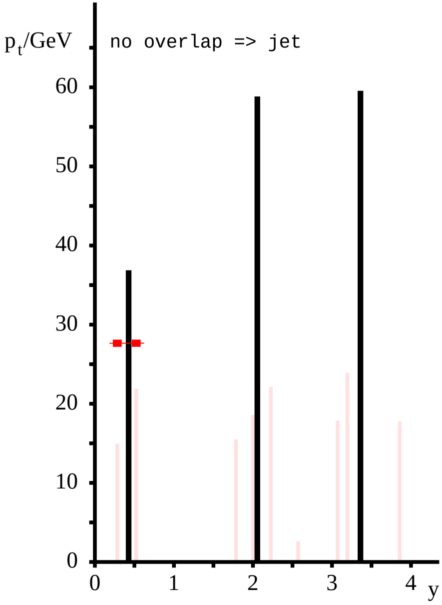
SM in Tevatron Run II formulation

but common to most xC-SM

Introduce overlap threshold f

- Identify hardest protojet (PJ), p_1
- Find hardest PJ that overlaps with it, p_2
- Calculated overlap, $O = p_{t,shared} / p_{t,2}$
  - if $O < f$, split along axis at center of two PJs
  - if $O > f$ merge the two PJs
- If there is no overlap, PJ $\rightarrow$ jet.
- repeat. . .

IC-SM: split–merge part



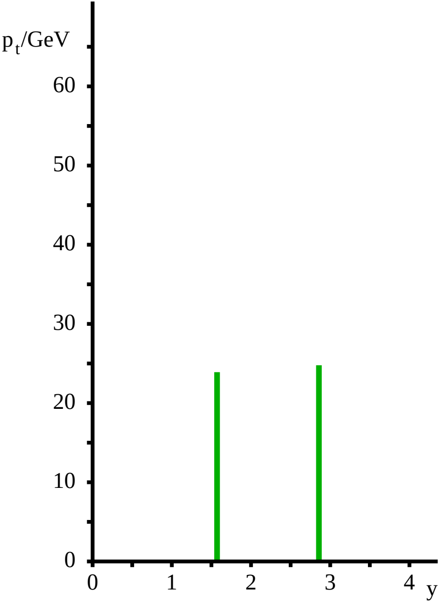
SM in Tevatron Run II formulation

but common to most xC-SM

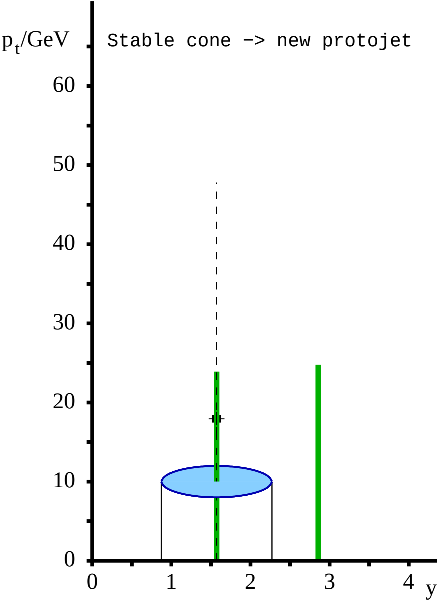
Introduce overlap threshold f

- Identify hardest protojet (PJ), p_1
- Find hardest PJ that overlaps with it, p_2
- Calculated overlap, $O = p_{t,shared}/p_{t,2}$
  - if $O < f$, split along axis at center of two PJs
  - if $O > f$ merge the two PJs
- If there is no overlap, PJ $\rightarrow$ jet.
- repeat. . .

