Jet Reconstruction Algorithms in Deep Inelastic Scattering

Summer Student Report

at

H1 Collaboration at DESY

submitted by

Anton Tamtögl

H1 Collaboration,
DESY (Deutsches Elektronen Synchrotron)
Notkestraße 85
22607 Hamburg, Germany

14.09.2007

Advisor: Dr. Günter Grindhammer
Co-Advisor: Dipl.-Ing. Roman Kogler

Contents

1 Introduction: Jets **4**

2 Jet-Finding-Algorithms **5**
2.1 The k_t-Jet Algorithm [1] 5
2.2 The Cone Jet Algorithm 6

3 The SiSCone jet algorithm **8**
3.1 Infrared safety 8
3.2 The SiSCone jet algorithm 8

4 Application and Comparison **10**
4.1 Hadron level 10
4.2 Detector level 14
4.3 Hadronization corrections 16
4.4 Influence of R and f in the SiSCone algorithm 16
4.5 Computation time 19

5 Jet Areas **20**
5.1 Why Jet areas? 20
5.2 Jet areas 20
5.2.1 Application 21
5.3 Determination of the background level 23
5.4 Correcting jets 24

Aknowledgement

I would like to thank Günter, Roman, Juraj and the people at H1 for their support and the pleasant and cooperative working atmosphere.

1 Introduction: Jets

A jet is a narrow cone of hadrons and other particles produced by the formation of a quark or gluon after high-energy collisions. When energetic partons from the hard subprocess hadronize, the created hadrons remain collimated around the original parton directions and the higher the energy the parton has the more collimated the hadrons are. These bunches of hadrons are called jets. They can be interpreted as immediate link to the partons and thus can provide a deeper view of the underlying parton interactions.

Jets sketch a rather simple picture of what happened in an event without taking particular into account the multiparticle dynamics. The reason for using the jets, rahter than directly the observed hadrons, is that they can be construed as infrared-safe observables. Therefore perturbative QCD may be used to make predictions of jets and their sensitvity to non-perturbative phenomena (hadronisation, underlying event and pileup effects) is rather small.

2 Jet-Finding-Algorithms

Jet analysis techniques involve jet reconstruction. To find jets which provide a stable basis for theoretical predictions, several jet finding algorithms have been developed. At present there are essentially two classes of jet algorithms in use. These are cone-type algorithms and clustering algorithms.

- In cone-type algorithms jets are defined by maximizing the amount of energy which can be covered by cones of fixed size. Thus a jet in this definition is a set of particles whose momentum vectors lie within a certain angular cone. Since this contribution is essentially determined by geometry it is relatively easy to estimate.
- In clustering algorithms particles are assigned to jets iteratively by recursively grouping sets of particles with "nearby"' momenta, as defined by some measure, into larger sets of particles.

In the case of hadron collisions the jet algorithm has to fulfill the requirements of being infrared and collinear safe. In addition the algorithm should be subject to small hadronization corrections and not strongly affected by contamination from hadron remnants and the underlying soft event.
Infrared and collinear safety of the algorithm implies that the number of found jets and their properties must not change when one of the objects radiates a very soft object, or splits into two collinear objects. Furthermore the output of the jet algorithm shall be invariant under a longitudinal Lorentz boost. Since the invariance is not only longitudinal it should be taken care of that the jet reconstruction is performed in an appropriate frame, e.g. the Breit frame in case of DIS.

2.1 The k_t-Jet Algorithm [1]

A type of clustering algorithm that fulfills the above mentioned criteria is the so called k_t-jet algorithm which represents at the moment one of the most widely used jet-finding algorithms at H1.
The jet algorithm starts with a list of objects (partons or hadrons depending on which level the jet algorithm is applied) called protojets, which are characterized by their transverse energy E_T, rapidity $\eta = -\ln\left[\tan\left(\frac{\theta}{2}\right)\right]$ and the azimuthal angle ϕ. These protojets are combined iteratively into jets, according to their distance in the η-ϕ plane and their transverse energies, using the following steps:

1. For each protojet i the beam distance d_i and for each pair of protojets ij the distance d_{ij} between those pair is calculated using:

$$d_i = E_{T,i}^2, \quad d_{ij} = min(E_{T,i}^2, E_{T,i}^2)[(\eta_1 - \eta_2)^2 + (\phi_1 - \phi_2)^2]/R^2 \tag{2.1}$$

   R is an adjustable parameter related to the opening angle of the jets. For the usage of the k_t-algorithm in my applications I was always using the theoretically preferred value $R = 1$.

2. The minimum d_{min} of all d_i and d_{ij} is found

3. If d_{min} is one of the d_{ij}'s, the corresponding protojets i and j are merged into a new protojet. There are many possibilities of how their kinematic quantities may be recombined. When using this algorithm I was always applying the so-called p_t-recombination scheme

4. If d_{min} is one of the d_i's, the corresponding protojet i is considered as a final jet and is removed from the list of protojets.

This procedure is repeated until there are no more protojets left. The algorithm gives rise to a list containing typically many jets for each event.
However, only the jets with large values of p_t are of particular physical interest. Therefore a minimal cut of the p_t of a jet, upon which the jet is rejected, could be done. Except the cases where it is explicitly mentioned, I was always using a $p_{t,cut} = 3GeV$.

2.2 The Cone Jet Algorithm

As mentioned above the algorithm makes use of the jet definition of a set of particles whose momentum vectors lie within a certain angular cone. The idea is to maximize the amount of energy covered by cones of fixed size given by the cone-radius R which is basically done in two steps:

1. First the algorithm tries to find all stable cones:
   - Event particles are taken as seeds i.e. trial cone directions.
   - For each seed a list of particles in the trial cone is established and the sum of the four-momenta of the particles inside the cone is calculated.
   - If this combined momentum of the cone-content is pointing in the same direction as the cone itself, a stable cone has been found.
   - If the cone axis does not coincide the combined momentum is now used as a new trial cone direction.

   This iteration is performed until the cone directions are no longer changing and all stable cones have been found.

2. In the case that cones are overlapping a split-merge procedure is run:

   - The sum of the transverse momentum $p_{t,\text{shared}}$ of all the shared particles between the cones i and j is calculated:

$$p_{t,\text{shared}} = \sum_{k \in i \& j} |p_{t,k}| \tag{2.2}$$

   - If the transverse momentum of the overlap is large compared to that of the jet that is

$$p_{t,\text{shared}} > f \cdot p_{t,j} \tag{2.3}$$

     than the two protojets are merged into a single new protojet. These procedure is determined by the split-merge parameter f which is usually set to 0.5 (which means that the two jets are merged if their overlap contains more than 50% of the transverse cone-momentum)

   - In the latter case (if $p_{t,\text{shared}} < f \cdot p_{t,j}$ holds) each shared particle is assigned to the cone to which it is closes. The protojets need than to be calculated.

The split merge part of the cone-algorithm also provides the possibility of cuts to the jet, that is to reject protojets with a p_t smaller than a threshold parameter $p_{t\text{min}}$.

The cone-algorithm provides a rather illustrative and geometrical image but suffered from a few problems i.e. it was neither infrared nor collinear-safe.

3 The SiSCone jet algorithm

3.1 Infrared safety

For an infrared and collinear safe algorithm the number of found jets and their properties should not change when one of the objects radiates a very soft object, or splits into two collinear objects. But it can be shown that the found cones in case of cone-type algorithms depend on the seeds, that is the trial cone directions which are chosen, and the found cones change if a soft particle is added in a hard event.

Thus the infrared-unsafety of the cone-type algorithm is a problem of the seed and may be eliminated by the usage of a a seedless cone-algorithm. This algorithm does not make use of seeds in the event as trial cone directions but uses all possible subsets of the event as trial cone directions. It guarantees that all stable cones are found but the computing time is of the order of $N \cdot 2^N$ (N = number of particles in the event) which makes the algorithm not applicable.

A faster cone-type algorithm that solves the problem of the infrared and collinear-unsafety is the SiS-Cone algorithm invented by Gavin P. Salam and Grégory Soyez that is mathematically proven to be infrared-safe.

3.2 The SiSCone jet algorithm

The idea of the SiSCone algorithm is to identify not all possible cones but just all cones with different contents and to test their stability. In fact, cones are just circles in the η-ϕ-plane, so one has to find all distinct circular enclosures of a set of points. To determine those, one may utilize the fact that any enclosure can be moved, until a pair of points lies on its edge, which is shown in figure 3.1

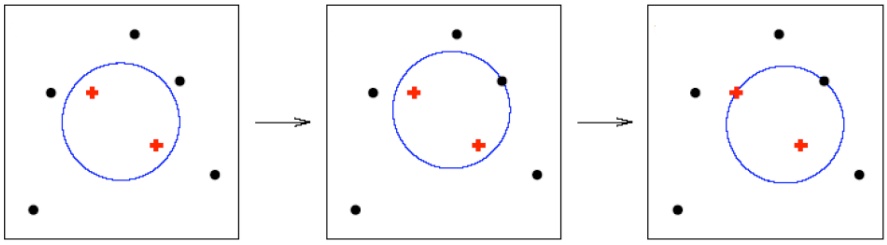


Figure 3.1:

The clue about it is, that one can always move the corresponding circle without changing the enclosure contents into a position, where two points lie on its boundary: First slide the circle until its point content changes. The circle edge hits now some point in the plane. Now pivot the circle around the boundary point until the second point touches the circle boundary.
Thus to find all distinct enclosures, one has to find all circles whose circumference lies on a pair of points form the set and to consider permutations of edge points, that is whether the edge points are included or not to the circle content. The computation time for this procedure is of the order $N \cdot \ln(n)$ with N the number of particles in the event and n the typical number of particles in a circle.

To reduce the computation time the algorithm first executes an ordering of the boundary points: For each pair of particles i and j two circles can be found and the angle ζ of the circle-centre relative to i is calculated according to the following equation:

$$\zeta = \arctan\left(\frac{\Delta\phi_{i,C}}{\Delta\eta_{i,C}}\right) \tag{3.1}$$

When working through this ordering the circle content changes only by one particle at a time, either by a particle entering or leaving the circle. Thus it is easy to update the momentum of the circle content by adding or removing the momentum of the particle that has entered or left the circle.
Another trick to reduce the computation time is to find the candidate cones first and not to check stability explicitly, but examine whether the edge points that define the cone affect the momentum axis. If they do, the cone is labeled as unstable, if not as stable. The full stability test is then carried out later, using all the stable candidate cones.

After all stable cones have been found a split merge porcedure according to equation 2.3 is run over the protojets and the list of final jets is established.

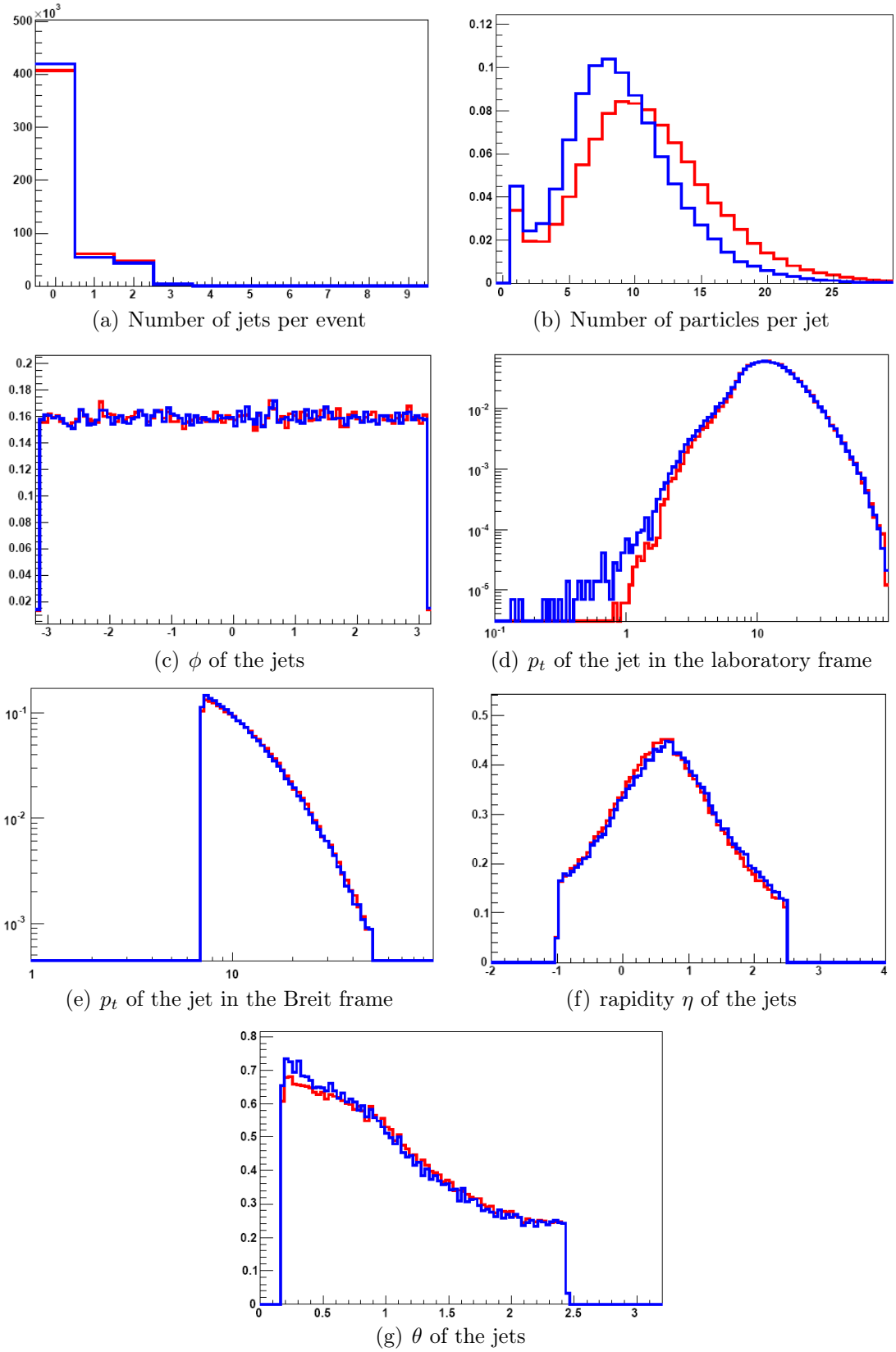
For more details about the implementation of the SiSCone algorithm see Appendix A or [2]-[4].