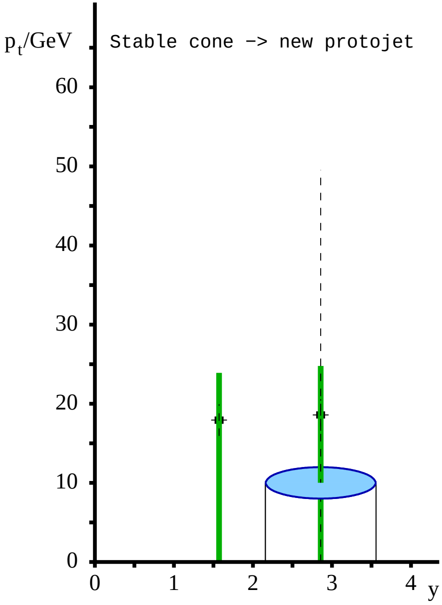
IC-SM: infrared safety



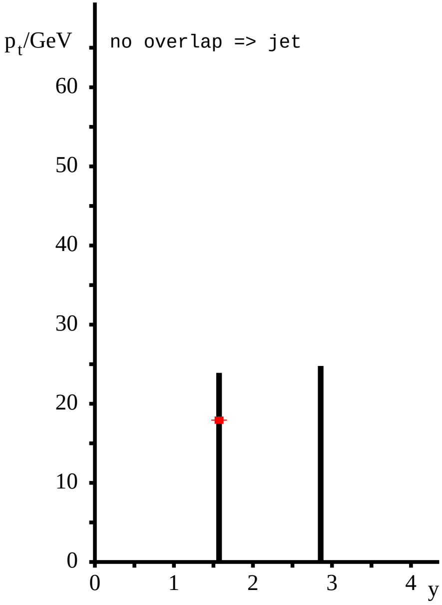
IC-SM: infrared safety



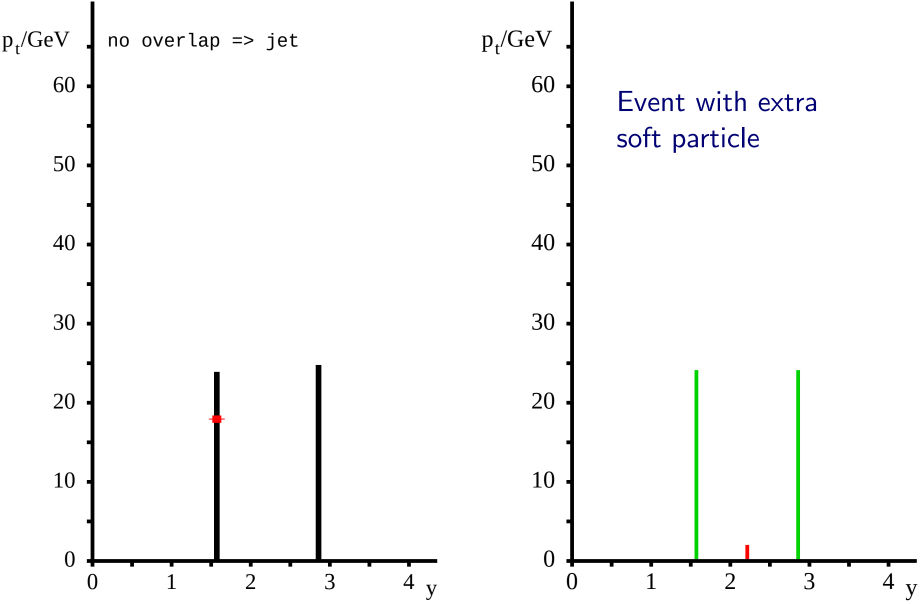
IC-SM: infrared safety



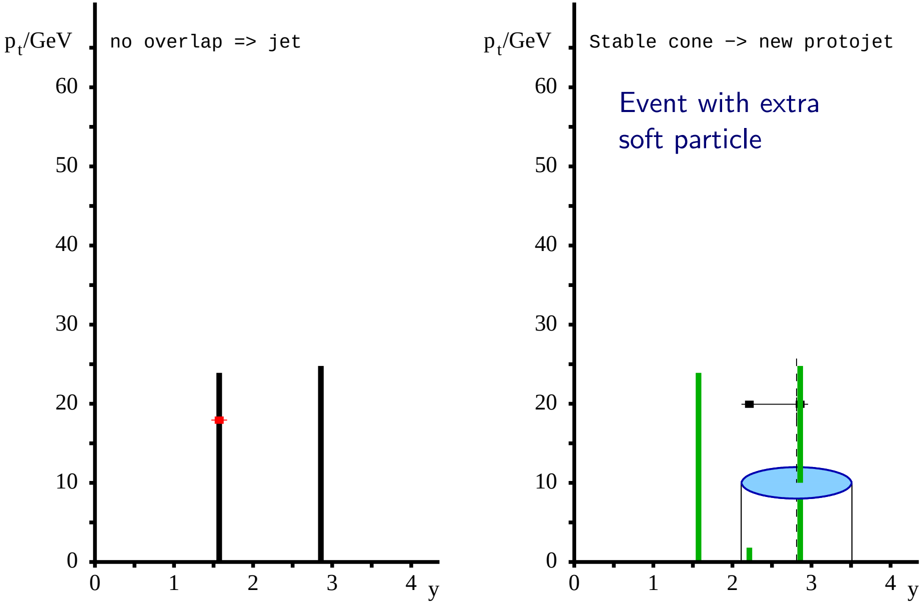
IC-SM: infrared safety



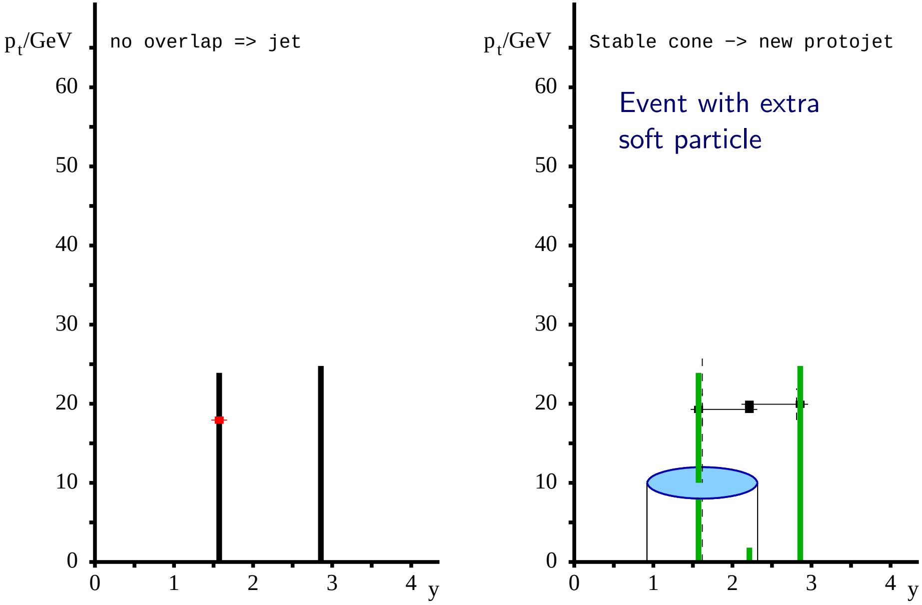
IC-SM: infrared safety



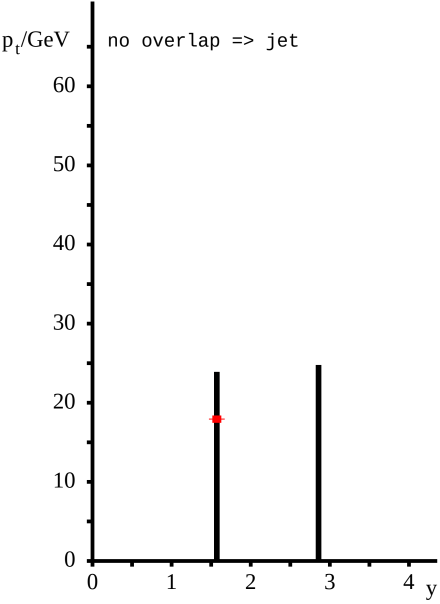
IC-SM: infrared safety

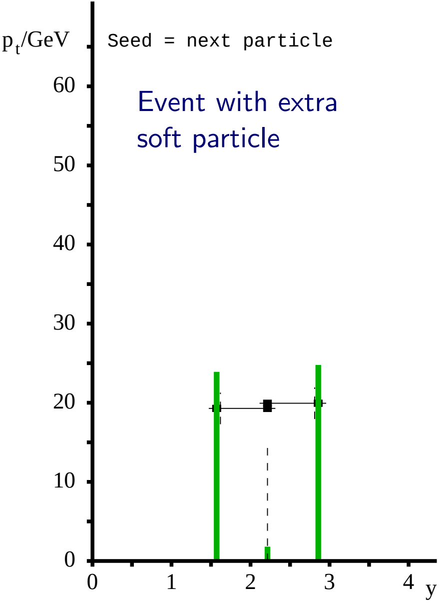


IC-SM: infrared safety

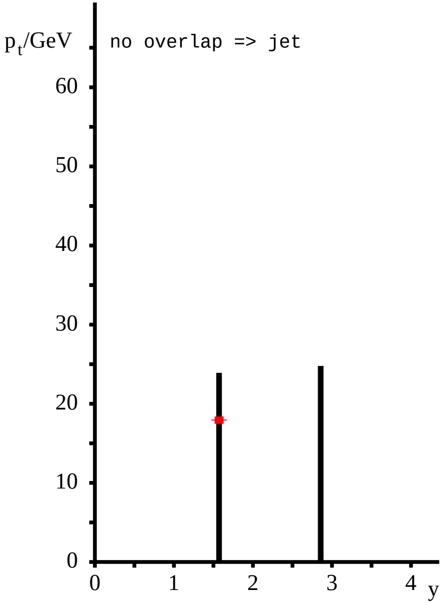


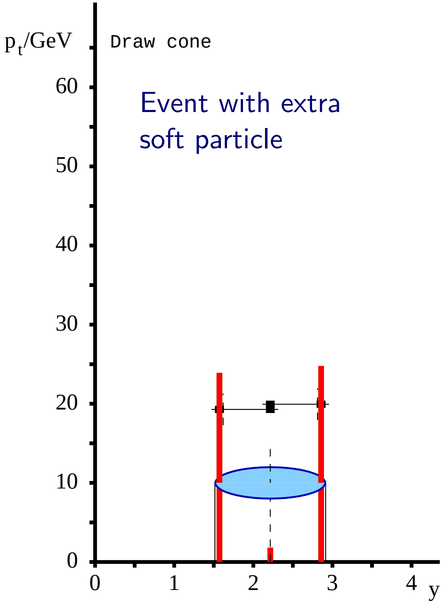
IC-SM: infrared safety



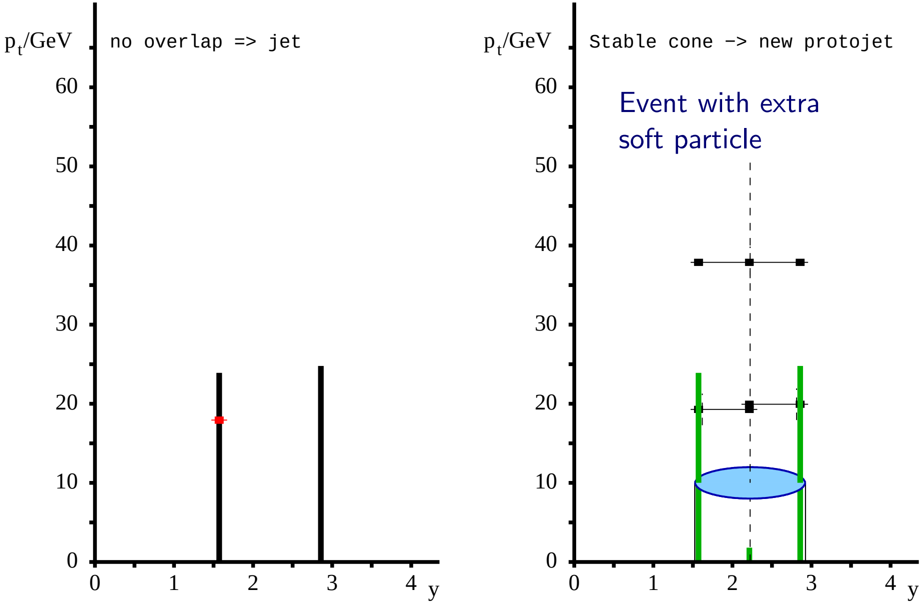


IC-SM: infrared safety

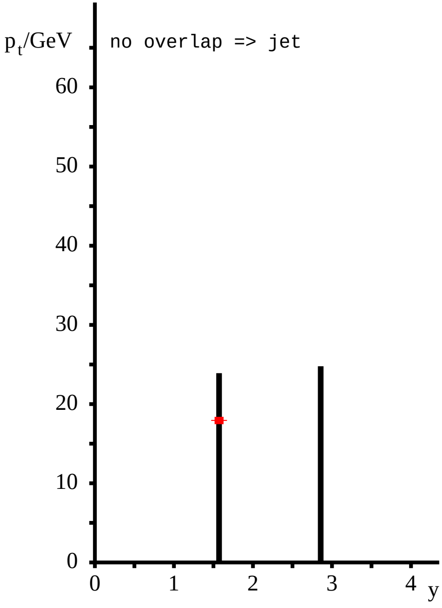


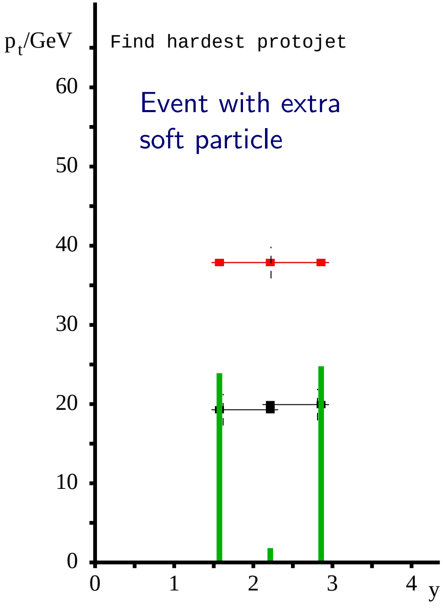


IC-SM: infrared safety



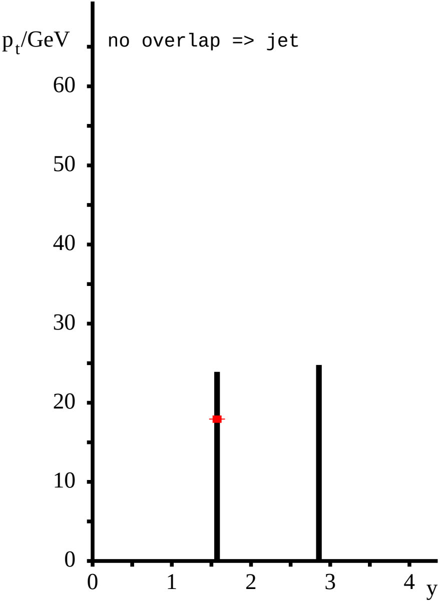
IC-SM: infrared safety

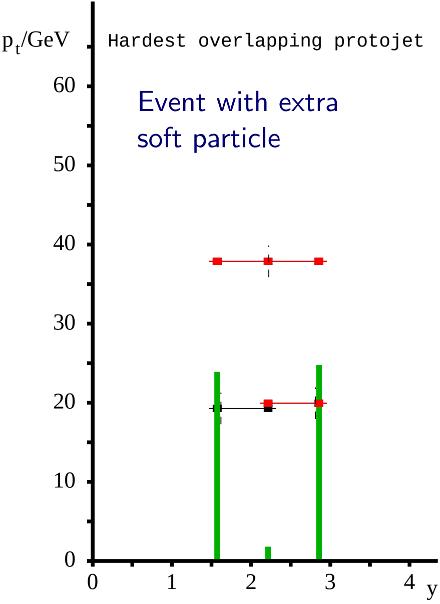




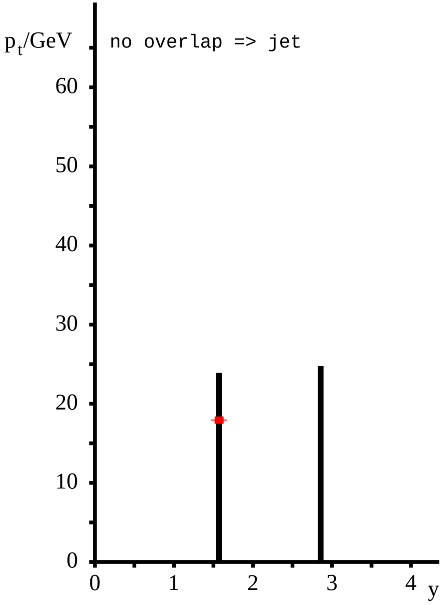
Event with extra soft particle

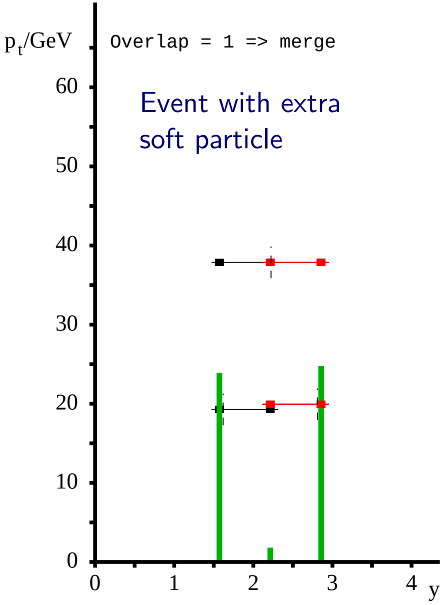
IC-SM: infrared safety



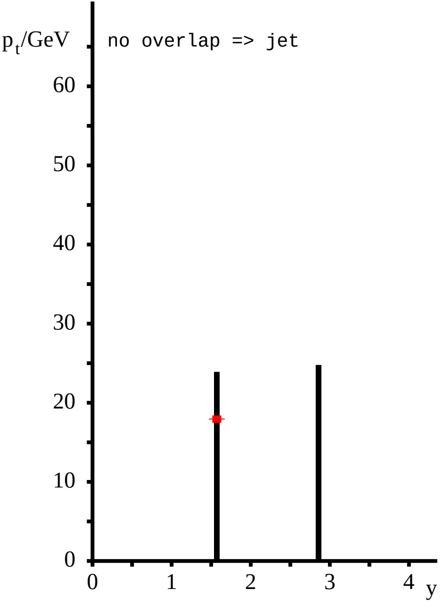


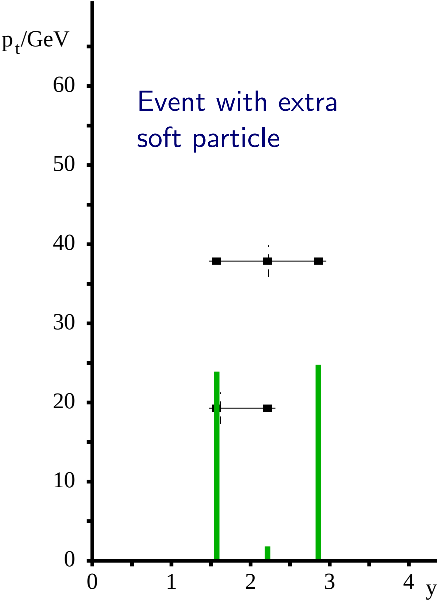
IC-SM: infrared safety



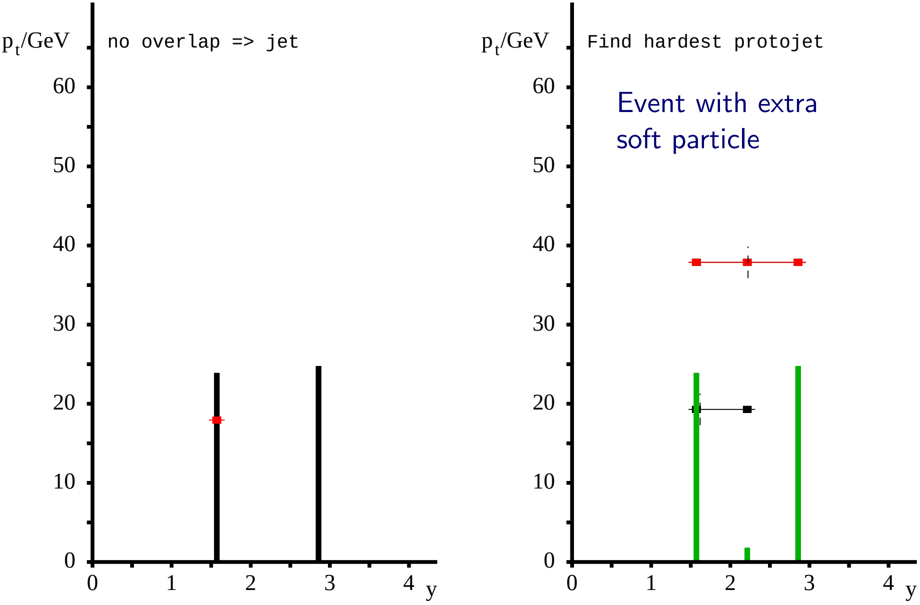


IC-SM: infrared safety

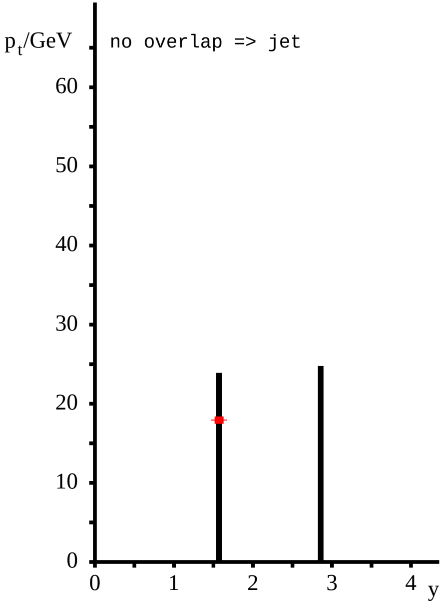


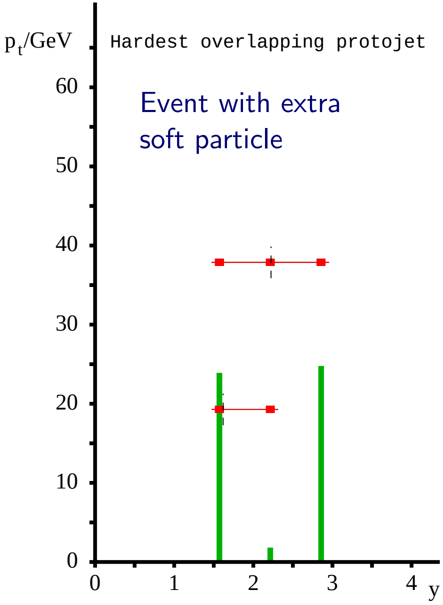


IC-SM: infrared safety

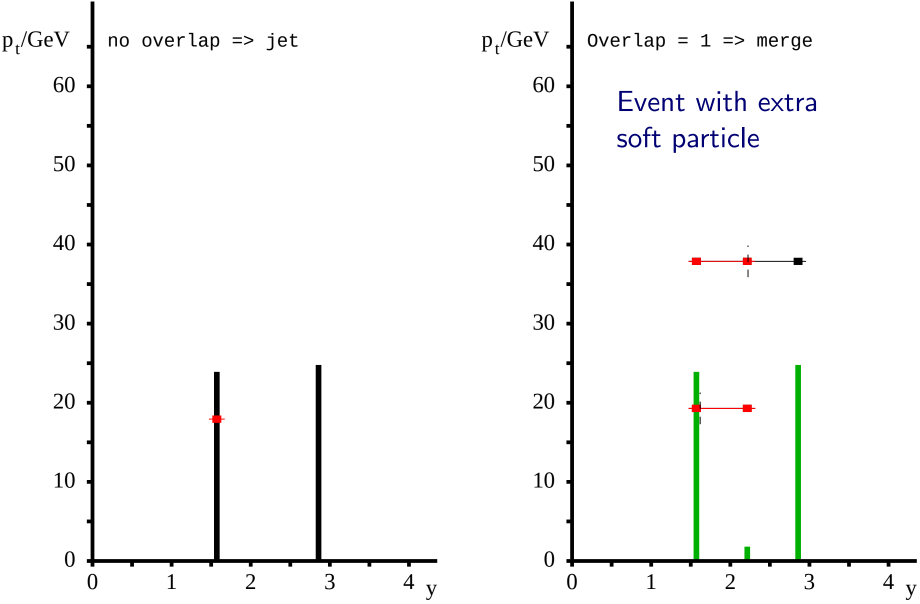


IC-SM: infrared safety

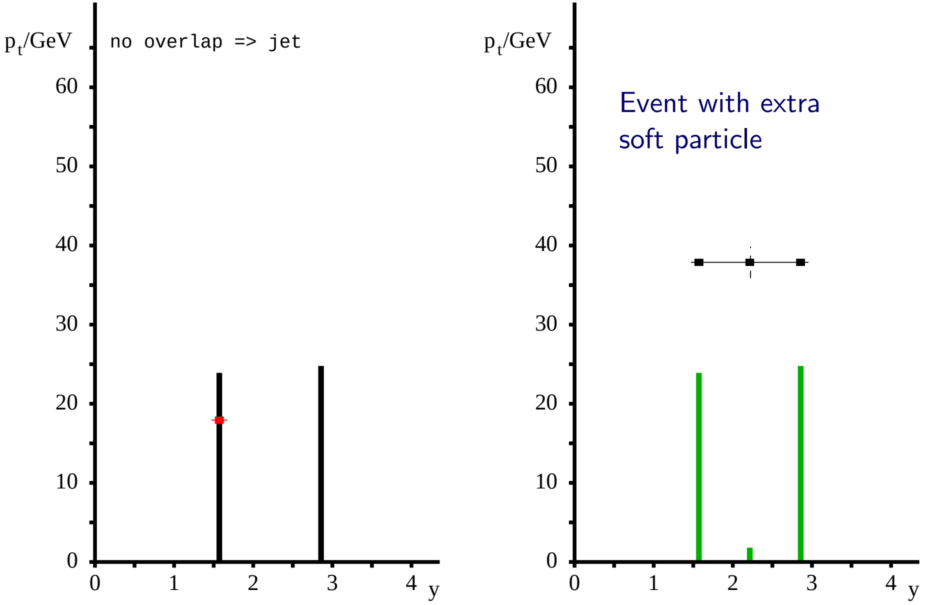




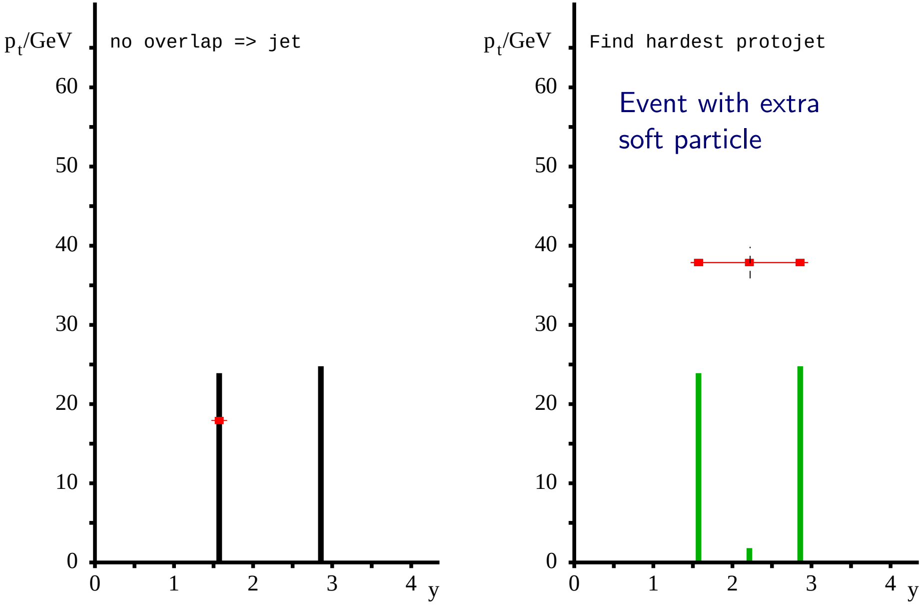
IC-SM: infrared safety



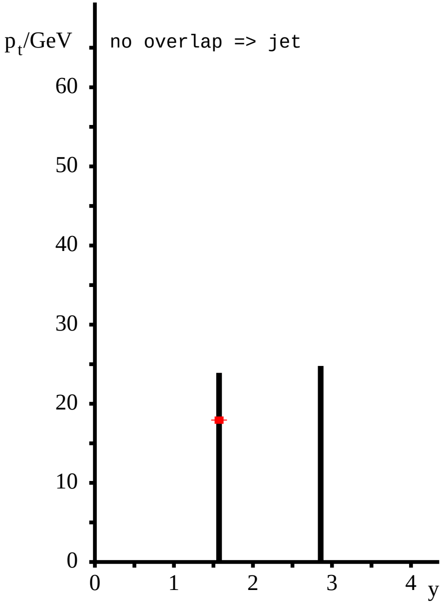
IC-SM: infrared safety

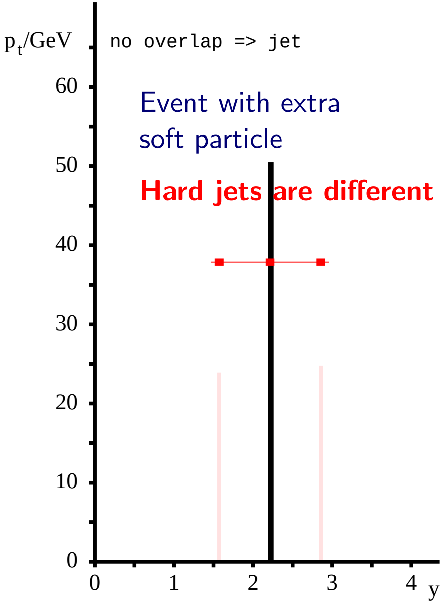


IC-SM: infrared safety



IC-SM: infrared safety

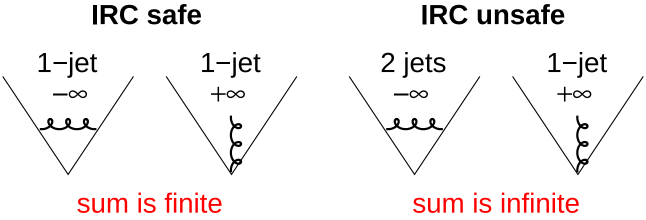




IRC safety crucial for theory

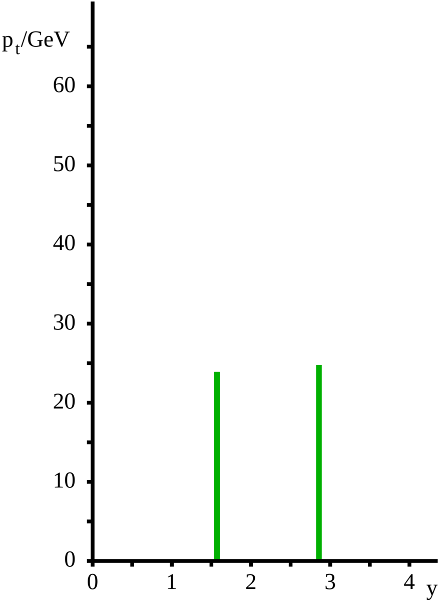
Soft emission, collinear splitting are both **infinite** in pert. QCD.

Infinities **cancel** with loop diagrams if jet-alg IRC safe



Some calculations simply become **meaningless**

Midpoint algorithm (IC_{mp}-SM)



Looking for stable cones ≃ finding local minima of a potential.

Problem: set of iterative solution depends on set of starting points.

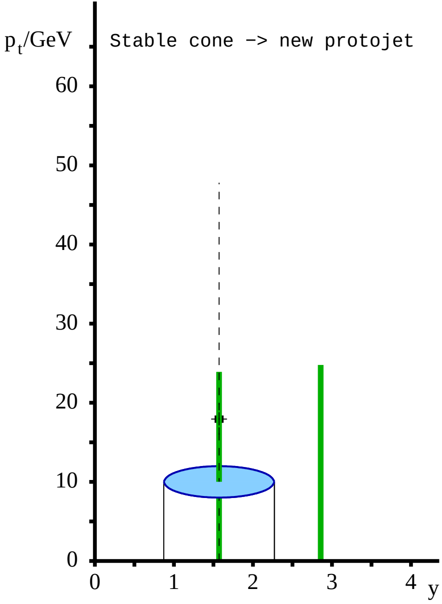
Patch: after 1st round of iteration, find midpoints between protojets, use as new seeds

CDF Midpoint algorithm
D0 Run II algorithm

This solves problem for 2-hard-particle configs.

[But it persists for 3-hard]

Midpoint algorithm (IC_{mp}-SM)



Looking for stable cones $\simeq$ finding local minima of a potential.

Problem: set of iterative solution depends on set of starting points.

Patch: after 1st round of iteration, find midpoints between protojets, use as new seeds

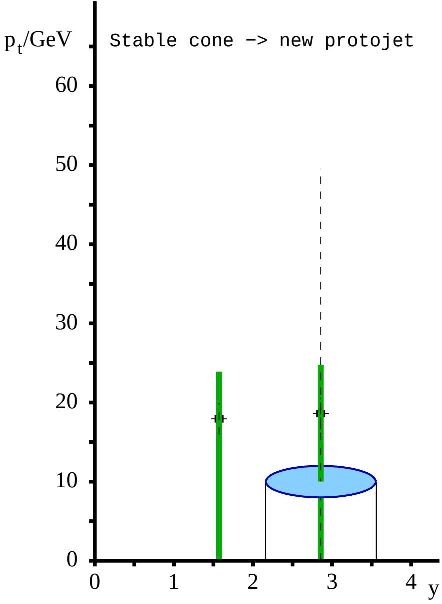
CDF Midpoint algorithm

D0 Run II algorithm

This solves problem for 2-hard-particle configs.

[But it persists for 3-hard]

Midpoint algorithm (IC$_{mp}$-SM)



Looking for stable cones $\simeq$ finding local minima of a potential.

Problem: set of iterative solution depends on set of starting points.

Patch: after 1st round of iteration, find midpoints between protojets, use as new seeds

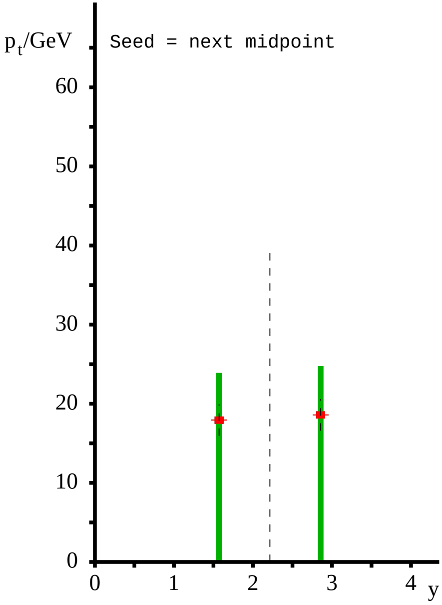
CDF Midpoint algorithm

D0 Run II algorithm

This solves problem for 2-hard-particle configs.

[But it persists for 3-hard]

Midpoint algorithm (IC_{mp}-SM)



Looking for stable cones ≃ finding local minima of a potential.

Problem: set of iterative solution depends on set of starting points.