4 Application and Comparison

The behavior of the SiSCone jet algorithm was studied by using Monte-Carlo Events of H1 and running the algorithm to reconstruct the jets out of the MC-data. The properties of the jets were extensively compared with the jet properties one achieves when using the k_t jet finding algorithm. For this purpose the parameters in the SiSCone jet algorithm were always set to $R = 0.7$, $f = 0.5$ and $p_{t\min} = 3GeV$ (corresponding to the $p_{t,\text{cut}}$ in the k_t-algo) except the few cases where it is mentioned explicitly.

4.1 Hadron level

The application of both algorithms on the hadron level of the event is shown in figure 4.1. They have been used to compute $2 \cdot 10^6$ MC events of H1 including initial and final state photon radiation. Several cuts on the jets where performed including a cut on all jets with a $p_t < 7GeV$ in the Breit frame and accepting only jets with $-1 < \eta < 2.5$.
Figure 4.1(a) shows the number of jets per event found whereas both algorithms match fairly good. The number of particles per jet (figure 4.1(b)) is shifted to larger values for the k_t algorithm than for the SiSCone algorithm, which may be explained by means that for the k_t algorithm it is more likely to find again a particle that may be combined to the jet, whereas this is somehow restricted in case of the SiSCone algorithm by the cone-radius R. The jets with just one particle my be traced to photons of initial or final state radiation. The high value of jets with just two or three particles is caused by jets with a small p_t, that are passing the cuts because photons of initial and finial state radiation are accounted to the jet. Therefore the jet gets a higher p_t after the clustering than the real one.
The ϕ-distribution of the jets (figure 4.1(c)) is uniform in case of both algorithms which one would except since there is no preferred direction in the x-y-plane. The transverse momentum p_t in the laboratory and Breit frame are shown in figure 4.1(d) and 4.1(e) respectively. There is a good coincidence of these properties for the different algorithms. Plots of η (figure 4.1(f)) and θ (figure 4.1(g)) give rise to the same conculsion.



(a) Number of jets per event

(b) Number of particles per jet

(c) ϕ of the jets

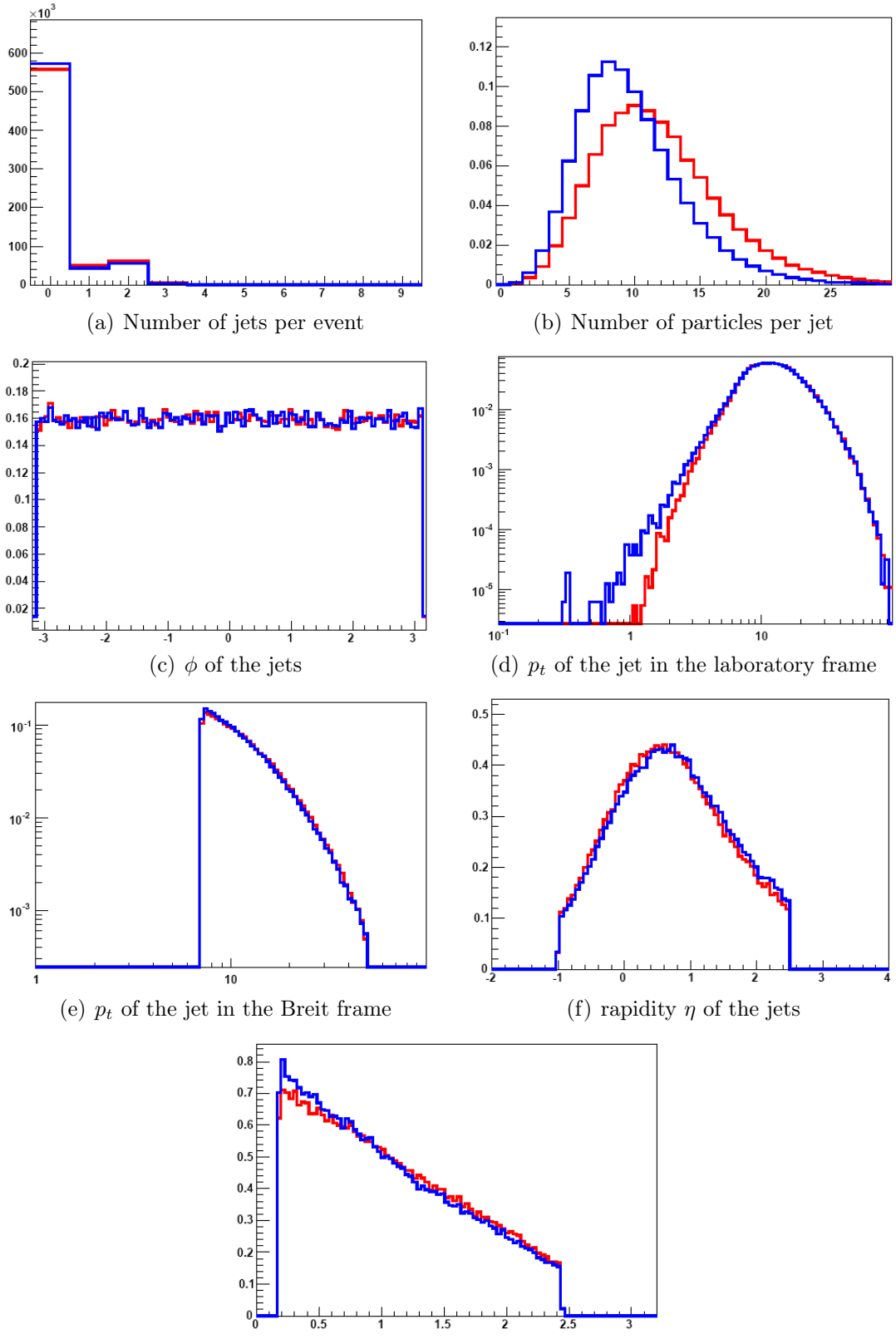
(d) p_t of the jet in the laboratory frame

(e) p_t of the jet in the Breit frame

(f) rapidity η of the jets

(g) θ of the jets

Figure 4.1: Comparison of the k_t jet finding algorithm (red) and the SiSCone jet finding algorithm (blue) on hadron level.



(a) Number of jets per event

(b) Number of particles per jet

(c) ϕ of the jets

(d) p_t of the jet in the laboratory frame

(e) p_t of the jet in the Breit frame

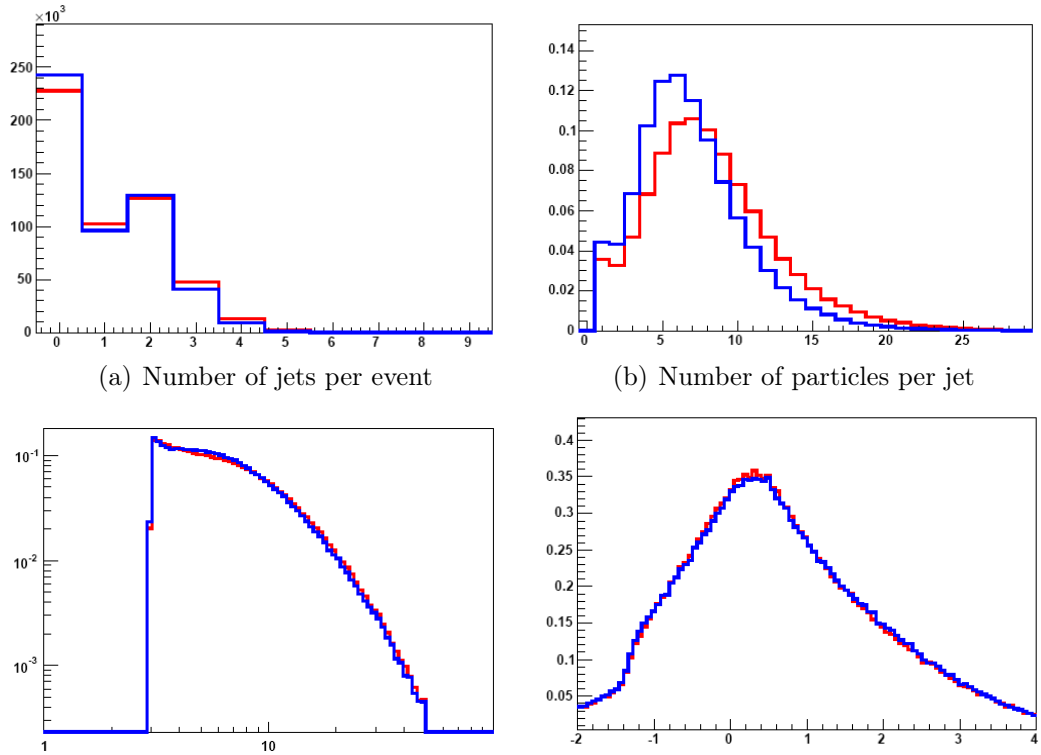
(f) rapidity η of the jets

(g) θ of the jets

Figure 4.2: Comparison of the k_t jet finding algorithm (red) and the SiSCone jet finding algorithm (blue) on hadron level without initial and final state QED-radiation.

The same computations as described above were performed again, but now using MC events excluding initial and final state photon radiation (see figure 4.2). This fact gives rise to more events with two jets than with one jet (figure 4.2(a)) because no more photons are clustered together with the jet and therefore less jets are passing the cuts. Thus this situation reflects the physics behind the process much better. In addition no jets with just one particle are observed and the number of jets with two or three particles is much smaller (figure 4.2(b)).
The other jet-properties (figure 4.2(c) - 4.2(g)) rarely show any significant changes and the coincidence for both algorithms is again quite good.

To exclude any loss of information because of the jet cuts the computation of MC events including initial and final state photon radiation was performed for both algorithms again but now without cuts on p_t and η. The results are shown in figure 4.3 and again, despite the already known fact that the number of particles per jet is different, there is a good coincidence of the algorithms.
Although there are no cuts in p_t, protojets with values of $p_t < 3GeV$ are still rejected. It was also tried to switch off this parameter, which gives rise to a little bit more jets with just one particle, but despite this fact the output did not change.



(a) Number of jets per event

(b) Number of particles per jet

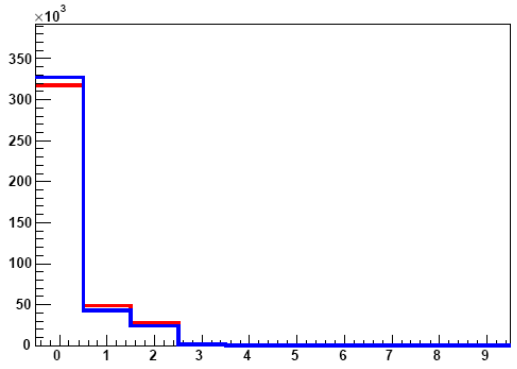
(c) p_t of the jet in the Breit frame

(d) rapidity η of the jets

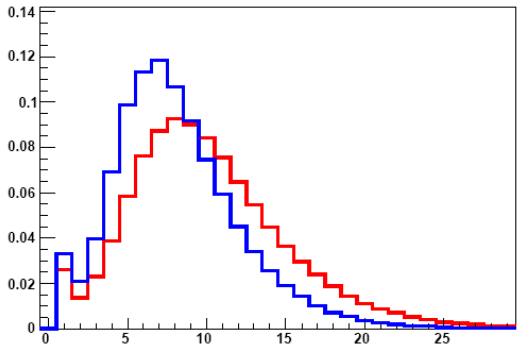
Figure 4.3: Comparison of the k_t jet finding algorithm (red) and the SiSCone jet finding algorithm (blue) on hadron level without performing cuts in p_t and *eta*

4.2 Detector level

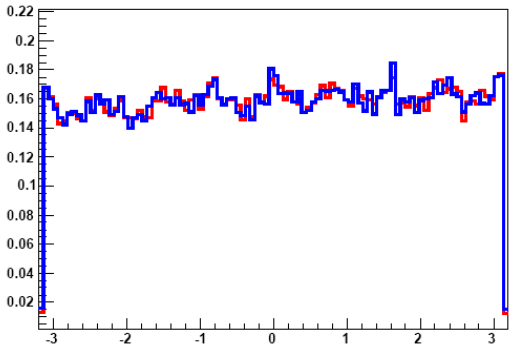
Again both algorithms were used to compute $2 \cdot 10^6$ MC events of H1 including initial and final state photon radiation but now on the detector level. Cuts on the jets were performed to accept only jets with $p_t > 7GeV$ in the Breit frame and $-1 < \eta < 2.5$.
Less events with two jets are now observed (see figure 4.4(a)) which may be explained by the fact that not all particles are realized in the detector. Thus the jet has a smaller p_t since less particles are clustered together and less jets are passing the cuts. Figure 4.4(b) shows that there are now less one-, two- and three- paricle-jets than compared to the jets on hadron level.
The ϕ-distribution of the jets (figure 4.4(c)) is not perfectly uniform any more, there are steps. This is caused by the geometry of the detector, that is the particles can not be detected over the whole range of the angle ϕ. The transverse momentum p_t in the laboratory and Breit frame, shown in figure 4.4(d) and 4.4(e) respectively, as well as the plots of η (figure 4.4(f)) and θ (figure 4.4(g)) match good.