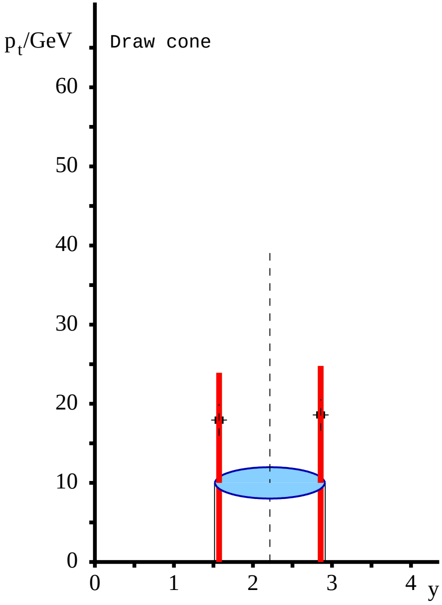
Patch: after 1st round of iteration, find midpoints between protojets, use as new seeds

CDF Midpoint algorithm
D0 Run II algorithm

This solves problem for 2-hard-particle configs.

[But it persists for 3-hard]

Midpoint algorithm (IC$_{mp}$-SM)



Looking for stable cones ≃ finding local minima of a potential.

Problem: set of iterative solution depends on set of starting points.

Patch: after 1st round of iteration, find midpoints between protojets, use as new seeds

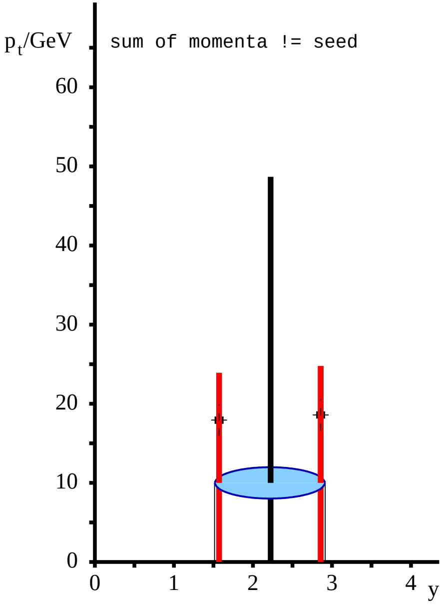
CDF Midpoint algorithm

D0 Run II algorithm

This solves problem for 2-hard-particle configs.

[But it persists for 3-hard]

Midpoint algorithm (IC_{mp}-SM)



Looking for stable cones ≃ finding local minima of a potential.

Problem: set of iterative solution depends on set of starting points.

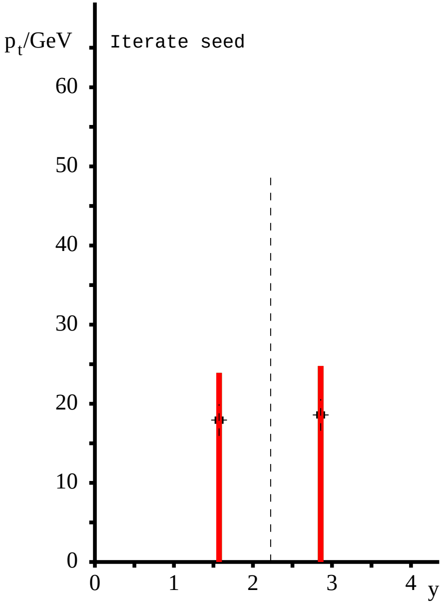
Patch: after 1st round of iteration, find midpoints between protojets, use as new seeds

CDF Midpoint algorithm
D0 Run II algorithm

This solves problem for 2-hard-particle configs.

[But it persists for 3-hard]

Midpoint algorithm (IC$_{mp}$-SM)



Looking for stable cones ≃ finding local minima of a potential.

Problem: set of iterative solution depends on set of starting points.

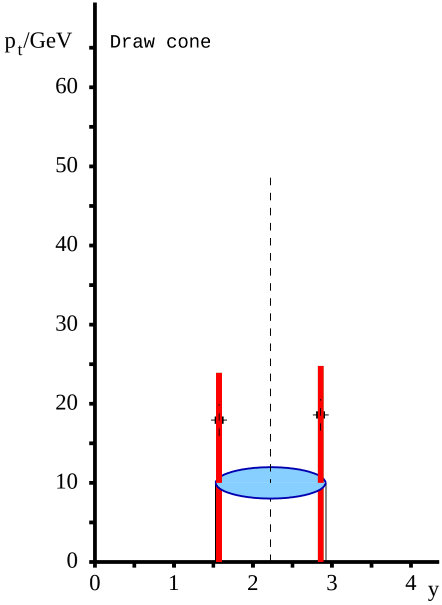
Patch: after 1st round of iteration, find midpoints between protojets, use as new seeds

CDF Midpoint algorithm
D0 Run II algorithm

This solves problem for 2-hard-particle configs.

[But it persists for 3-hard]

Midpoint algorithm (IC$_{mp}$-SM)



Looking for stable cones ≃ finding local minima of a potential.

Problem: set of iterative solution depends on set of starting points.

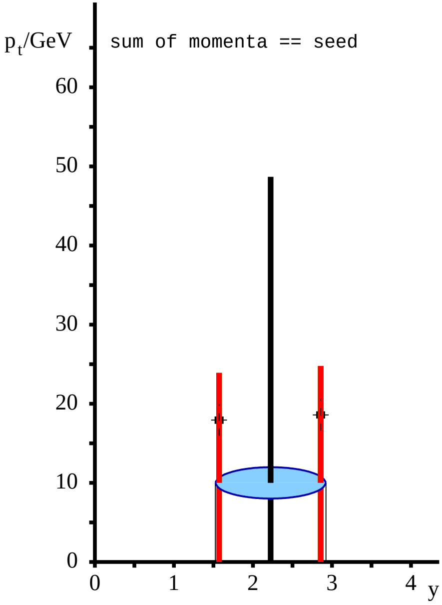
Patch: after 1st round of iteration, find midpoints between protojets, use as new seeds

CDF Midpoint algorithm
D0 Run II algorithm

This solves problem for 2-hard-particle configs.

[But it persists for 3-hard]

Midpoint algorithm (IC_{mp}-SM)



Looking for stable cones $\simeq$ finding local minima of a potential.

Problem: set of iterative solution depends on set of starting points.

Patch: after 1st round of iteration, find midpoints between protojets, use as new seeds

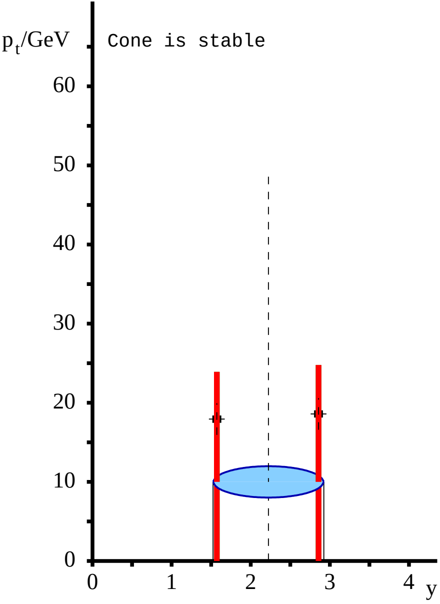
CDF Midpoint algorithm

D0 Run II algorithm

This solves problem for 2-hard-particle configs.

[But it persists for 3-hard]

Midpoint algorithm (IC_{mp}-SM)



Looking for stable cones ≃ finding local minima of a potential.

Problem: set of iterative solution depends on set of starting points.

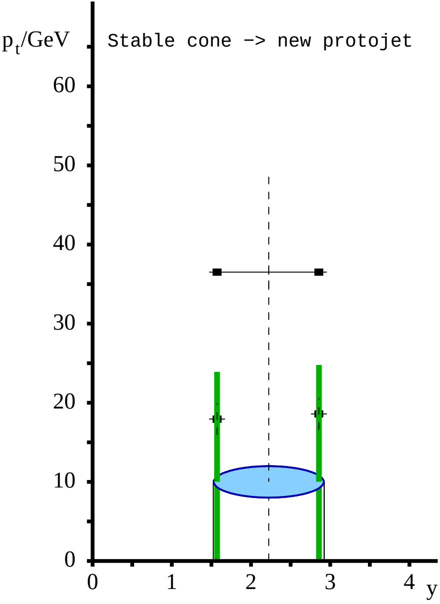
Patch: after 1st round of iteration, find midpoints between protojets, use as new seeds

CDF Midpoint algorithm
D0 Run II algorithm

This solves problem for 2-hard-particle configs.

[But it persists for 3-hard]

Midpoint algorithm (IC_{mp}-SM)



Looking for stable cones ≃ finding local minima of a potential.

Problem: set of iterative solution depends on set of starting points.

Patch: after 1st round of iteration, find midpoints between protojets, use as new seeds

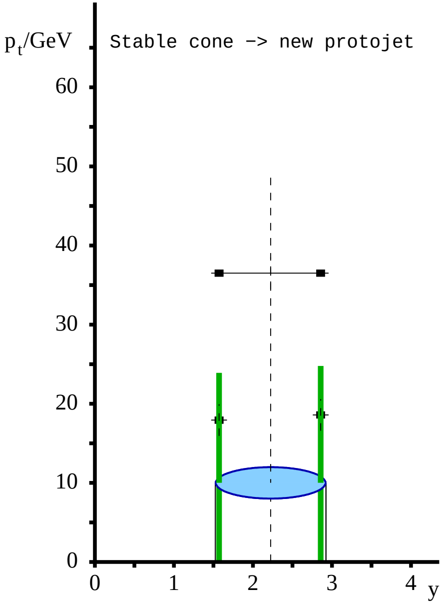
CDF Midpoint algorithm

D0 Run II algorithm

This solves problem for 2-hard-particle configs.

[But it persists for 3-hard]

Midpoint algorithm (IC_{mp}-SM)



Looking for stable cones ≃ finding local minima of a potential.

Problem: set of iterative solution depends on set of starting points.

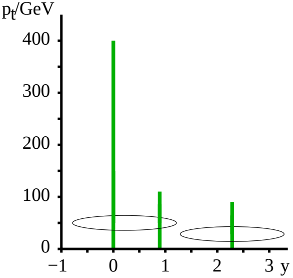
Patch: after 1st round of iteration, find midpoints between protojets, use as new seeds

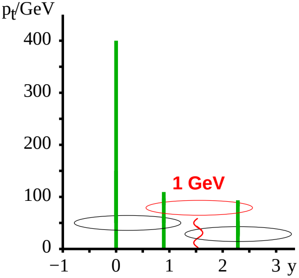
CDF Midpoint algorithm
D0 Run II algorithm

This solves problem for 2-hard-particle configs.

[But it persists for 3-hard]

Midpoint IR problem





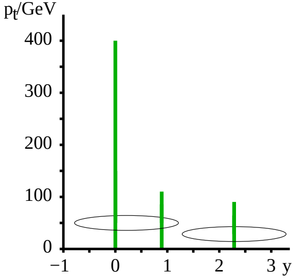
Stable cones with midpoint:	{1,2} & {3}	{1,2} & {2,3} & {3}
Jets with midpoint ($f = 0.5$)	{1,2} & {3}	{1,2,3}

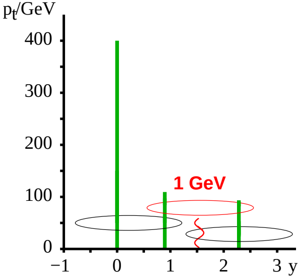
Midpoint cone alg. misses some stable cones; extra soft particle → extra starting point → extra stable cone found

MIDPOINT IS INFRARED UNSAFE

Or collinear unsafe with seed threshold

Midpoint IR problem





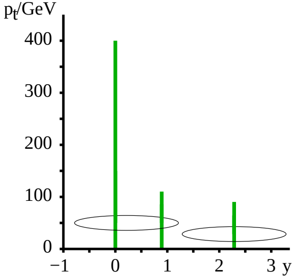
Stable cones with midpoint:	{1,2} & {3}	{1,2} & {2,3} & {3}
Jets with midpoint ($f = 0.5$)	{1,2} & {3}	{1,2,3}

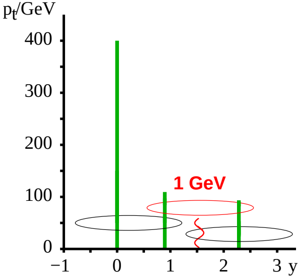
Midpoint cone alg. misses some stable cones; extra soft particle → extra starting point → extra stable cone found

MIDPOINT IS INFRARED UNSAFE

Or collinear unsafe with seed threshold

Midpoint IR problem





Stable cones with midpoint:	{1,2} & {3}	{1,2} & {2,3} & {3}
Jets with midpoint ($f = 0.5$)	{1,2} & {3}	{1,2,3}

Midpoint cone alg. misses some stable cones; extra soft particle → extra starting point → extra stable cone found

MIDPOINT IS INFRARED UNSAFE

Or collinear unsafe with seed threshold

Does IRC safety really matter?

IRC safety & real-life

Real life does not have infinities, but pert. infinity leaves a real-life trace

$$\alpha_s^2 + \alpha_s^3 + \alpha_s^4 \times \infty \to \alpha_s^2 + \alpha_s^3 + \alpha_s^4 \times \ln p_t/\Lambda \to \alpha_s^2 + \underbrace{\alpha_s^3 + \alpha_s^3}_{\text{BOTH WASTED}}$$

Among consequences of IR unsafety:

	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 ($\to$ 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

NB: 50,000,000\$/£/CHF/€ investment in NLO

Multi-jet contexts much more sensitive: **ubiquitous at LHC**

And LHC will rely on QCD for background double-checks

extraction of cross sections, extraction of parameters

IRC safety & real-life

Real life does not have infinities, but pert. infinity leaves a real-life trace

$$\alpha_s^2 + \alpha_s^3 + \alpha_s^4 \times \infty \to \alpha_s^2 + \alpha_s^3 + \alpha_s^4 \times \ln p_t/\Lambda \to \alpha_s^2 + \underbrace{\alpha_s^3 + \alpha_s^3}_{\text{BOTH WASTED}}$$

Among consequences of IR unsafety:

	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 (→ 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

NB: 50,000,000$/£/CHF/€ investment in NLO

Multi-jet contexts much more sensitive: **ubiquitous at LHC**

And LHC will rely on QCD for background double-checks

extraction of cross sections, extraction of parameters

IRC safety & real-life

Real life does not have infinities, but pert. infinity leaves a real-life trace

$$\alpha_s^2 + \alpha_s^3 + \alpha_s^4 \times \infty \to \alpha_s^2 + \alpha_s^3 + \alpha_s^4 \times \ln p_t/\Lambda \to \alpha_s^2 + \underbrace{\alpha_s^3 + \alpha_s^3}_{\text{BOTH WASTED}}$$

Among consequences of IR unsafety:

	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 ($\to$ 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

NB: 50,000,000\$/£/CHF/€ investment in NLO

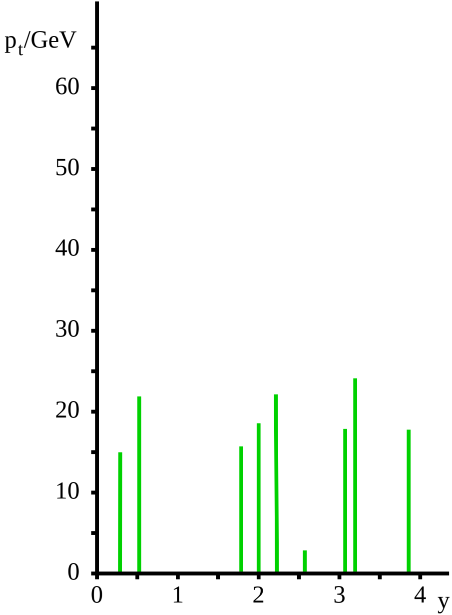
Multi-jet contexts much more sensitive: **ubiquitous at LHC**

And LHC will rely on QCD for background double-checks

extraction of cross sections, extraction of parameters

Can we cure this IR safety problem?

Seedless [Infrared Safe] cones (SC-SM)



Aim to identify *all* stable cones, independently of any seeds

Procedure in 1 dimension (y):

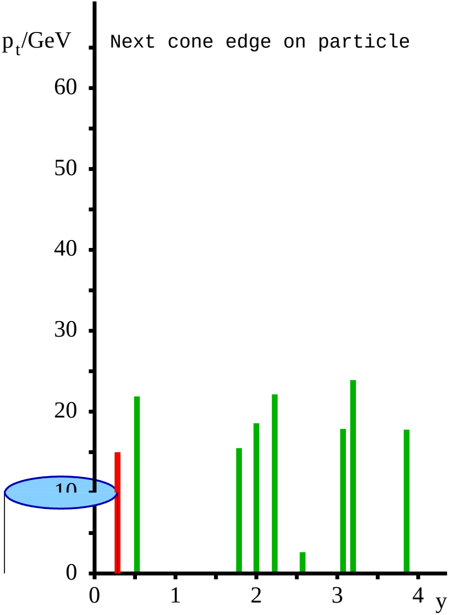
- find all distinct enclosures of radius R by repeatedly sliding a cone sideways until edge touches a particle
- check each for stability
- then run usual split–merge

In 2 dimensions (y, ϕ) can design analogous procedure SISCone

GPS & Soyez '07

This gives an IRC safe cone alg.

Seedless [Infrared Safe] cones (SC-SM)



Aim to identify *all* stable cones, independently of any seeds

Procedure in 1 dimension (y):

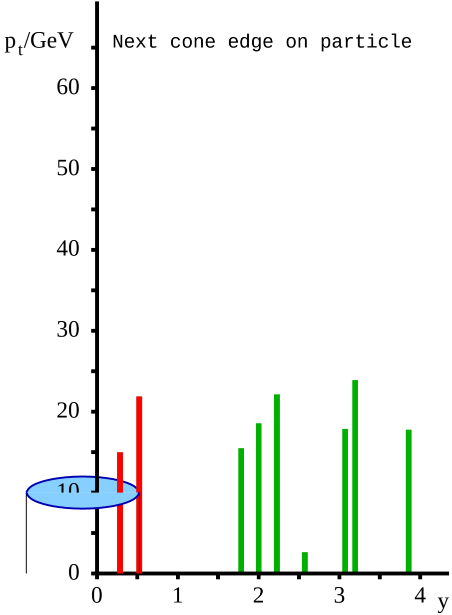
- find all distinct enclosures of radius R — by repeatedly sliding a cone sideways until edge touches a particle
- check each for stability
- then run usual split–merge

In 2 dimensions (y, ϕ) can design analogous procedure — SISCone

GPS & Soyez '07

This gives an IRC safe cone alg.

Seedless [Infrared Safe] cones (SC-SM)



Aim to identify *all* stable cones, independently of any seeds

Procedure in 1 dimension (y):

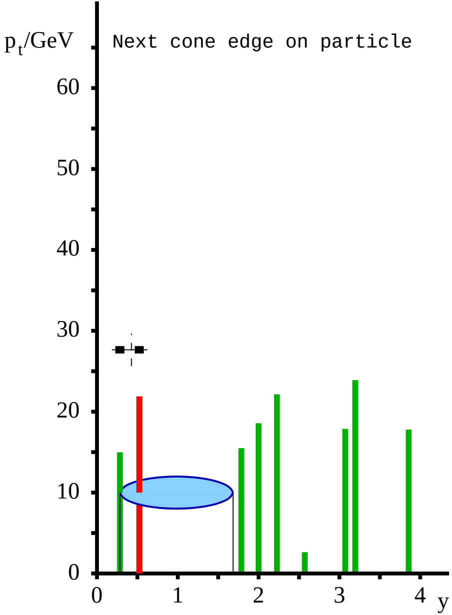
- find all distinct enclosures of radius R by repeatedly sliding a cone sideways until edge touches a particle
- check each for stability
- then run usual split–merge

In 2 dimensions (y,ϕ) can design analogous procedure SISCone

GPS & Soyez '07

This gives an IRC safe cone alg.

Seedless [Infrared Safe] cones (SC-SM)



Aim to identify *all* stable cones, independently of any seeds

Procedure in 1 dimension (y):

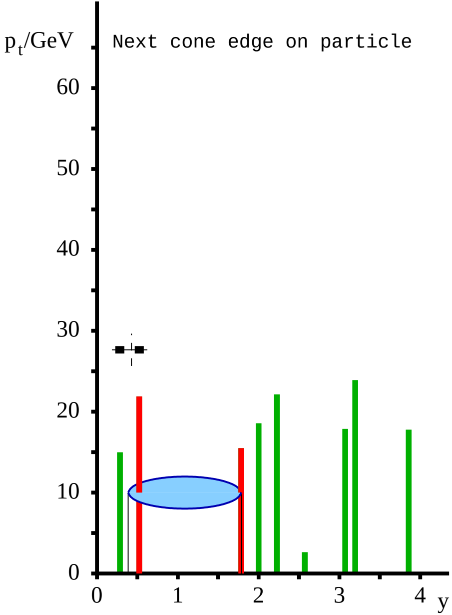
- find all distinct enclosures of radius R by repeatedly sliding a cone sideways until edge touches a particle
- check each for stability
- then run usual split–merge

In 2 dimensions (y, ϕ) *can design analogous procedure* SISCone

GPS & Soyez '07

This gives an IRC safe cone alg.

Seedless [Infrared Safe] cones (SC-SM)



Aim to identify *all* stable cones, independently of any seeds

Procedure in 1 dimension (y):

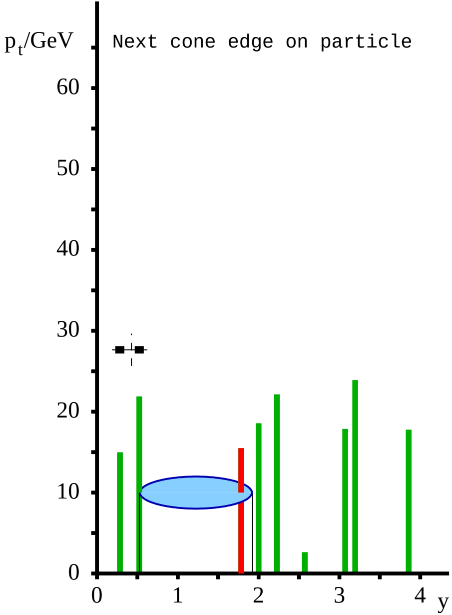
- find all distinct enclosures of radius R by repeatedly sliding a cone sideways until edge touches a particle
- check each for stability
- then run usual split–merge

In 2 dimensions (y, ϕ) *can design analogous procedure* SISCone

GPS & Soyez '07

This gives an IRC safe cone alg.

Seedless [Infrared Safe] cones (SC-SM)



Aim to identify *all* stable cones, independently of any seeds

Procedure in 1 dimension (y):

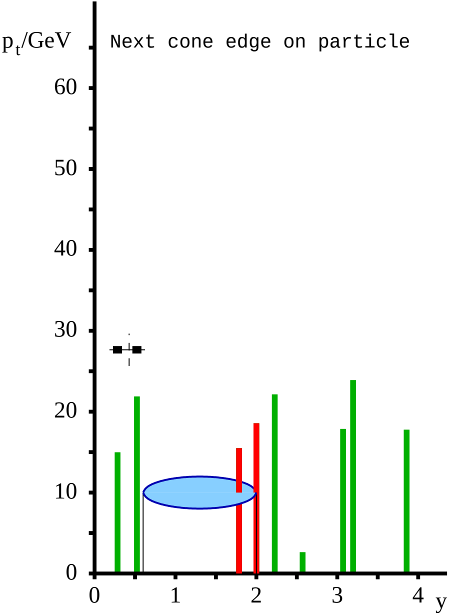
- find all distinct enclosures of radius R by repeatedly sliding a cone sideways until edge touches a particle
- check each for stability
- then run usual split–merge

In 2 dimensions (y, ϕ) can design analogous procedure SISCone

GPS & Soyez '07

This gives an IRC safe cone alg.

Seedless [Infrared Safe] cones (SC-SM)



Aim to identify *all* stable cones, independently of any seeds

Procedure in 1 dimension (y):

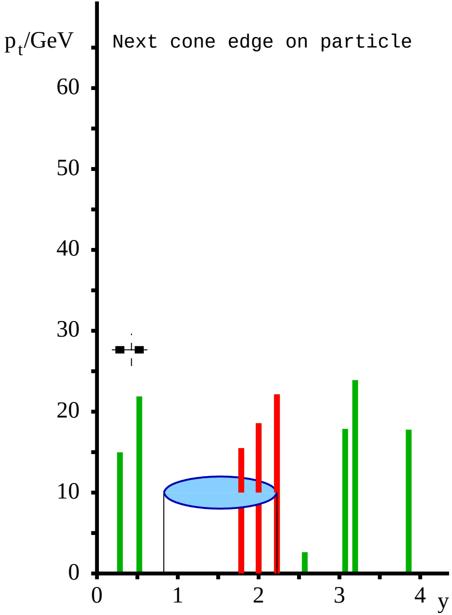
- find all distinct enclosures of radius R — by repeatedly sliding a cone sideways until edge touches a particle
- check each for stability
- then run usual split–merge

In 2 dimensions (y, ϕ) can design analogous procedure — SISCone

GPS & Soyez '07

This gives an IRC safe cone alg.

Seedless [Infrared Safe] cones (SC-SM)



Aim to identify *all* stable cones, independently of any seeds

Procedure in 1 dimension (y):

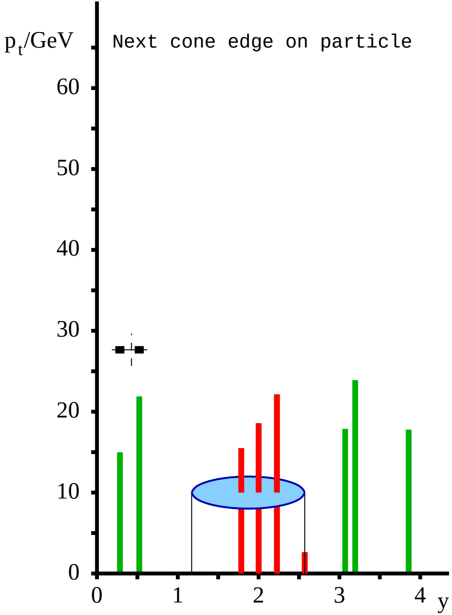
- find all distinct enclosures of radius R — by repeatedly sliding a cone sideways until edge touches a particle
- check each for stability
- then run usual split–merge

In 2 dimensions (y,ϕ) can design analogous procedure — SISCone

GPS & Soyez '07

This gives an IRC safe cone alg.

Seedless [Infrared Safe] cones (SC-SM)



Aim to identify *all* stable cones, independently of any seeds

Procedure in 1 dimension (y):

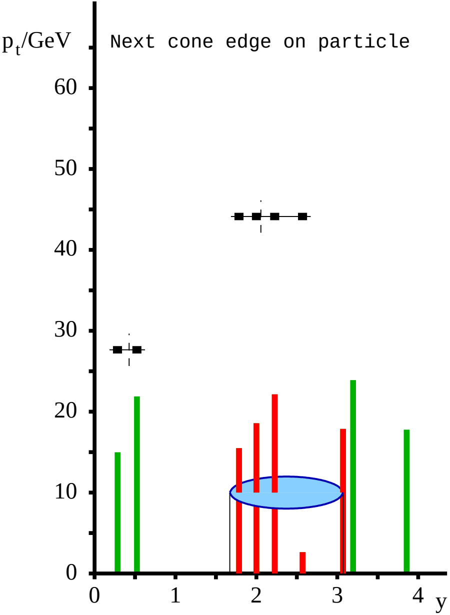
- find all distinct enclosures of radius R — by repeatedly sliding a cone sideways until edge touches a particle
- check each for stability
- then run usual split–merge

In 2 dimensions (y,ϕ) can design analogous procedure — SISCone

GPS & Soyez '07

This gives an IRC safe cone alg.

Seedless [Infrared Safe] cones (SC-SM)



Aim to identify *all* stable cones, independently of any seeds

Procedure in 1 dimension (y):

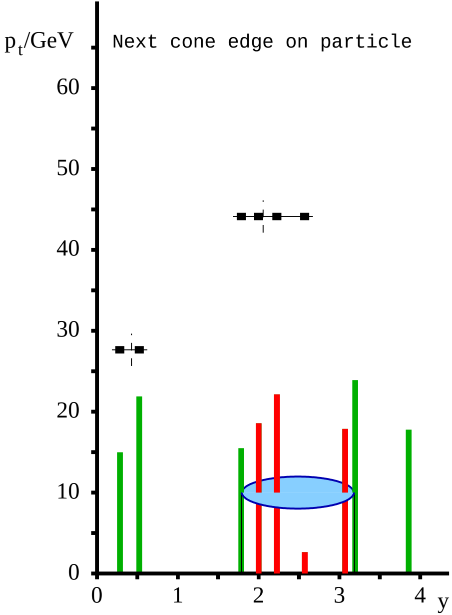
- find all distinct enclosures of radius R — by repeatedly sliding a cone sideways until edge touches a particle
- check each for stability
- then run usual split–merge

In 2 dimensions (y, ϕ) can design analogous procedure SISCone

GPS & Soyez '07

This gives an IRC safe cone alg.

Seedless [Infrared Safe] cones (SC-SM)



Aim to identify *all* stable cones, independently of any seeds

Procedure in 1 dimension (y):

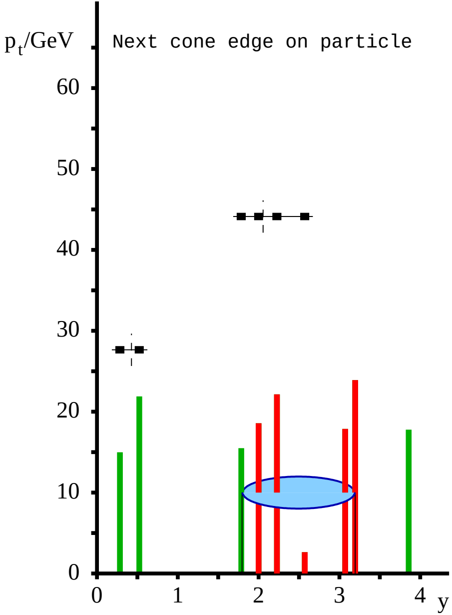
- find all distinct enclosures of radius R by repeatedly sliding a cone sideways until edge touches a particle
- check each for stability
- then run usual split–merge

In 2 dimensions (y,ϕ) can design analogous procedure SISCone

GPS & Soyez '07

This gives an IRC safe cone alg.

Seedless [Infrared Safe] cones (SC-SM)



Aim to identify *all* stable cones, independently of any seeds

Procedure in 1 dimension (y):

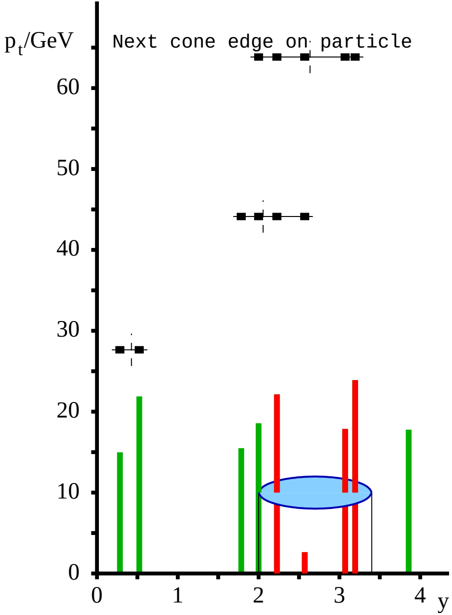
- find all distinct enclosures of radius R — by repeatedly sliding a cone sideways until edge touches a particle
- check each for stability
- then run usual split–merge

In 2 dimensions (y, ϕ) can design analogous procedure — SISCone

GPS & Soyez '07

This gives an IRC safe cone alg.

Seedless [Infrared Safe] cones (SC-SM)



Aim to identify *all* stable cones, independently of any seeds

Procedure in 1 dimension (y):

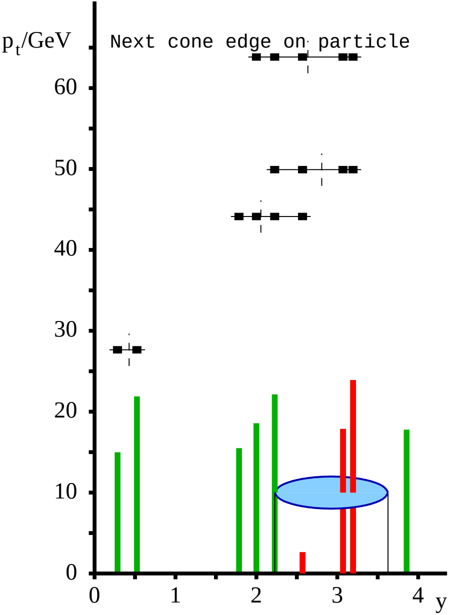
- find all distinct enclosures of radius R by repeatedly sliding a cone sideways until edge touches a particle
- check each for stability
- then run usual split–merge

In 2 dimensions (y,ϕ) can design analogous procedure SISCone

GPS & Soyez '07

This gives an IRC safe cone alg.

Seedless [Infrared Safe] cones (SC-SM)



Aim to identify *all* stable cones, independently of any seeds

Procedure in 1 dimension (y):

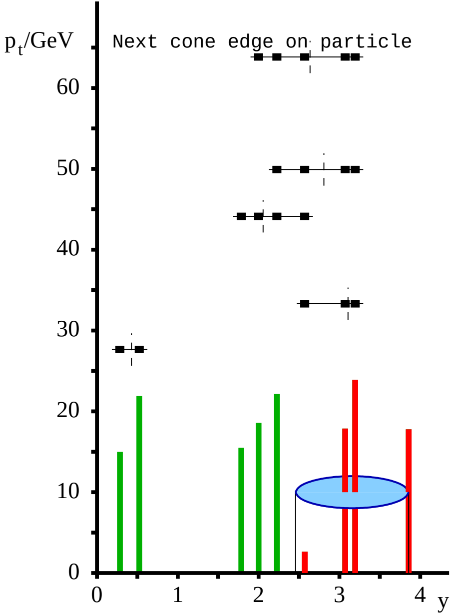
- find all distinct enclosures of radius R by repeatedly sliding a cone sideways until edge touches a particle
- check each for stability
- then run usual split–merge

In 2 dimensions (y,ϕ) can design analogous procedure SISCone

GPS & Soyez '07

This gives an IRC safe cone alg.

Seedless [Infrared Safe] cones (SC-SM)



Aim to identify *all* stable cones, independently of any seeds

Procedure in 1 dimension (y):

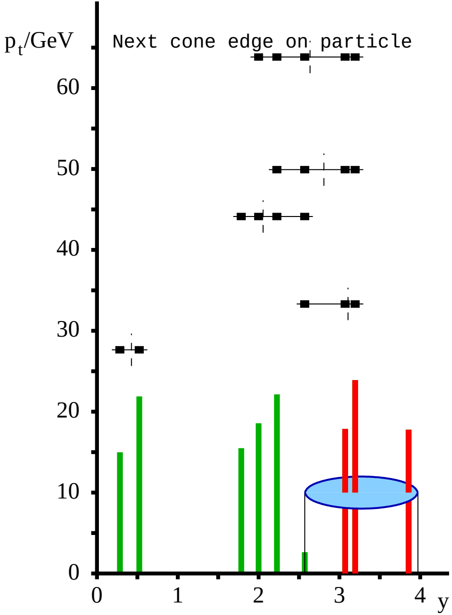
- find all distinct enclosures of radius R by repeatedly sliding a cone sideways until edge touches a particle
- check each for stability
- then run usual split–merge

In 2 dimensions (y,ϕ) can design analogous procedure SISCone

GPS & Soyez '07

This gives an IRC safe cone alg.

Seedless [Infrared Safe] cones (SC-SM)



Aim to identify *all* stable cones, independently of any seeds

Procedure in 1 dimension (y):

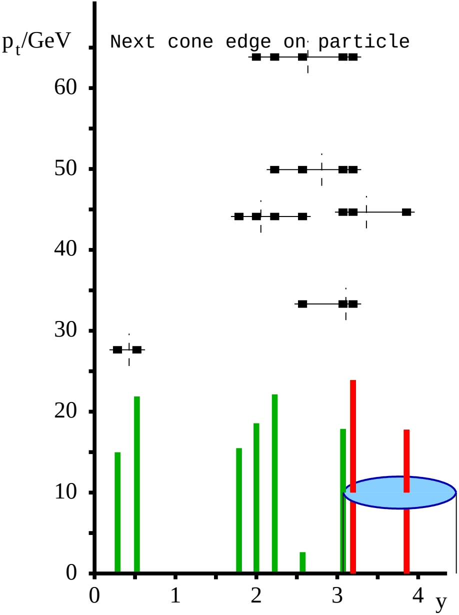
- find all distinct enclosures of radius R by repeatedly sliding a cone sideways until edge touches a particle
- check each for stability
- then run usual split–merge

In 2 dimensions (y,ϕ) can design analogous procedure SISCone

GPS & Soyez '07

This gives an IRC safe cone alg.

Seedless [Infrared Safe] cones (SC-SM)



Aim to identify *all* stable cones, independently of any seeds

Procedure in 1 dimension (y):

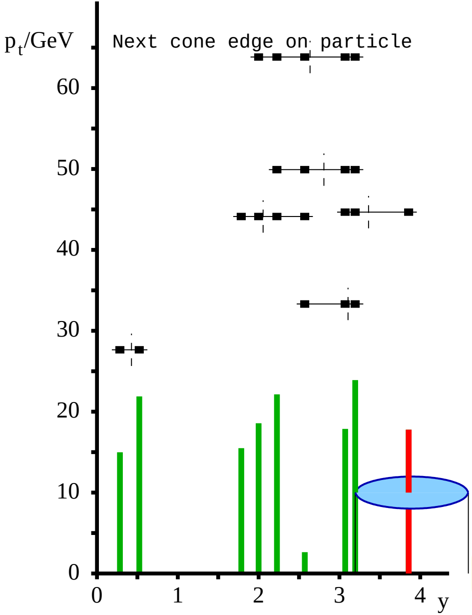
- find all distinct enclosures of radius R — by repeatedly sliding a cone sideways until edge touches a particle
- check each for stability
- then run usual split–merge

In 2 dimensions (y,ϕ) can design analogous procedure **SISCone**

GPS & Soyez '07

This gives an IRC safe cone alg.