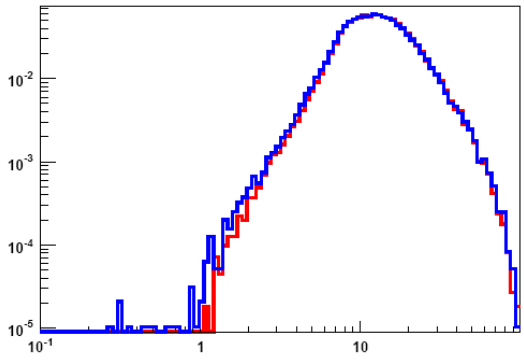
(a) Number of jets per event



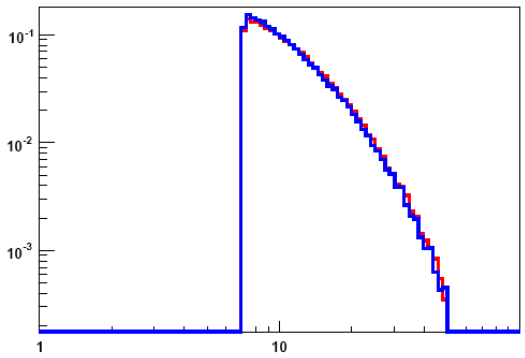
(b) Number of particles per jet



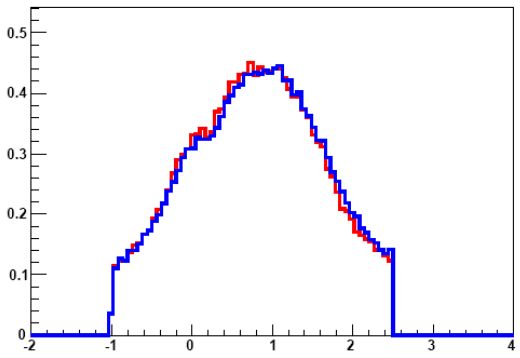
(c) ϕ of the jets



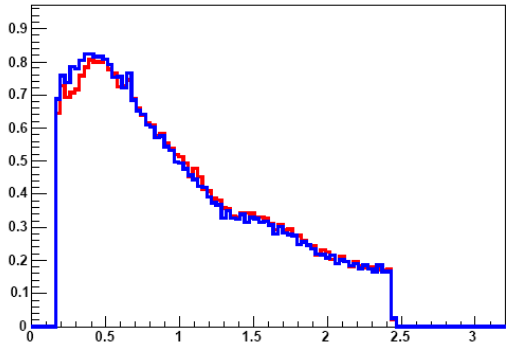
(d) p_t of the jet in the laboratory frame



(e) p_t of the jet in the Breit frame



(f) rapidity η of the jets



(g) θ of the jets

Figure 4.4: Comparison of the k_t jet finding algorithm (red) and the SiSCone jet finding algorithm (blue) on detector level.

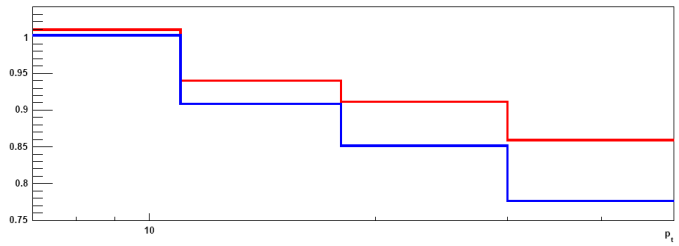
4.3 Hadronization corrections

Usually the outcome of jet-algorithms is different for the parton and hadron level. This difference between measured observables on the parton and hadron level may be corrected using so called hadronization correction factors. To determine these factors, histograms of the observables with a fixed binning are plotted. For each observable the definition of the correction factor is:

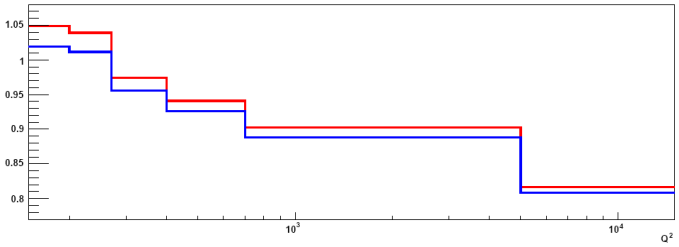
$$c_{had}^{corr} = \frac{N_{\text{bin},i}^{\text{part}}}{N_{\text{bin},i}^{\text{had}}} \tag{4.1}$$

Thus one has to determine the content of each bin of the histogram on the parton level and to divide it with the content on the hadron level.

Figure 4.5 shows the correction factors for p_t and Q^2. For p_t (figure 4.5(a)) the correction for the SiSCone algorithm is up to 10% worse than for the k_t algorithm. For Q^2 (figure 4.5(a)) it is up to 5% worse in case of the SiSCone algorithm.



(a) c_{had}^{corr} for p_t



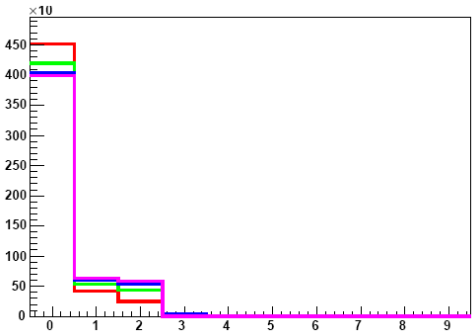
(b) c_{had}^{corr} for Q^2

Figure 4.5: Hadronization correction factors c_{had}^{corr} for the k_t jet finding algorithm (red) and the SiSCone jet finding algorithm (blue).

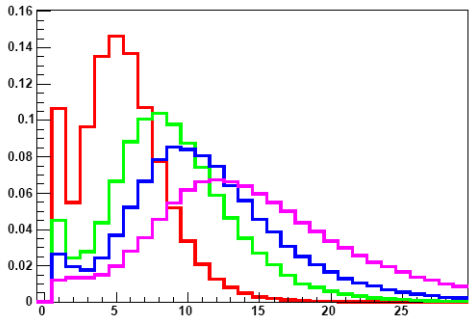
4.4 Influence of R and f in the SiSCone algorithm

To determine the influence of the cone-radius R and the split merge parameter f in the SiSCone algorithm on the jet parameters, $2 \cdot 10^6$ MC events including initial and final state photon radiation were computed on the hadron level with different values of the parameters.