Seedless [Infrared Safe] cones (SC-SM)



Aim to identify *all* stable cones, independently of any seeds

Procedure in 1 dimension (y):

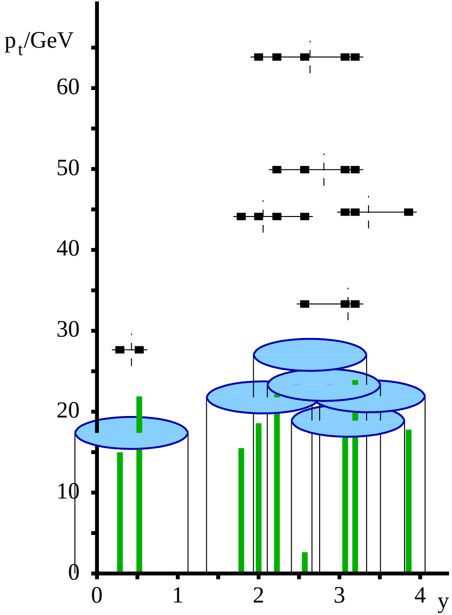
- find all distinct enclosures of radius R by repeatedly sliding a cone sideways until edge touches a particle
- check each for stability
- then run usual split–merge

In 2 dimensions (y,ϕ) can design analogous procedure **SISCone**

GPS & Soyez '07

This gives an IRC safe cone alg.

Seedless [Infrared Safe] cones (SC-SM)



Aim to identify *all* stable cones, independently of any seeds

Procedure in 1 dimension (y):

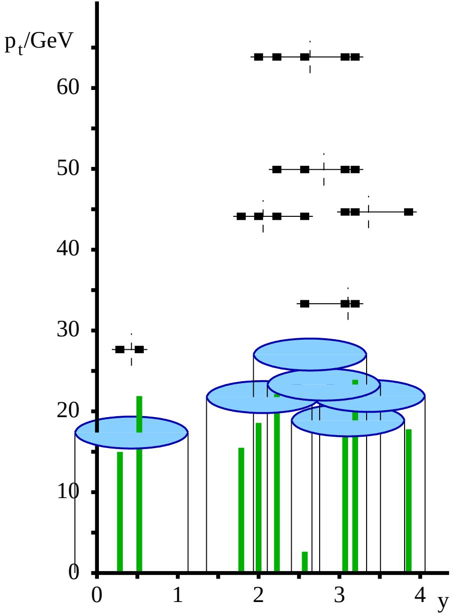
- find all distinct enclosures of radius R by repeatedly sliding a cone sideways until edge touches a particle
- check each for stability
- then run usual split–merge

In 2 dimensions (y,ϕ) can design analogous procedure **SISCone**

GPS & Soyez '07

This gives an IRC safe cone alg.

Seedless [Infrared Safe] cones (SC-SM)



Aim to identify *all* stable cones, independently of any seeds

Procedure in 1 dimension (y):

- find all distinct enclosures of radius R — by repeatedly sliding a cone sideways until edge touches a particle
- check each for stability
- then run usual split–merge

In 2 dimensions (y,ϕ) can design analogous procedure **SISCone**
GPS & Soyez '07

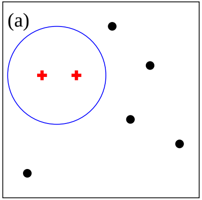
This gives an IRC safe cone alg.

Solve IR issue: find all stable cones

Cones are just *circles* in the $y - \phi$ plane. To find all stable cones:

1. Find all distinct ways of enclosing a subset of particles in a $y - \phi$ circle
2. Check, for each enclosure, if it corresponds to a stable cone

Finding all distinct circular enclosures of a set of points is **geometry:**



Any enclosure can be moved until a pair of points lies on its edge.

Result: Seedless Infrared Safe Cone algorithm (SISCone)

Runs in $N^2 \ln N$ time ($\simeq$ midpoint's N^3)

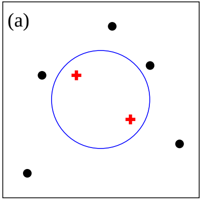
Fast large-N code: GPS & Soyez '07, low-N code: Weinzierl '11

Solve IR issue: find all stable cones

Cones are just *circles* in the $y - \phi$ plane. To find all stable cones:

1. Find all distinct ways of enclosing a subset of particles in a $y - \phi$ circle
2. Check, for each enclosure, if it corresponds to a stable cone

Finding all distinct circular enclosures of a set of points is **geometry:**



Any enclosure can be moved until a pair of points lies on its edge.

Result: Seedless Infrared Safe Cone algorithm (SISCone)

Runs in $N^2 \ln N$ time ($\simeq$ midpoint's N^3)

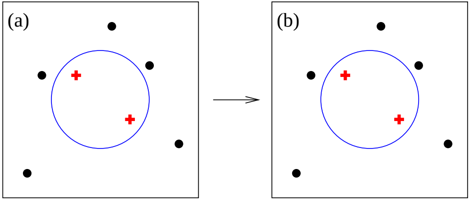
Fast large-N code: GPS & Soyez '07, low-N code: Weinzierl '11

Solve IR issue: find all stable cones

Cones are just *circles* in the $y - \phi$ plane. To find all stable cones:

1. Find all distinct ways of enclosing a subset of particles in a $y - \phi$ circle
2. Check, for each enclosure, if it corresponds to a stable cone

Finding all distinct circular enclosures of a set of points is **geometry:**



Any enclosure can be moved until a pair of points lies on its edge.

Result: Seedless Infrared Safe Cone algorithm (SISCone)

Runs in $N^2 \ln N$ time ($\simeq$ midpoint's N^3)

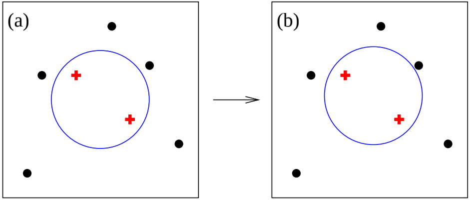
Fast large-N code: GPS & Soyez '07, low-N code: Weinzierl '11

Solve IR issue: find all stable cones

Cones are just *circles* in the $y - \phi$ plane. To find all stable cones:

1. Find all distinct ways of enclosing a subset of particles in a $y - \phi$ circle
2. Check, for each enclosure, if it corresponds to a stable cone

Finding all distinct circular enclosures of a set of points is **geometry:**



Any enclosure can be moved until a pair of points lies on its edge.

Result: Seedless Infrared Safe Cone algorithm (SISCone)

Runs in $N^2 \ln N$ time ($\simeq$ midpoint's N^3)

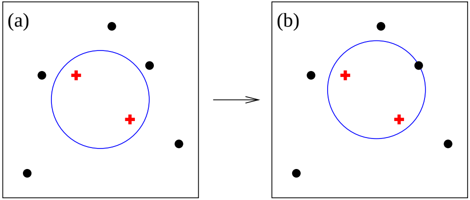
Fast large-N code: GPS & Soyez '07, low-N code: Weinzierl '11

Solve IR issue: find all stable cones

Cones are just *circles* in the $y - \phi$ plane. To find all stable cones:

1. Find all distinct ways of enclosing a subset of particles in a $y - \phi$ circle
2. Check, for each enclosure, if it corresponds to a stable cone

Finding all distinct circular enclosures of a set of points is **geometry:**



Any enclosure can be moved until a pair of points lies on its edge.

Result: Seedless Infrared Safe Cone algorithm (SISCone)

Runs in $N^2 \ln N$ time ($\simeq$ midpoint's N^3)

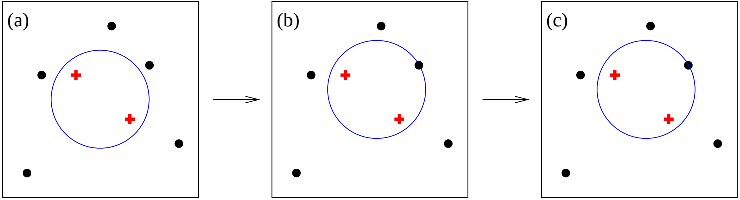
Fast large-N code: GPS & Soyez '07, low-N code: Weinzierl '11

Solve IR issue: find all stable cones

Cones are just *circles* in the $y - \phi$ plane. To find all stable cones:

1. Find all distinct ways of enclosing a subset of particles in a $y - \phi$ circle
2. Check, for each enclosure, if it corresponds to a stable cone

Finding all distinct circular enclosures of a set of points is **geometry:**



Any enclosure can be moved until a pair of points lies on its edge.

Result: Seedless Infrared Safe Cone algorithm (SISCone)

Runs in $N^2 \ln N$ time ($\simeq$ midpoint's N^3)

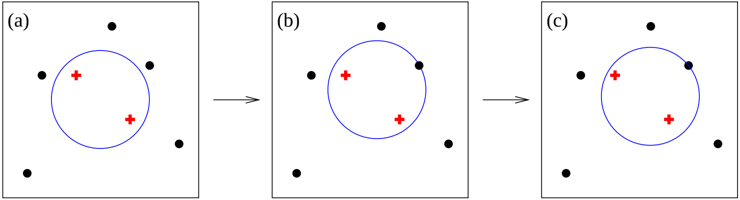
Fast large-N code: GPS & Soyez '07, low-N code: Weinzierl '11

Solve IR issue: find all stable cones

Cones are just *circles* in the $y - \phi$ plane. To find all stable cones:

1. Find all distinct ways of enclosing a subset of particles in a $y - \phi$ circle
2. Check, for each enclosure, if it corresponds to a stable cone

Finding all distinct circular enclosures of a set of points is **geometry:**



Any enclosure can be moved until a pair of points lies on its edge.

Result: Seedless Infrared Safe Cone algorithm (SISCone)

Runs in $N^2 \ln N$ time ($\simeq$ midpoint's N^3)

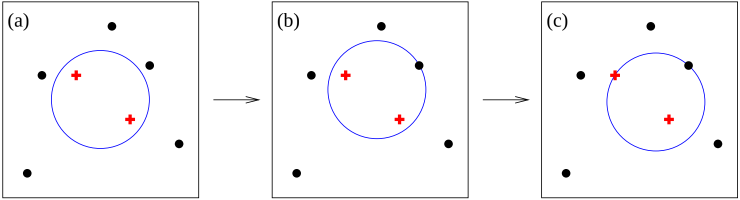
Fast large-N code: GPS & Soyez '07; low-N code: Weinzierl '11

Solve IR issue: find all stable cones

Cones are just *circles* in the $y - \phi$ plane. To find all stable cones:

1. Find all distinct ways of enclosing a subset of particles in a $y - \phi$ circle
2. Check, for each enclosure, if it corresponds to a stable cone

Finding all distinct circular enclosures of a set of points is **geometry:**



Any enclosure can be moved until a pair of points lies on its edge.

Result: Seedless Infrared Safe Cone algorithm (SISCone)

Runs in $N^2 \ln N$ time ($\simeq$ midpoint's N^3)

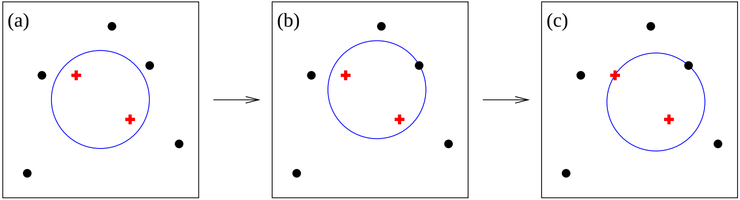
Fast large-N code: GPS & Soyez '07; low-N code: Weinzierl '11

Solve IR issue: find all stable cones

Cones are just *circles* in the $y - \phi$ plane. To find all stable cones:

1. Find all distinct ways of enclosing a subset of particles in a $y - \phi$ circle
2. Check, for each enclosure, if it corresponds to a stable cone

Finding all distinct circular enclosures of a set of points is **geometry:**



Any enclosure can be moved until a pair of points lies on its edge.

Result: Seedless Infrared Safe Cone algorithm (SISCone)

Runs in $N^2 \ln N$ time ($\simeq$ midpoint's N^3)

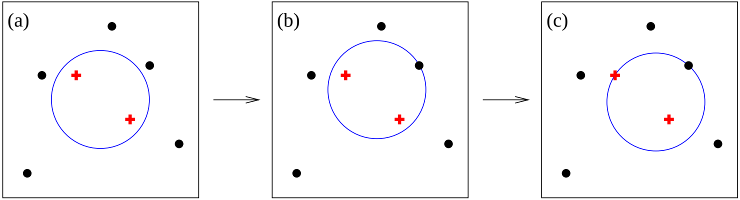
Fast large-N code: GPS & Soyez '07; low-N code: Weinzierl '11

Solve IR issue: find all stable cones

Cones are just *circles* in the $y - \phi$ plane. To find all stable cones:

1. Find all distinct ways of enclosing a subset of particles in a $y - \phi$ circle
2. Check, for each enclosure, if it corresponds to a stable cone

Finding all distinct circular enclosures of a set of points is **geometry:**



Any enclosure can be moved until a pair of points lies on its edge.

Result: Seedless Infrared Safe Cone algorithm (SISCone)

Runs in $N^2 \ln N$ time ($\simeq$ midpoint's N^3)

Fast large-N code: GPS & Soyez '07; low-N code: Weinzierl '11

Is it truly IR safe?

- 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.

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

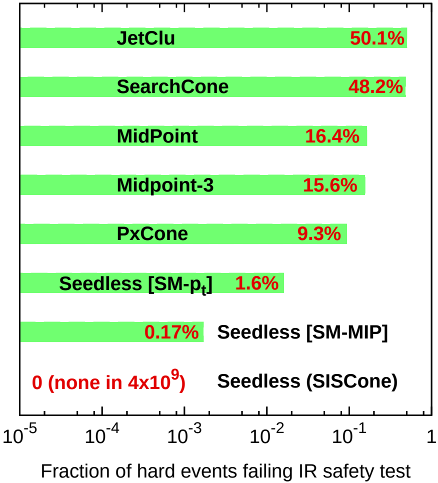
Be careful with split–merge too

Is it truly IR safe?

- 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.

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



A full set of IRC-safe jet algorithms

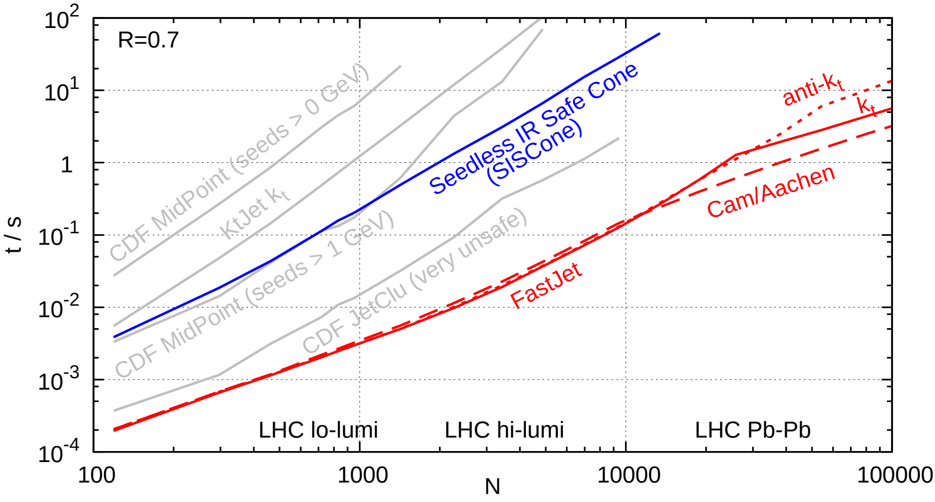
Generalise inclusive-type sequential recombination with

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

	Alg. name	Comment	time
$p = 1$	k_t CDOSTW '91-93; ES '93	Hierarchical in rel. k_t	$N \ln N$ exp.
$p = 0$	Cambridge/Aachen Dok, Leder, Moretti, Webber '97 Wengler, Wobisch '98	Hierarchical in angle Scan multiple R at once $\leftrightarrow$ QCD angular ordering	$N \ln N$
$p = -1$	anti-k_t Cacciari, GPS, Soyez '08 $\sim$ reverse-k_t Delsart	Hierarchy meaningless, jets like CMS cone (IC-PR)	$N^{3/2}$
SC-SM	SISCone GPS Soyez '07 + Tevatron run II '00	Replaces JetClu, ATLAS MidPoint (xC-SM) cones	$N^2 \ln N$ exp.

All these algorithms [& much more] coded in (efficient) C++ at
`http://fastjet.fr/` (Cacciari, GPS & Soyez '05-'11)

FastJet: time to cluster N particles



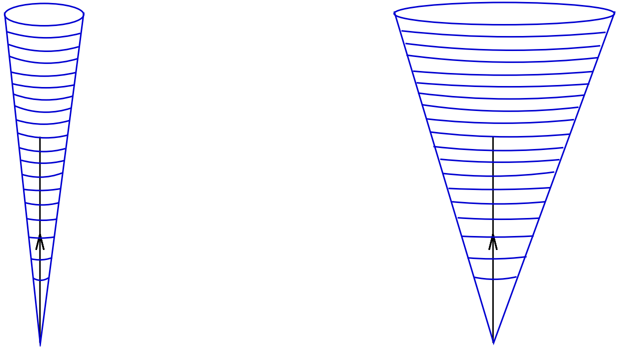
Towards an understanding of jets

How a jet is and isn't like a parton — quantitatively

And how this relationship is affected by the jet radius

Small v. large jet radius (R) $\equiv$ HSBC

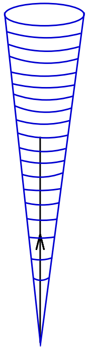
Small jet radius **Large jet radius**



single parton @ LO: **jet radius irrelevant**

Small v. large jet radius (R) ≡ HSBC

Small jet radius

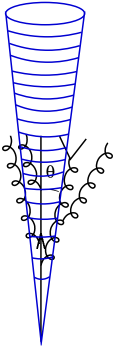


single par

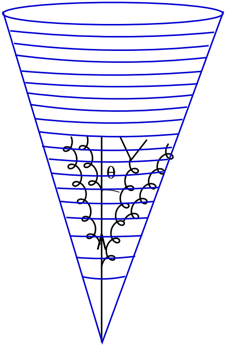
Large jet radius

Small v. large jet radius (R) ≡ HSBC

Small jet radius



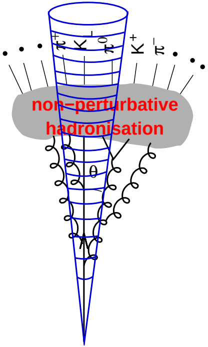
Large jet radius



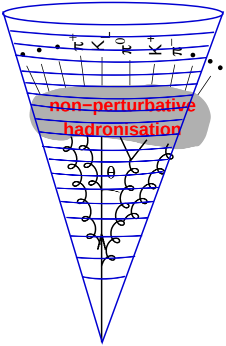
perturbative fragmentation: **large jet radius better**
(it captures more)

Small v. large jet radius (R) $\equiv$ HSBC

Small jet radius



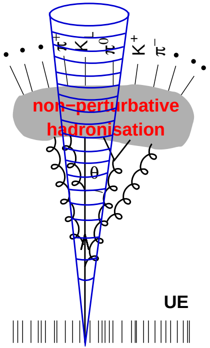
Large jet radius



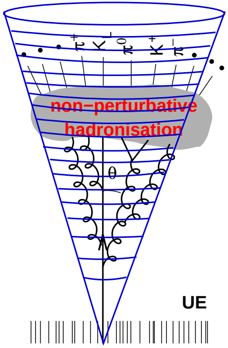
non-perturbative fragmentation: **large jet radius better**
(it captures more)

Small v. large jet radius (R) $\equiv$ HSBC

Small jet radius



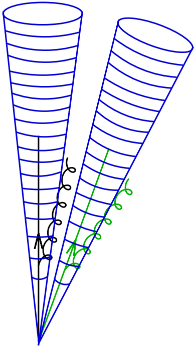
Large jet radius



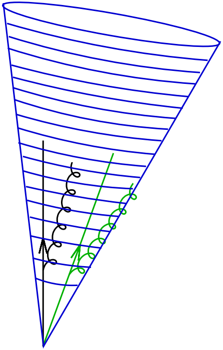
underlying ev. & pileup “noise”: **small jet radius better**
(it captures less)

Small v. large jet radius (R) $\equiv$ HSBC

Small jet radius



Large jet radius



multi-hard-parton events: **small jet radius better**

(it resolves partons more effectively)

Parton p_t v. jet p_t

3 physical effects:

1. Gluon radiation from the parton
2. Hadronisation
3. Underlying Event

One important consideration:

Whether the parton is a quark or a gluon

[quarks radiate with colour factor $C_F = 4/3$
gluons radiate with colour factor $C_A = 3$]

Jet p_t v. parton p_t: perturbatively?