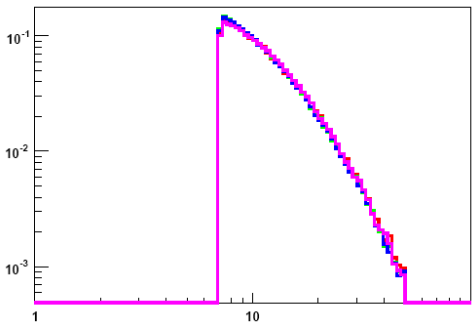
Figure 4.6 shows the jet properties for different values of the cone-radius R. It is quite obvious that the number of jets (figure 4.6(a)), as well as the number of particles per jet (figure 4.6(b)), increases with increasing R, since more particles may be assigned to a larger cone.
The transverse momentum of the jets (figure 4.6(c)) does not show any significant changes when modifying the cone-radius, but in figure 4.6(d) and 4.6(e) it is observed that the number of jets with a small θ increases with increasing cone radius. This may be explained by means, that with increasing cone radius more particles flying in backward direction are assigned to the jet and therefore the angle θ of the jet becomes smaller.



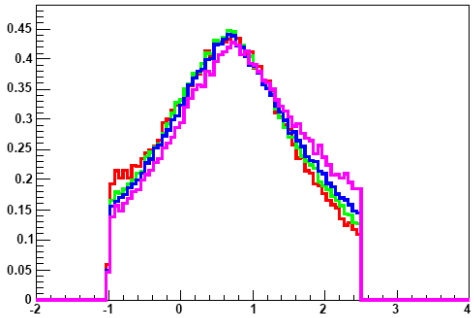
(a) Number of jets per event



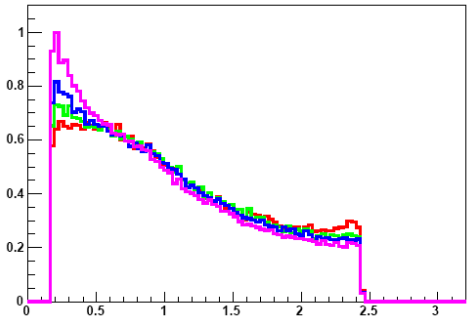
(b) Number of particles per jet



(c) p_t of the jet in the Breit frame



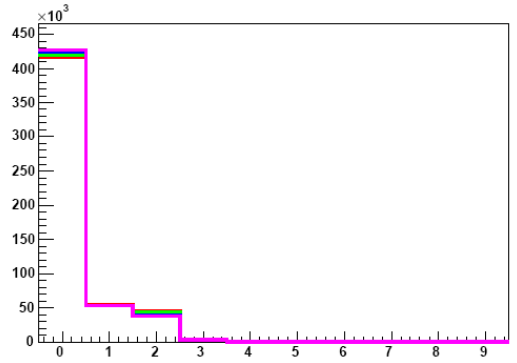
(d) rapidity η of the jets



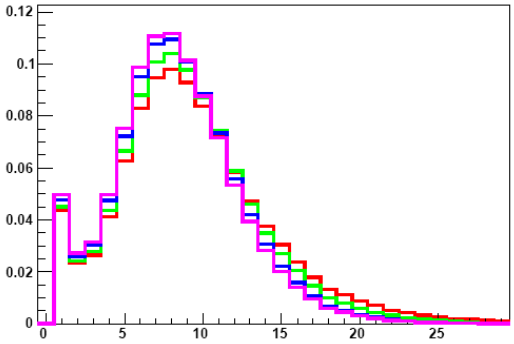
(e) θ of the jets

Figure 4.6: Jet reconstruction on hadron level using the SiS-cone algorithm with different values of the cone radius R: $R = 0.3$(red), $R = 0.7$(green), $R = 1.0$(blue), $R = 1.5$(magenta)

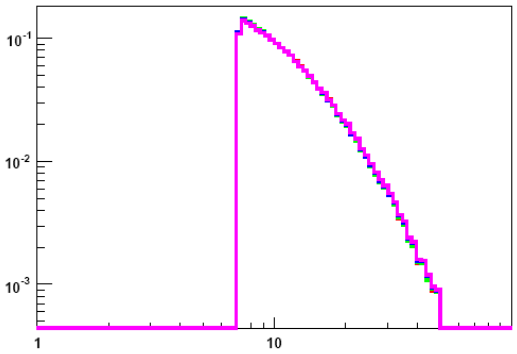
In Figure 4.7 the jet properties for different values f of the split merge procedure (see equation 2.3) are shown. The greater f gets the greater the amount of momentum in the overlap compared to the cone has to be, so that the cones are merged. Thus with increasing f the chances that two cones are merged become smaller and smaller. This gives rise to a decrease in the number of jets per event (figure 4.7(a)) and a decrease of particles per jet (figure 4.7(b)). The transverse momentum (figure 4.7(c)) is hardly influence by this parameter.



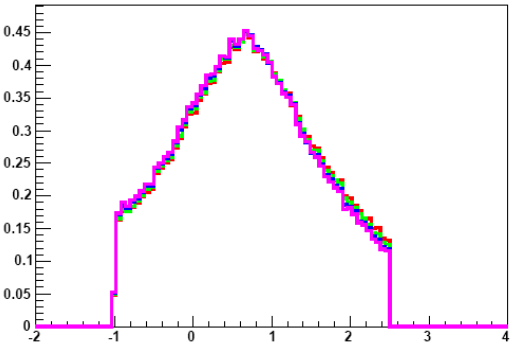
(a) Number of jets per event



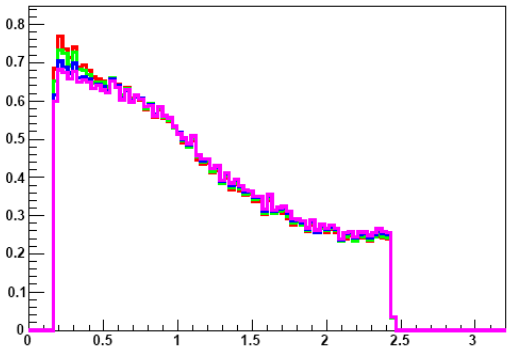
(b) Number of particles per jet



(c) p_t of the jet in the Breit frame



(d) rapidity η of the jets



(e) θ of the jets

Figure 4.7: Jet reconstruction on hadron level using the SiS-cone algorithm with different values of the split merge parameter f (see formula 2.3): $f = 0.25$(red), $f = 0.5$(green), $f = 0.75$(blue), $f = 0.9$(magenta)

4.5 Computation time

A comparison of the computation time for the two algorithms is shown in figure 4.8 whereas it is observed that the SiSCone algorithm is even a little bit faster. So to say the computing time for the k_t algorithm is of a factor of about 1.3 greater then those of the SiSCone algorithm.