The question's dangerous: a "parton" is an ambiguous concept

Three limits can help you:

- Threshold limit — e.g. de Florian & Vogelsang '07
- Parton from color-neutral object decay (Z')
- Small-R (radius) limit for jet

One simple result (small-R limit)

$$\frac{\langle p_{t,jet} - p_{t,parton} \rangle}{p_t} = \frac{\alpha_s}{\pi} \ln R \times \begin{cases} 1.01 C_F & \textit{quarks} \\ 0.94 C_A + 0.07 n_f & \textit{gluons} \end{cases} + \mathcal{O}(\alpha_s)$$

only $\mathcal{O}(\alpha_s)$ depends on algorithm & process
cf. Dasgupta, Magnea & GPS '07

Jet p_t v. parton p_t: hadronisation?

Hadronisation: the "parton-shower" $\to$ hadrons transition

Method:

- "infrared finite α_s" — à la Dokshitzer & Webber '95
- **prediction** based on e^+e^- event shape data
- could have been deduced from old work — Korchemsky & Sterman '95, Seymour '97

Main result

$$\langle p_{t,jet} - p_{t,parton-shower} \rangle \simeq -\frac{0.4\ \text{GeV}}{R} \times \begin{cases} C_F & quarks \\ C_A & gluons \end{cases}$$

cf. Dasgupta, Magnea & GPS '07

coefficient holds for anti-k_t; see Dasgupta & Delenda '09 for k_t alg.

Underlying Event (UE)

"Naive" prediction (UE $\simeq$ colour dipole between pp):

$$\Delta p_t \simeq 0.4 \text{ GeV} \times \frac{R^2}{2} \times \begin{cases} C_F & q\bar{q} \text{ dipole} \\ C_A & \text{gluon dipole} \end{cases}$$

Modern Monte Carlo tunes tell you ($\sqrt{s} = 7$ TeV):

$$\Delta p_t \simeq \mathbf{8\ GeV} \times \frac{R^2}{2} \simeq 1.2 \text{ GeV} \times (\pi R^2)$$

This big coefficient motivates special effort to understand interplay between jet algorithm and UE: "jet areas"

How does coefficient depend on algorithm?

How does it depend on jet p_t? How does it fluctuate?

cf. Cacciari, GPS & Soyez '08

Underlying Event (UE)

"Naive" prediction (UE $\simeq$ colour dipole between pp):

$$\Delta p_t \simeq 0.4 \text{ GeV} \times \frac{R^2}{2} \times \begin{cases} C_F & q\bar{q} \text{ dipole} \\ C_A & \text{gluon dipole} \end{cases}$$

Modern Monte Carlo tunes tell you ($\sqrt{s} = 7$ TeV):

$$\Delta p_t \simeq \mathbf{8\ GeV} \times \frac{R^2}{2} \simeq 1.2 \text{ GeV} \times (\pi R^2)$$

This big coefficient motivates special effort to understand interplay between jet algorithm and UE: "jet areas"

How does coefficient depend on algorithm?

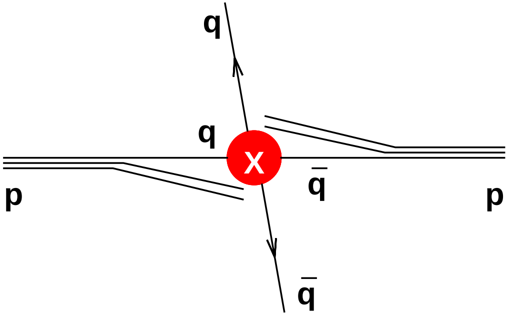
How does it depend on jet p_t? How does it fluctuate?

cf. Cacciari, GPS & Soyez '08

Using our understanding to help discover a dijet resonance, $q\bar{q} \to X \to q\bar{q}$.

What R is best for an isolated jet?

E.g. to reconstruct $m_X \sim (p_{tq} + p_{t\bar{q}})$



PT radiation:

$$q: \quad \langle \Delta p_t \rangle \simeq \frac{\alpha_s C_F}{\pi} p_t \ln R$$

Hadronisation:

$$q: \quad \langle \Delta p_t \rangle \simeq -\frac{C_F}{R} \cdot 0.4 \text{ GeV}$$

Underlying event:

$$q, g: \quad \langle \Delta p_t \rangle \simeq \frac{R^2}{2} \cdot 2.5\text{—}15 \text{ GeV}$$

Minimise fluctuations in p_t

Use crude approximation:

$$\langle \Delta p_t^2 \rangle \simeq \langle \Delta p_t \rangle^2$$

in small-R limit (!)

NB: full calc, correct fluct: Soyez '10

What R is best for an isolated jet?

PT radiation:

$$q: \quad \langle \Delta p_t \rangle \simeq \frac{\alpha_s C_F}{\pi} p_t \ln R$$

Hadronisation:

$$q: \quad \langle \Delta p_t \rangle \simeq -\frac{C_F}{R} \cdot 0.4 \text{ GeV}$$

Underlying event:

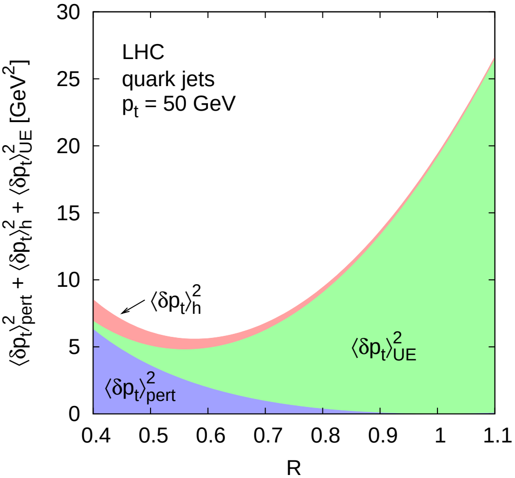
$$q, g: \quad \langle \Delta p_t \rangle \simeq \frac{R^2}{2} \cdot 2.5 - 15 \text{ GeV}$$

Minimise fluctuations in p_t

Use crude approximation:

$$\langle \Delta p_t^2 \rangle \simeq \langle \Delta p_t \rangle^2$$

50 GeV quark jet



in small-R limit (!)

NB: full calc, correct fluct: Soyez '10

What R is best for an isolated jet?

PT radiation:

$$q: \quad \langle \Delta p_t \rangle \simeq \frac{\alpha_s C_F}{\pi} p_t \ln R$$

Hadronisation:

$$q: \quad \langle \Delta p_t \rangle \simeq -\frac{C_F}{R} \cdot 0.4 \text{ GeV}$$

Underlying event:

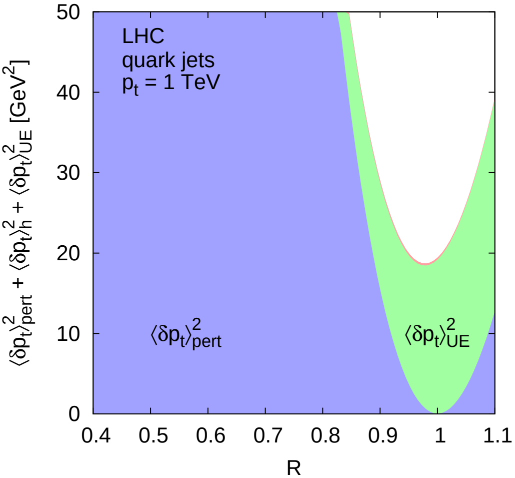
$$q, g: \quad \langle \Delta p_t \rangle \simeq \frac{R^2}{2} \cdot 2.5\text{—}15 \text{ GeV}$$

Minimise fluctuations in p_t

Use crude approximation:

$$\langle \Delta p_t^2 \rangle \simeq \langle \Delta p_t \rangle^2$$

1 TeV quark jet



in small-R limit (!)

NB: full calc, correct fluct: Soyez '10

What R is best for an isolated jet?

PT radiation:

$$q: \quad \langle \Delta p_t \rangle \simeq \frac{\alpha_s C_F}{\pi} p_t \ln R$$

Hadr

$q:$

At low p_t, small R limits relative impact of UE

At high p_t, perturbative effects dominate over non-perturbative $\rightarrow R_{best} \sim 1$.

Underlying event:

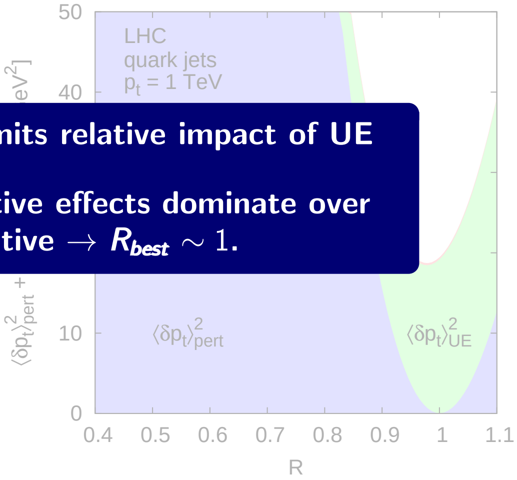
$$q, g: \quad \langle \Delta p_t \rangle \simeq \frac{R^2}{2} \cdot 2.5\text{—}15 \text{ GeV}$$

Minimise fluctuations in p_t

Use crude approximation:

$$\langle \Delta p_t^2 \rangle \simeq \langle \Delta p_t \rangle^2$$

1 TeV quark jet

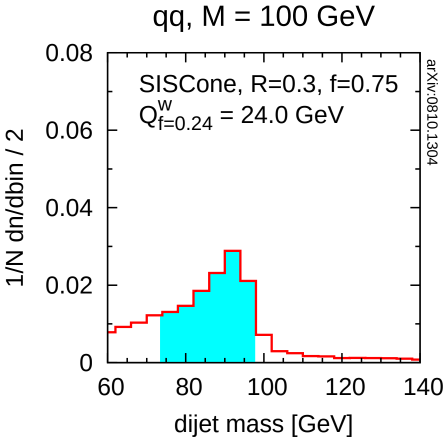


in small-R limit (!)

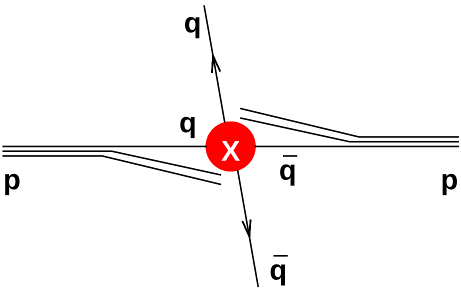
NB: full calc, correct fluct: Soyez '10

Dijet mass: scan over R [Pythia 6.4]

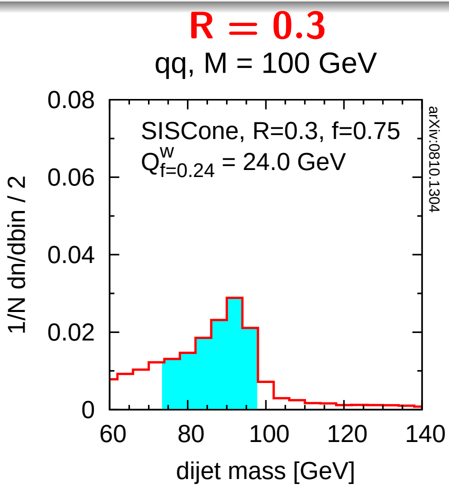
R = 0.3

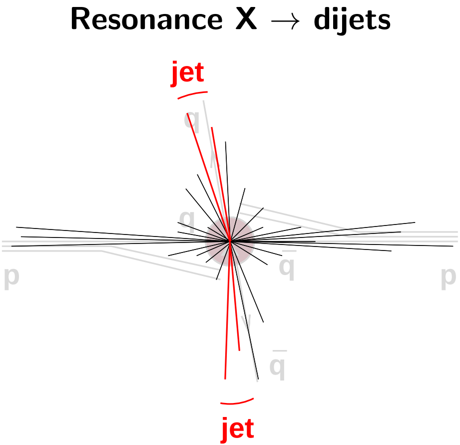


Resonance X → dijets

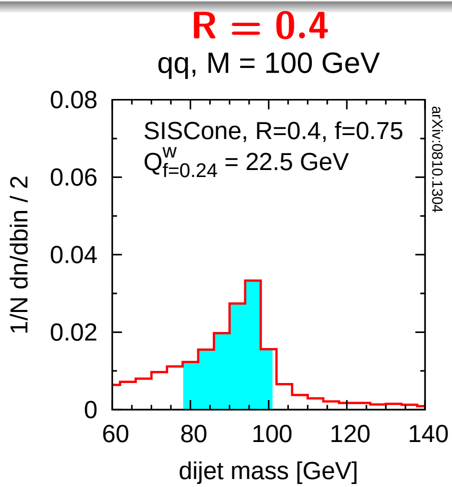


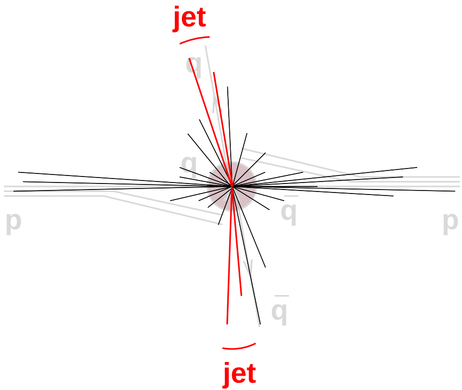
Dijet mass: scan over R [Pythia 6.4]





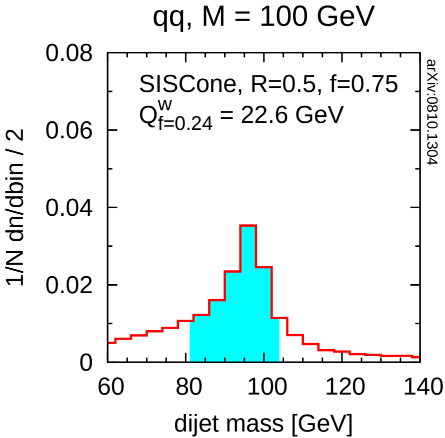
Dijet mass: scan over R [Pythia 6.4]



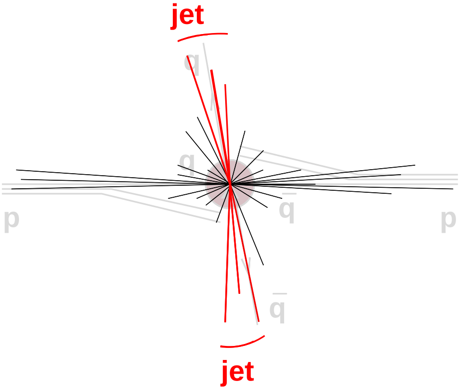


Dijet mass: scan over *R* [Pythia 6.4]

R = 0.5



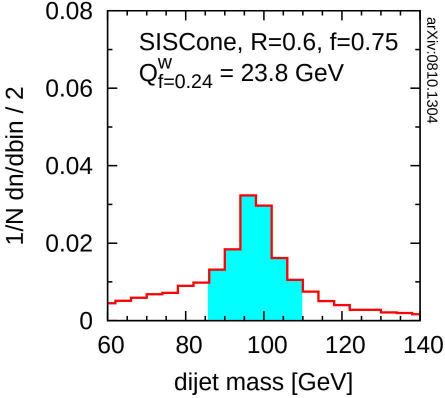
Resonance X → dijets



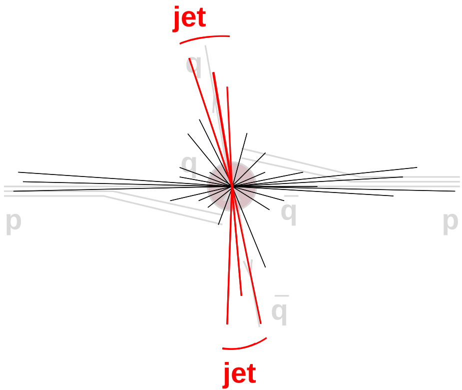
Dijet mass: scan over *R* [Pythia 6.4]

R = 0.6

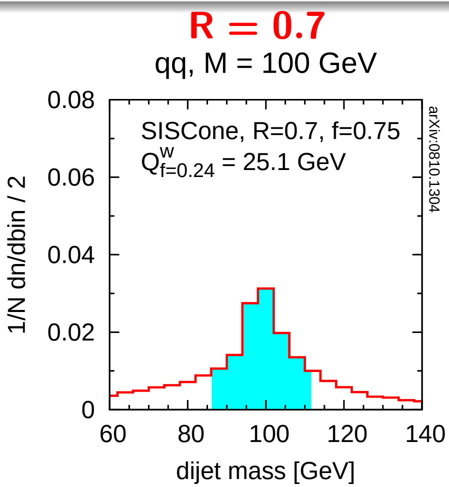
qq, M = 100 GeV

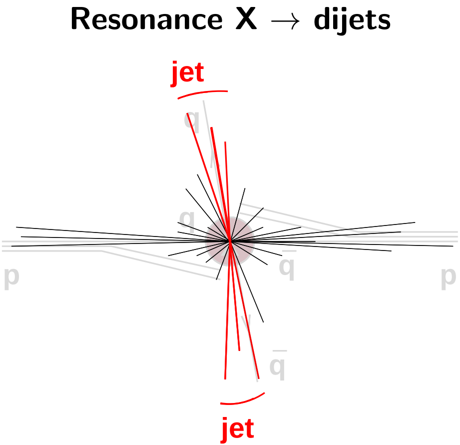


Resonance X → dijets



Dijet mass: scan over R [Pythia 6.4]

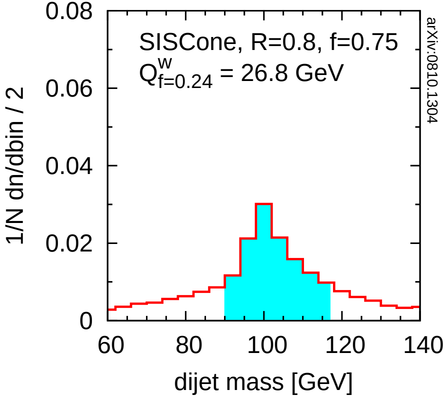




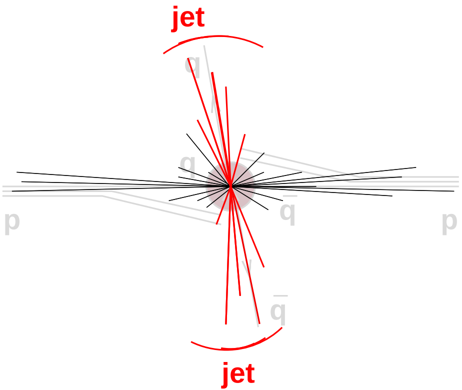
Dijet mass: scan over R [Pythia 6.4]

$R = 0.8$

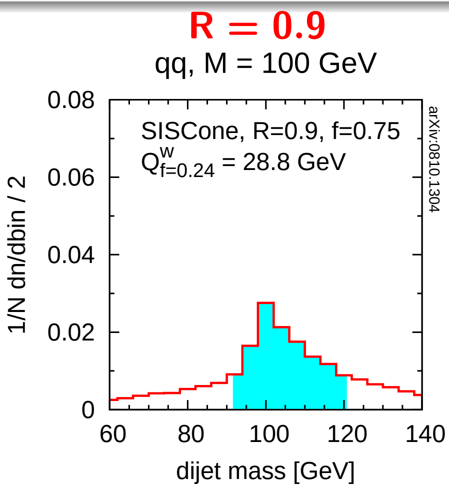
qq, M = 100 GeV



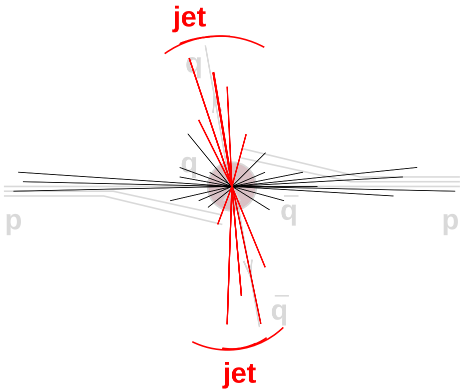
Resonance X → dijets



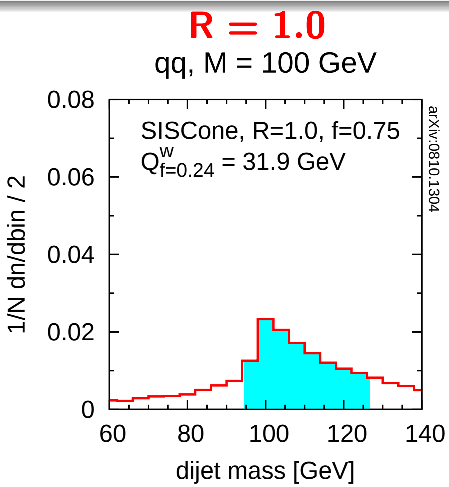
Dijet mass: scan over R [Pythia 6.4]

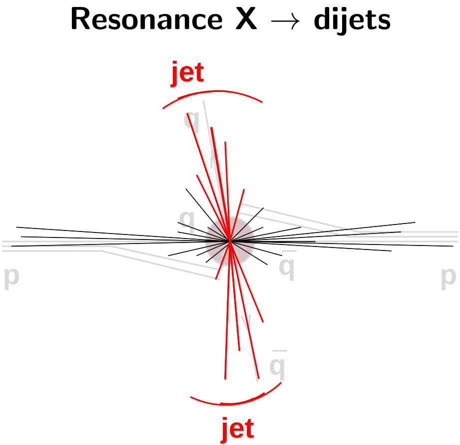


Resonance X → dijets

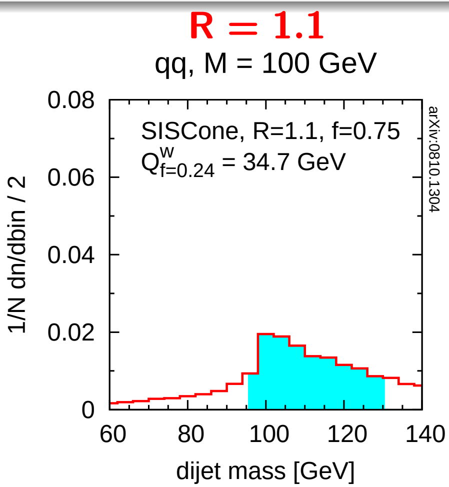


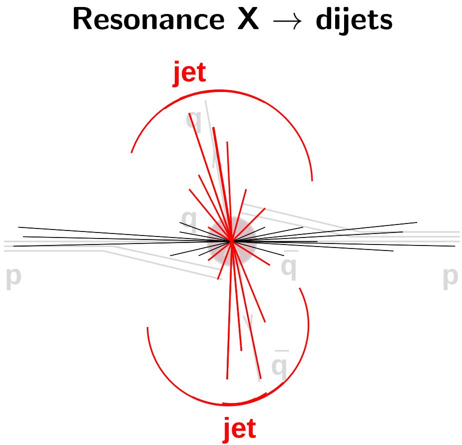
Dijet mass: scan over *R* [Pythia 6.4]



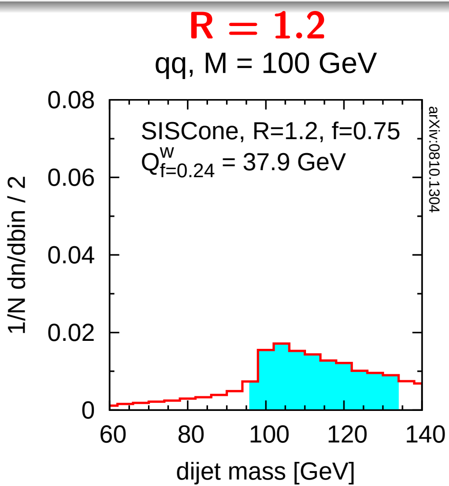


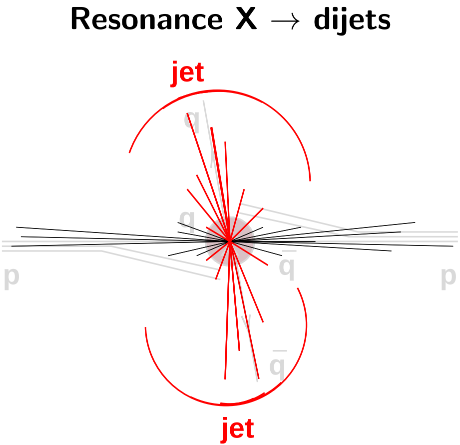
Dijet mass: scan over R [Pythia 6.4]





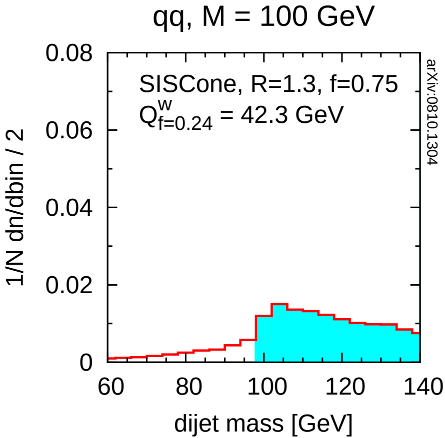
Dijet mass: scan over *R* [Pythia 6.4]



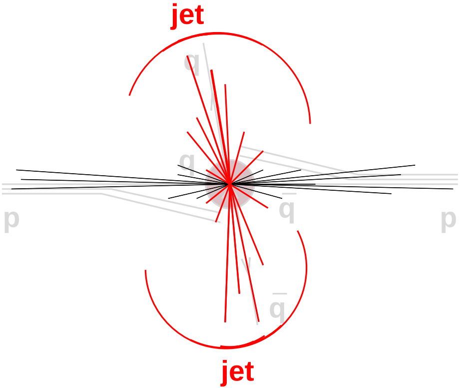


Dijet mass: scan over R [Pythia 6.4]

R = 1.3

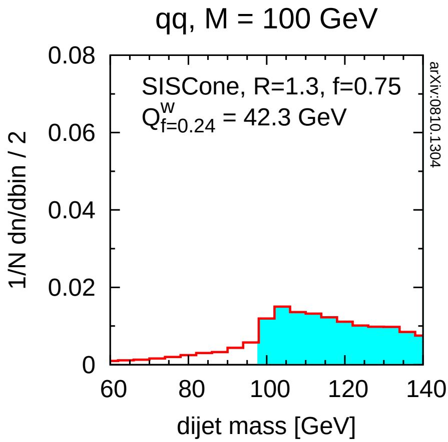


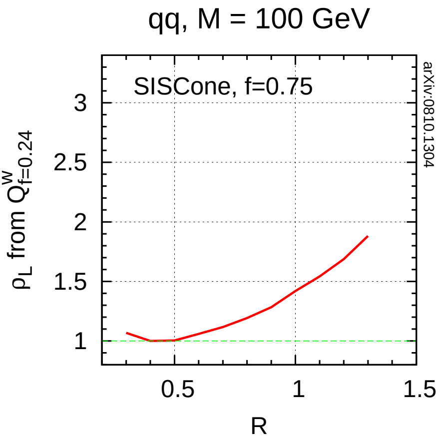
Resonance X → dijets



Dijet mass: scan over *R* [Pythia 6.4]

R = 1.3



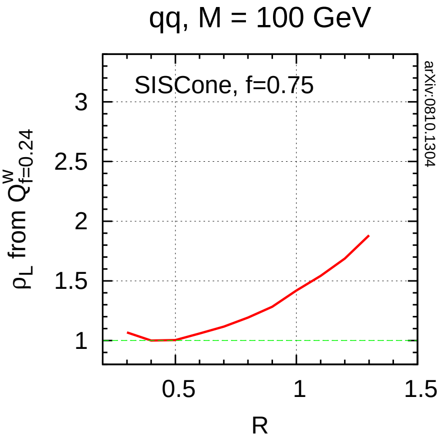


After scanning, summarise "quality" v. *R*. Minimum ≡ BEST

picture not so different from crude analytical estimate

Scan through $q\bar{q}$ mass values

$m_{qq} = 100$ GeV



Best R is at minimum of curve

- Best R depends strongly on mass of system
- Increases with mass

can reproduce this anayltically
Soyez '10

Message received by CMS: they combine all $R = 0.5$ jets ($p_t >$ 10 GeV) within $\Delta R = 1.1$ of two hardest to improve resolution.

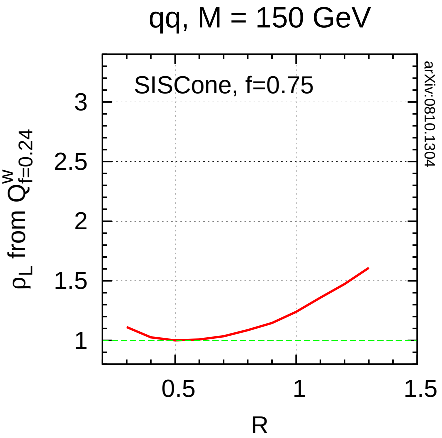
ATLAS '11 still just use $R = 0.6$

NB: 100,000 plots for various jet algorithms, narrow qq and gg resonances from `http://quality.fastjet.fr` Cacciari, Rojo, GPS & Soyez '08

Other related work: Krohn, Thaler & Wang '09

Scan through $q\bar{q}$ mass values

$m_{qq} = 150$ GeV



Best R is at minimum of curve

- Best R depends strongly on mass of system
- Increases with mass

can reproduce this anayltically

Soyez '10

Message received by CMS: they combine all $R = 0.5$ jets ($p_t >$ 10 GeV) within $\Delta R = 1.1$ of two hardest to improve resolution.

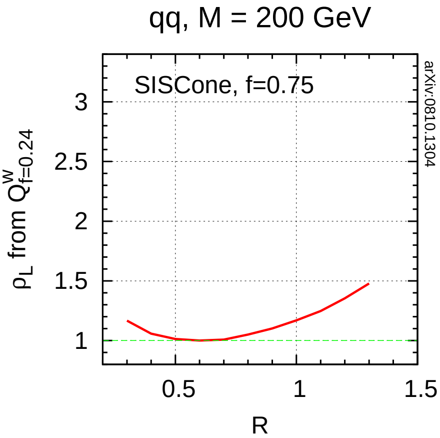
ATLAS '11 still just use $R = 0.6$

NB: 100,000 plots for various jet algorithms, narrow qq and gg resonances from `http://quality.fastjet.fr` Cacciari, Rojo, GPS & Soyez '08

Other related work: Krohn, Thaler & Wang '09

Scan through $q\bar{q}$ mass values

$m_{qq} = 200$ GeV



Best R is at minimum of curve

- Best R depends strongly on mass of system
- Increases with mass

can reproduce this anayltically

Soyez '10

Message received by CMS: they combine all $R = 0.5$ jets ($p_t > 10$ GeV) within $\Delta R = 1.1$ of two hardest to improve resolution.

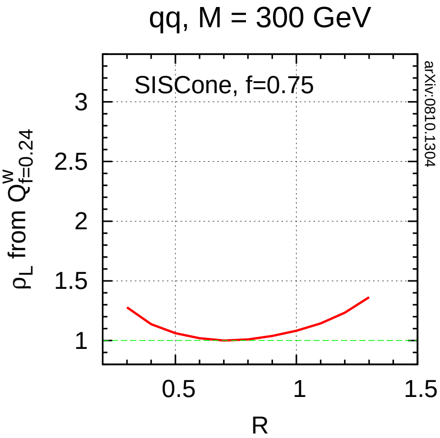
ATLAS '11 still just use $R = 0.6$

NB: 100,000 plots for various jet algorithms, narrow qq and gg resonances from `http://quality.fastjet.fr` Cacciari, Rojo, GPS & Soyez '08

Other related work: Krohn, Thaler & Wang '09

Scan through $q\bar{q}$ mass values

$\mathbf{m_{qq} = 300\ GeV}$



Best R is at minimum of curve

- Best R depends strongly on mass of system
- Increases with mass

can reproduce this anayltically
Soyez '10

Message received by CMS: they combine all $R = 0.5$ jets ($p_t > 10$ GeV) within $\Delta R = 1.1$ of two hardest to improve resolution.

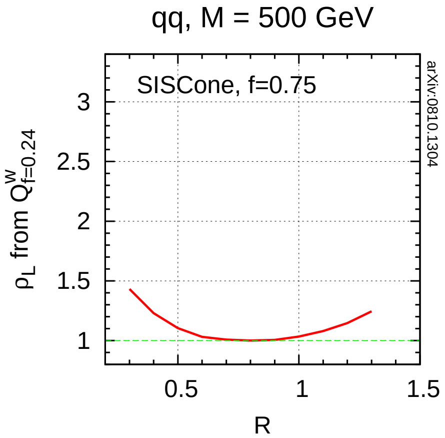
ATLAS '11 still just use $R = 0.6$

NB: 100,000 plots for various jet algorithms, narrow qq and gg resonances from `http://quality.fastjet.fr` Cacciari, Rojo, GPS & Soyez '08

Other related work: Krohn, Thaler & Wang '09

Scan through $q\bar{q}$ mass values

$\mathbf{m_{qq} = 500\ GeV}$



Best R is at minimum of curve

- Best R depends strongly on mass of system
- Increases with mass

can reproduce this anayltically

Soyez '10

Message received by CMS: they combine all $R = 0.5$ jets ($p_t > 10$ GeV) within $\Delta R = 1.1$ of two hardest to improve resolution.

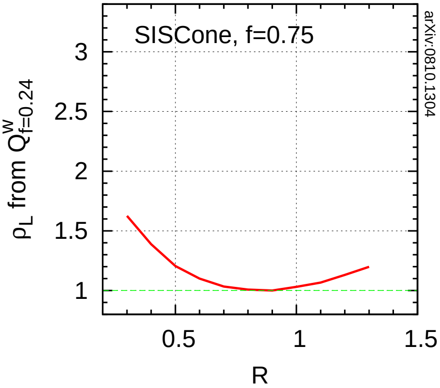
ATLAS '11 still just use $R = 0.6$

NB: 100,000 plots for various jet algorithms, narrow qq and gg resonances from `http://quality.fastjet.fr` Cacciari, Rojo, GPS & Soyez '08

Other related work: Krohn, Thaler & Wang '09

Scan through $q\bar{q}$ mass values

m_{qq} = 700 GeV



Best R is at minimum of curve

- Best R depends strongly on mass of system
- Increases with mass

can reproduce this anayltically

Soyez '10

Message received by CMS: they combine all $R = 0.5$ jets ($p_t > 10$ GeV) within $\Delta R = 1.1$ of two hardest to improve resolution.

ATLAS '11 still just use $R = 0.6$

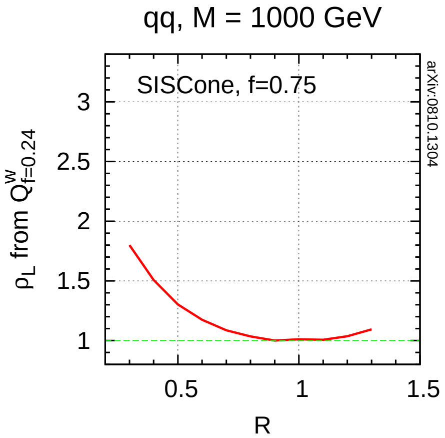
NB: 100,000 plots for various jet algorithms, narrow qq and gg resonances from http://quality.fastjet.fr

Cacciari, Rojo, GPS & Soyez '08

Other related work: Krohn, Thaler & Wang '09

Scan through $q\bar{q}$ mass values

$m_{qq} = 1000$ GeV



Best R is at minimum of curve

- Best R depends strongly on mass of system
- Increases with mass

can reproduce this anayltically

Soyez '10

Message received by CMS: they combine all $R = 0.5$ jets ($p_t >$ 10 GeV) within $\Delta R = 1.1$ of two hardest to improve resolution.

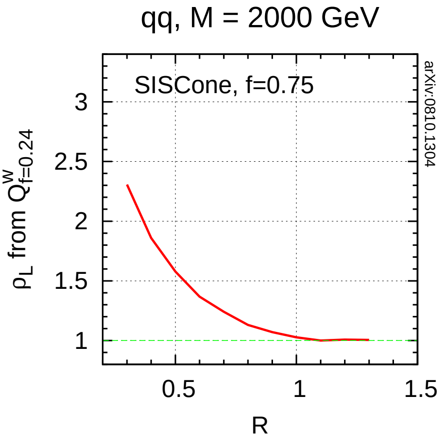
ATLAS '11 still just use $R = 0.6$

NB: 100,000 plots for various jet algorithms, narrow qq and gg resonances from `http://quality.fastjet.fr` Cacciari, Rojo, GPS & Soyez '08

Other related work: Krohn, Thaler & Wang '09

Scan through $q\bar{q}$ mass values

$m_{qq} = 2000$ GeV



Best R is at minimum of curve

- Best R depends strongly on mass of system
- Increases with mass

can reproduce this anayltically
Soyez '10

Message received by CMS: they combine all $R = 0.5$ jets ($p_t > 10$ GeV) within $\Delta R = 1.1$ of two hardest to improve resolution.

ATLAS '11 still just use $R = 0.6$

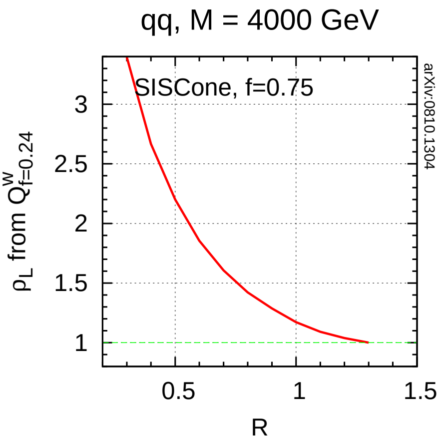
NB: 100,000 plots for various jet algorithms, narrow qq and gg resonances from `http://quality.fastjet.fr`

Cacciari, Rojo, GPS & Soyez '08

Other related work: Krohn, Thaler & Wang '09

Scan through $q\bar{q}$ mass values

$m_{qq} = 4000$ GeV



Best R is at minimum of curve

- Best R depends strongly on mass of system
- Increases with mass

can reproduce this anayltically

Soyez '10

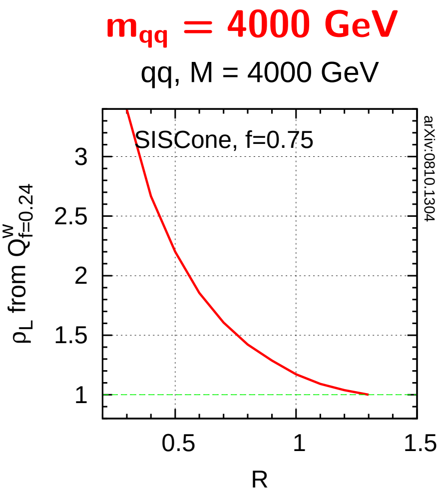
Message received by CMS: they combine all $R = 0.5$ jets ($p_t > 10$ GeV) within $\Delta R = 1.1$ of two hardest to improve resolution.