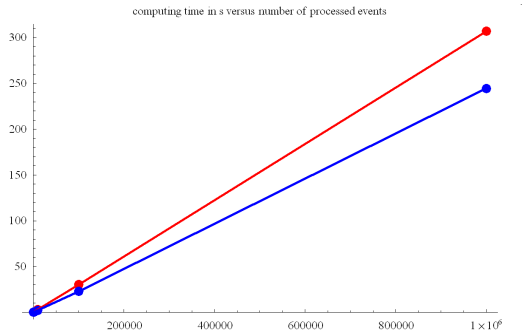


Figure 4.8: Computing time versus number of processed events of the k_t(red) and the SiSCone jet finding algorithm (blue).

5 Jet Areas

5.1 Why Jet areas?

Jets show a certain susceptibility to contamination from soft radiation e.g. a uniform diffuse background ρ. This background may be originated from pileup by multiple minimum bias collisions in high luminosity hadron colliders and non perturbative QCD radiation. The amount of this diffuse background radiation that is clustered together with the jet is proportional to the jet-area A. To perform jet corrections, according to the background, one has to determine the jet area and the background level ρ.

5.2 Jet areas

One can define the jet area as the surface in the η-ϕ-plane over which the particles, clustered to one jet, are distributed.
Starting with the particles in an event, after the clustering the particles are assigned to the jets in the event. To determine the area of these jets one can tile the plane, count the cells of a jet and then sum the areas (see figure 5.1).

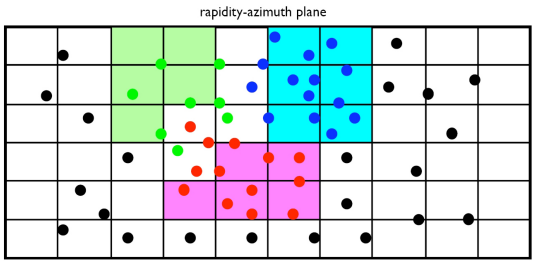


Figure 5.1: Determination of the jet area by tiling the η-ϕ-plane

But the problem is that it is not clear to which jet the particles in the tile between two different jets belongs. In addition the tiles can not be made infinitesimally small because then the area would approach zero.

A better concept, introduced by Matteo Cacciari and Gavin P.Salam ([5] and [6]), is to add a large number of uniformly distributed and extremely soft particles, called ghosts. This ghost particles form a grid in the η-ϕ-plane. If the jet-algorithm is

infrared-safe the final set of hard jets will not change when a number of soft particles is added to the event. If the jet algorithm is now running on the event including the ghost grid, the ghost particles are clustered together with the real particles which is illustrated in figure 5.2.

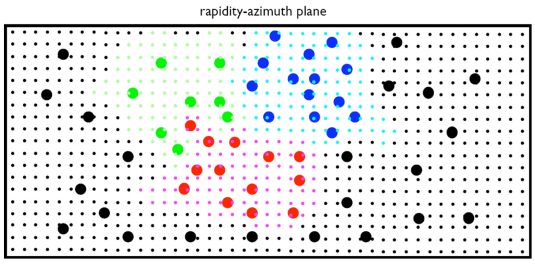


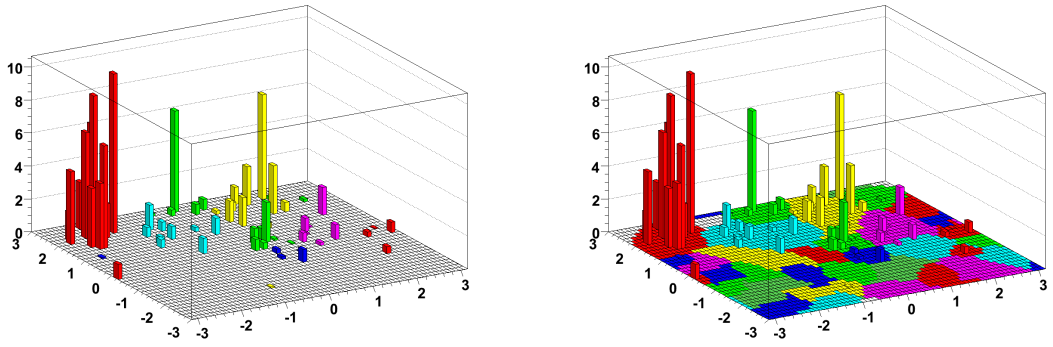
Figure 5.2: Determination of the jet area by adding a grid of soft particles

Now a given set of ghosts belongs to each jet and the area of the jet can be determined by counting the number of ghosts belonging to the jet and multiplying this number with the average area of a single ghost.

The definition of jet areas in this context is as follows:
The active area of a jet is proportional to the number of uniformly distributed, infinitely soft particles that get clustered in it.

5.2.1 Application

To illustrate the process of determining the jet area, a 3-dimensional histogram of one single event was ploted (see figure 5.3) with η, ϕ and the energy E of the particle to represent the third dimension. After running the jet algorithm on the event, each particle is assigned to a jet which is shown in figure 5.3(b). Each jet was given a different colour so that one can see to which jet the particle belongs. The same histogram is plotted after adding the ghost particles to the hard event and running the algorithm again (figure 5.3(b)). Thus it is quite easy now to imagine the jet area. The purely ghosted jets, which are also included in the picture, are executed later because of the cut in p_t.



(a) Event with hard particles (b) Event including the ghost particles

Figure 5.3: Energy of the particles in the η-ϕ-plane

But to run the algorithm over a reasonable size of events takes quite a long time. Therefore a grid size of 30 corresponding to the addition of 900 ghost particles was chosen to determine the jet area. In figure 5.4 the distributions of the jet area for the SiSCone and the k_t algorithm are shown. The distribution of the SiSCone jets is quite sharp with the maximum at a value of 1.1, whereas the distribution of the k_t jet-areas is rather flat and the maximum is shifted to a higher value. Again this may be explained by the fact that it is more likely for the k_t algorithm to find a particle, even within a larger distance, that may be combined to the jet, which is in case of the SiSCone algorithm more restricted by the cone-radius R

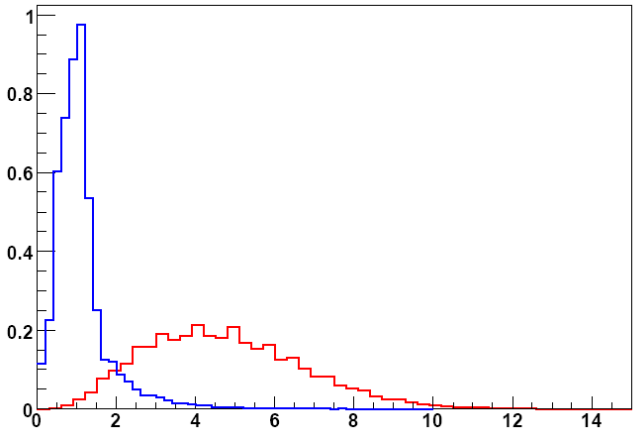


Figure 5.4: Area of the jets of the k_t (red) and SiSCone (blue) algorithm

It was also tried to determine the jet areas using a regular grid of ghost particles and a random grid with uniformly distributed ghost particles but the distribution of the jet area did not show any significant difference. However the computing time for the random grid seems to be smaller than for the regular, which has to be confirmed.