ATLAS '11 still just use $R = 0.6$

NB: 100,000 plots for various jet algorithms, narrow qq and gg resonances from `http://quality.fastjet.fr` Cacciari, Rojo, GPS & Soyez '08

Other related work: Krohn, Thaler & Wang '09

Scan through $q\bar{q}$ mass values



Best R is at minimum of curve

- Best R depends strongly on mass of system
- Increases with mass

can reproduce this anayltically
Soyez '10

Message received by CMS: they combine all $R = 0.5$ jets ($p_t > 10$ GeV) within $\Delta R = 1.1$ of two hardest to improve resolution.

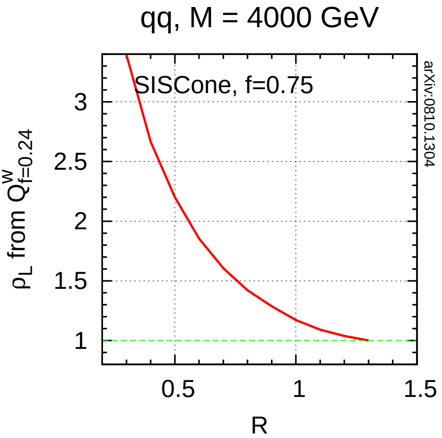
ATLAS '11 still just use $R = 0.6$

NB: 100,000 plots for various jet algorithms, narrow qq and gg resonances from `http://quality.fastjet.fr` Cacciari, Rojo, GPS & Soyez '08

Other related work: Krohn, Thaler & Wang '09

Scan through $q\bar{q}$ mass values

m_{qq} = 4000 GeV



Best R is at minimum of curve

- Best R depends strongly on mass of system
- Increases with mass

can reproduce this anayltically
Soyez '10

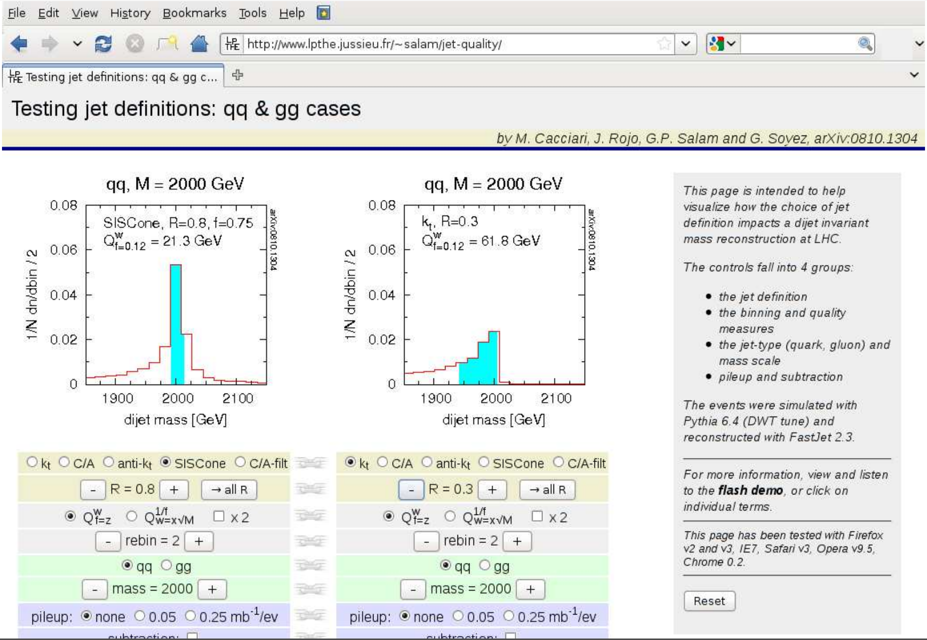
Message received by CMS: they combine all $R = 0.5$ jets ($p_t >$ 10 GeV) within $\Delta R = 1.1$ of two hardest to improve resolution.

ATLAS '11 still just use $R = 0.6$

NB: 100,000 plots for various jet algorithms, narrow qq and gg resonances from `http://quality.fastjet.fr` Cacciari, Rojo, GPS & Soyez '08

Other related work: Krohn, Thaler & Wang '09

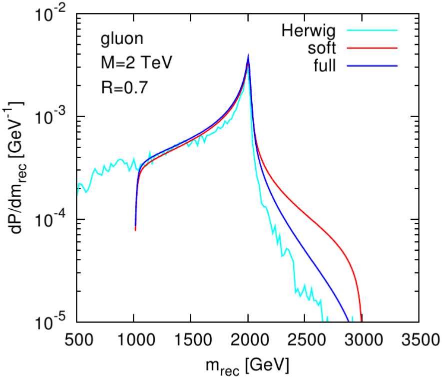
http://quality.fastjet.fr/



Analytic quality estimates

Soyez '10

Analytic v. MC lineshape

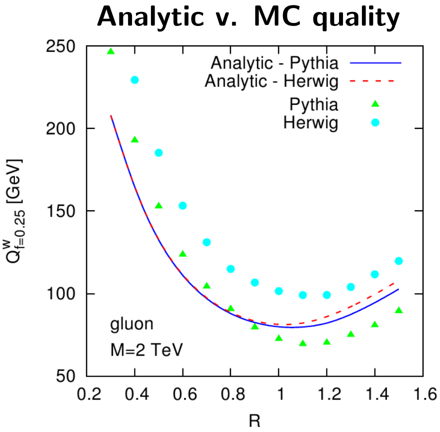


Perturbatively resum resonance "line-shape", convolute with model for non-perturbative effects.

etermine "quality" of line-shape from the analytic results, as a function of jet radius R

Analytic quality estimates

Soyez '10

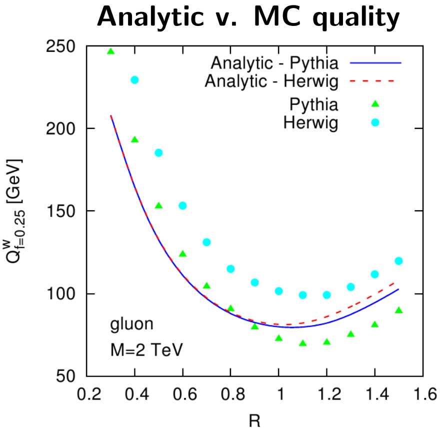


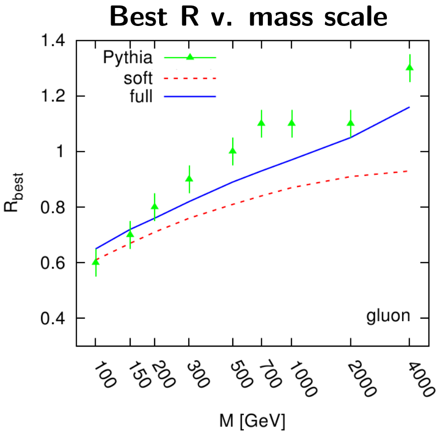
Perturbatively resum resonance "line-shape", convolute with model for non-perturbative effects.

Determine "quality" of line-shape from the analytic results, as a function of jet radius R

Analytic quality estimates

Soyez '10





Interim conclusions

Cone algorithms can be made infrared safe through an efficient exhaustive search for all stable cones — SISCone

Relation between a parton and a jet is ambiguous (because "partons" are ambiguous)

But many rule-of-thumb relations can be derived, e.g. for R-dependence from different physics contributions
[perturbative radiation, hadronisation, underlying event]

This understanding can be used to optimize choice of jet definitions

Supplementary material

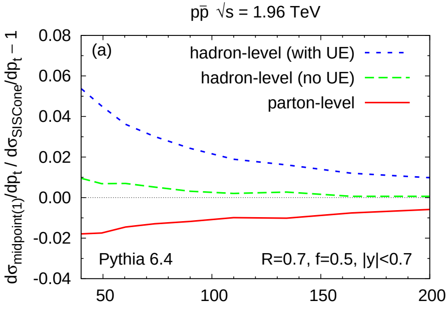
How much does IR safety *really* matter?

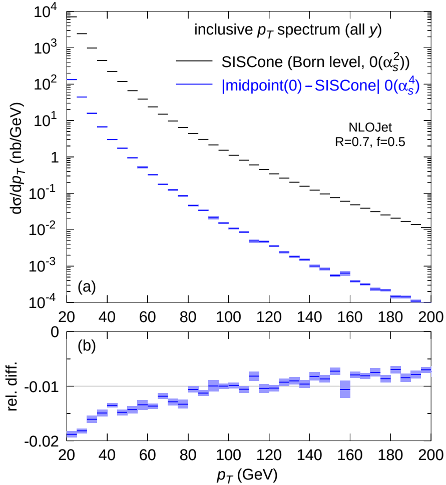
Compare midpoint and SISCone

Result depends on observable:

- inclusive jet spectrum is the least sensitive (affected at NNLO)
- larger differences ($5 - 10\%$) at hadron level

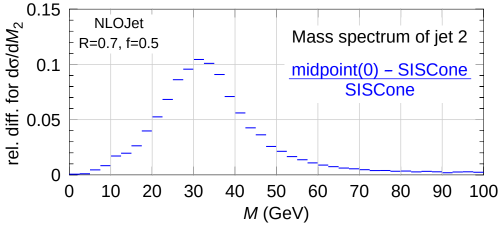
seedless reduces UE effect





IR safety & multi-jet observables

Look at jet masses in multijet events. NB: Jet masses reconstruct boosted $W/Z/H/top$ in BSM searches



Select 3-jet events

$$p_{t1,2,3} > \{120, 60, 20\} \text{ GeV},$$

Calculate LO jet-mass spectrum for jet 2, compare midpoint with SISCone.

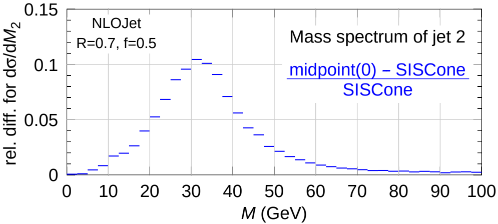
- 10% differences by default
- **40% differences** with extra cut $\Delta R_{2,3} < 1.4$

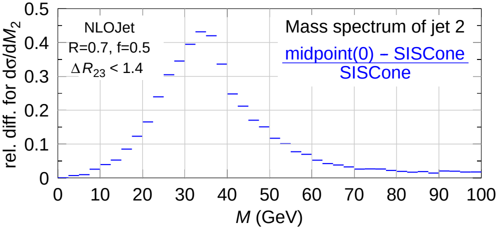
e.g. for jets from common decay chain

In complex events, IR safety matters

IR safety & multi-jet observables

Look at jet masses in multijet events. NB: Jet masses reconstruct boosted $W/Z/H/top$ in BSM searches





Select 3-jet events

$$p_{t1,2,3} > \{120, 60, 20\} \text{ GeV},$$

Calculate LO jet-mass spectrum for jet 2, compare midpoint with SISCone.

- 10% differences by default
- **40% differences** with extra cut $\Delta R_{2,3} < 1.4$

  e.g. for jets from common decay chain

In complex events, IR safety matters

Bottom line on IR safety

- IR safety often matters less in *inclusive* quantities
- It matters more in multi-jet cases

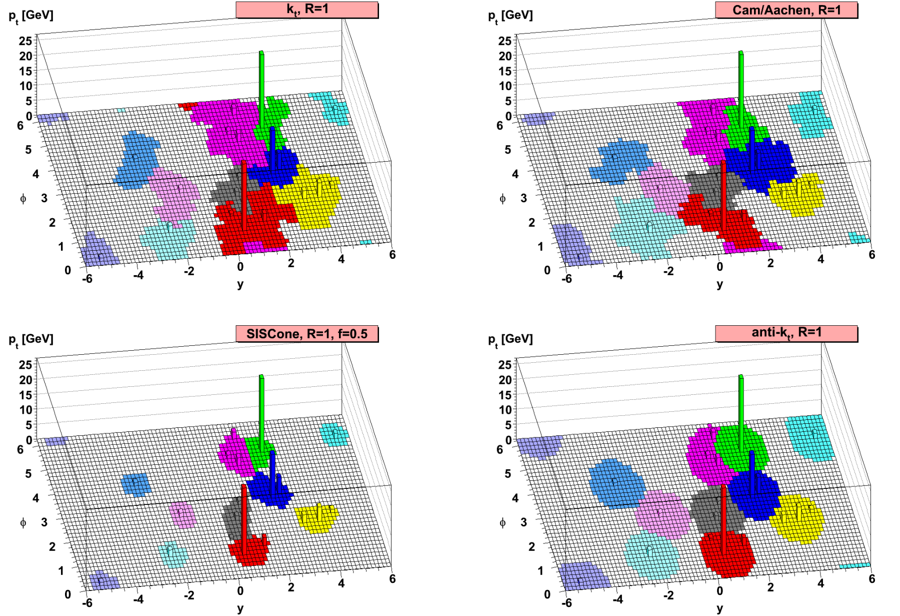
- JetClu (IC-SM) is *very bad* So is ATLAS cone (no longer used)
- Midpoint (IC_{mp}-SM) *moderately bad*

So is CMS cone (IC-PR), now only used in trigger

- An IRC safe cone algorithm exists (SISCone)
- **Avoid trouble later: use IR-safe algs from the start**

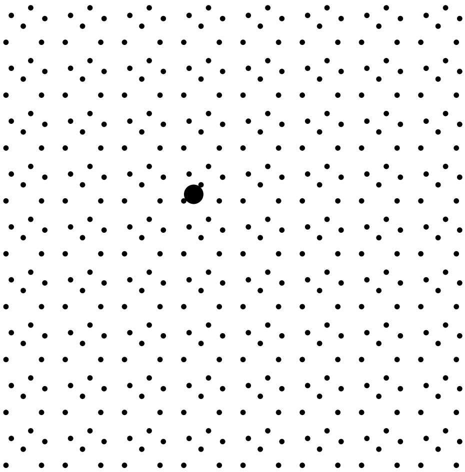
cf. CDF W+jets

Jet contours – visualised



E.g. SISCone jet area

1. One hard particle, many soft



SISCone, any R, $f \gtrsim 0.391$

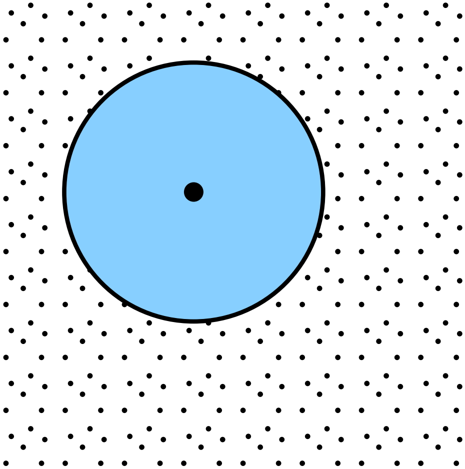
Jet area =

Measure of jet's susceptibility to uniform soft radiation

Depends on details of an algorithm's clustering dynamics.

E.g. SISCone jet area

2. One hard stable cone, area $= \pi R^2$



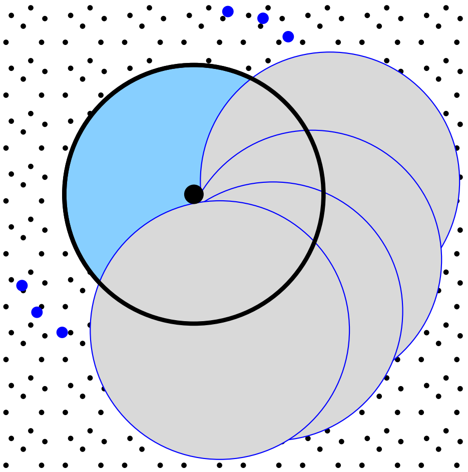
SISCone, any R, $f \gtrsim 0.391$

Jet area =
Measure of jet's susceptibility to uniform soft radiation

Depends on details of an algorithm's clustering dynamics.

E.g. SISCone jet area

3. Overlapping "soft" stable cones



SISCone, any R, $f \gtrsim 0.391$

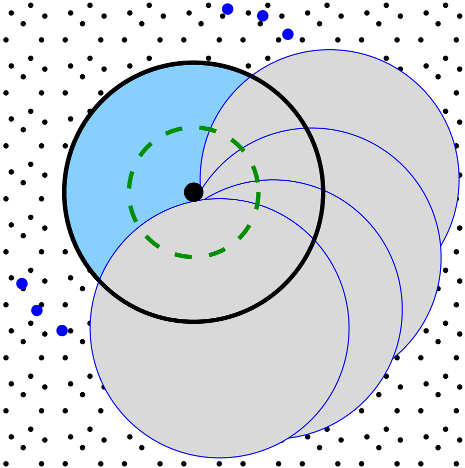
Jet area =

Measure of jet's susceptibility to uniform soft radiation

Depends on details of an algorithm's clustering dynamics.

E.g. SISCone jet area

4. "Split" the overlapping parts



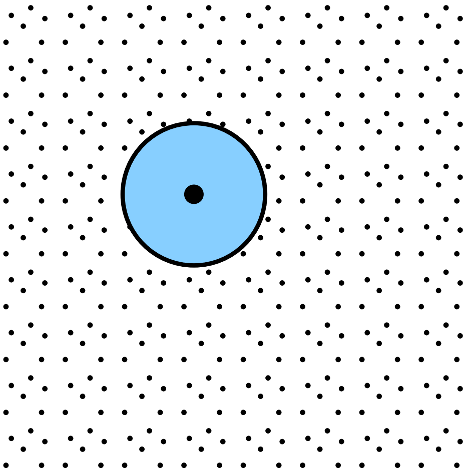
SISCone, any R, $f \gtrsim 0.391$

Jet area =
Measure of jet's susceptibility to uniform soft radiation

Depends on details of an algorithm's clustering dynamics.

E.g. SISCone jet area

5. Final hard jet (reduced area)



SISCone, any R, $f \gtrsim 0.391$

Jet area =
Measure of jet's susceptibility to uniform soft radiation

Depends on details of an algorithm's clustering dynamics.

SISCone's area (1 hard particle)
$$= \mathbf{\frac{1}{4}}\,\pi R^2$$

Small area ≡
low sensitivity to UE & pileup

Jet algorithm properties: summary

	k_t	Cam/Aachen	anti-k_t	SISCone
reach	R	R	R	$(1+\frac{p_{t2}}{p_{t1}})R$
$\Delta p_{t,PT} \simeq \frac{\alpha_s C_i}{\pi} \times$	$\ln R$	$\ln R$	$\ln R$	$\ln 1.35R$
$\Delta p_{t,hadr} \simeq -\frac{0.4\,\mathrm{GeV}\,C_i}{R} \times$	0.7	?	1	?
area $= \pi R^2 \times$	0.81 ± 0.28	0.81 ± 0.26	1	0.25
$+\pi R^2 \frac{C_i}{\pi b_0} \ln \frac{\alpha_s(Q_0)}{\alpha_s(Rp_t)} \times$	0.52 ± 0.41	0.08 ± 0.19	0	0.12 ± 0.07

In words:

- k_t: area fluctuates a lot, depends on p_t (bad for UE)
- Cam/Aachen: area fluctuates somewhat, depends less on p_t
- anti-k_t: area is constant (circular jets)
- SISCone: reaches far for hard radiation (good for resolution, bad for multijets), area is smaller (good for UE)