Furthermore it is observed, that with increasing parameter f (equation 2.3) of the split merge procedure, less jets with higher values of the area are occur (see figure 5.5). This can be explained by the fact that less jets get merged but again, further investigation should be carried out.

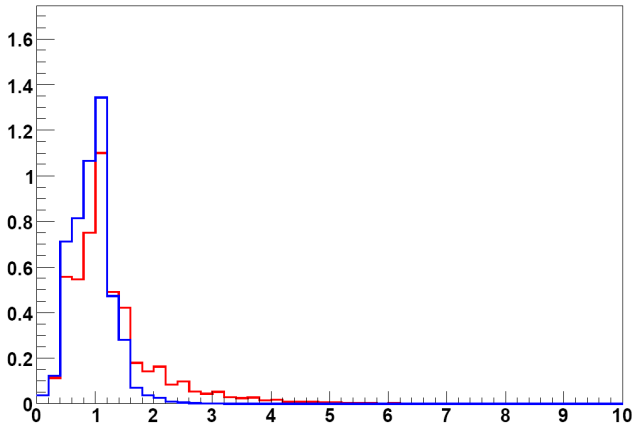


Figure 5.5: Area of the jets of the SiSCone algorithm with $f = 0.5$ (red) and $f = 0.7$ (blue)

5.3 Determination of the background level

In order to perform correction on the jets one has to determine the background level ρ that is a typical p_t of the background. Figure 5.6 shows the transverse momentum of the jets over the jet area versus the rapidity η taken form simulated pp-collisions at LHC.

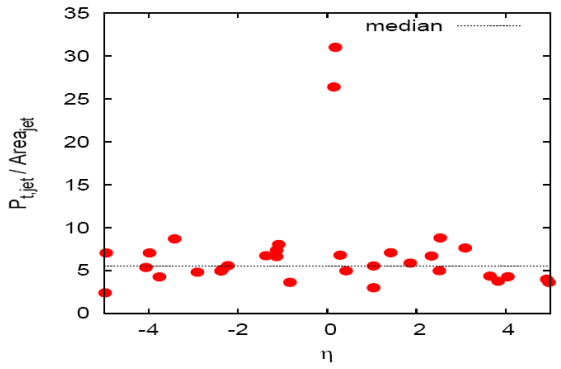


Figure 5.6: $\frac{p_{t,jet}}{\text{Area}_{jet}}$ for simulated pp collisions

It indicates that for the background jets the variable $\frac{p_{t,jet}}{\text{Area}_{jet}}$ lies within a band whereas for the hard jets this variable is far above the band.
One may exploit this different behavior of hard and background jets to determine

the noise level by calculating the median:

$$\rho = \text{median}\left[\left\{\frac{p_{t,i}}{A_i}\right\}\right] \tag{5.1}$$

of all jets i that lie within the band.

5.4 Correcting jets

The determined background level and jet area may now be used to perform corrections on the jets. Since the amount of background radiation clustered together with the jet is proportional to the jet-area A one might correct the transverse jet-momentum according to equation 5.2.

$$p_t^{(sub)} = p_t - \rho \cdot A \tag{5.2}$$

$p_t^{(sub)}$ is then the corrected transverse jet momentum according to the background. Of course the image drawn here is a quite simple one and if the jet-observable is sensitive to the jet direction one has to correct the full 4-vector. The jet area is then defined as the integral of a massless 4-vector over the surface of the jet.

Bibliography

[1] Stephen D. Ellis and Davison E. Soper, Phys. Rev. D **48** (Oct 1993) 7.

[2] Gavin P. Salam, arXiv:0705.2696v1 (May 2007).

[3] Gavin P. Salam and Grégory Soyez, JHEP **05**(2007) 086

[4] http://projects.hepforge.org/siscone/

[5] Matteo Cacciari and Gavin P. Salam, arXiv:0707.1378v1 (Jul 2007)

[6] Matteo Cacciari, arXiv:0706.2728v1 (Jun 2007)

Appendix: A

Schemes of the main algorithm and the algorithm protocones.cpp that determins the stable cones are shown in figure 5.7 and 5.8.

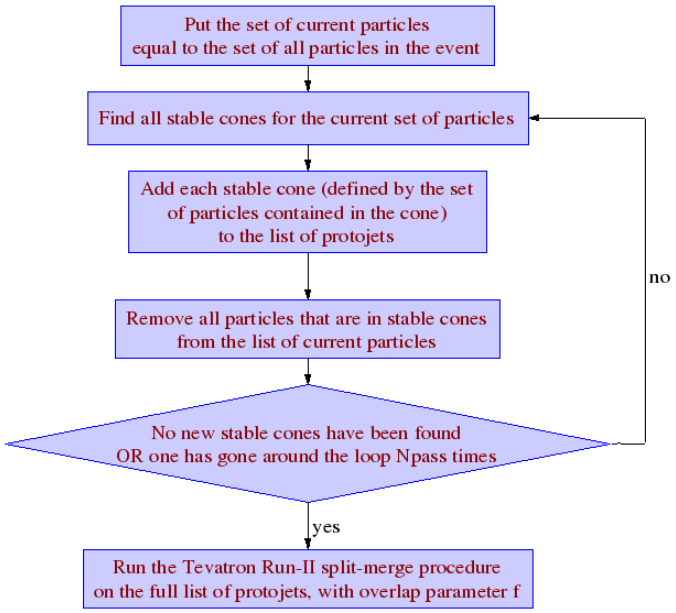


Figure 5.7: Sheme of the main algorithm

Parameters of the SiSCone jet algorithm

- R: The radius of the cone used to scan for stable cones in the member protocones.cpp (default: $R = 0.7$)
- f: The overlap parameter according to equation 2.3 (default: $f = 0.5$)
- n_pass_max: Maximum number of passes. After running the algorithm it may happen that some particles do not enter into a jet. The algorithm can therefore be rerun with those particles only. The parameter n_pass_max controls the number of these passes. (default: n_pass_max = 1, n_pass_max = 0 means infinity passes (until all particles are associated with a jet))
- p_tmin: Minimal p_t for the protojets in the split-merge process. At each step of the split-merge process, jet candidates which have a $p_t < p_{t,min}$ are removed. (default: $p_{t,min} = 0.$)

- split_merge_scale: The kind of recombination scheme used in the split-merge procedure: (default: SM_pttilde)
  - SM_pttilde: p-scheme p_t according to equation 2.2
  - SM_Et: transverse energy E_t
  - SM_mt: transverse mass m_t
  - SM_pt: transverse momentum p_t

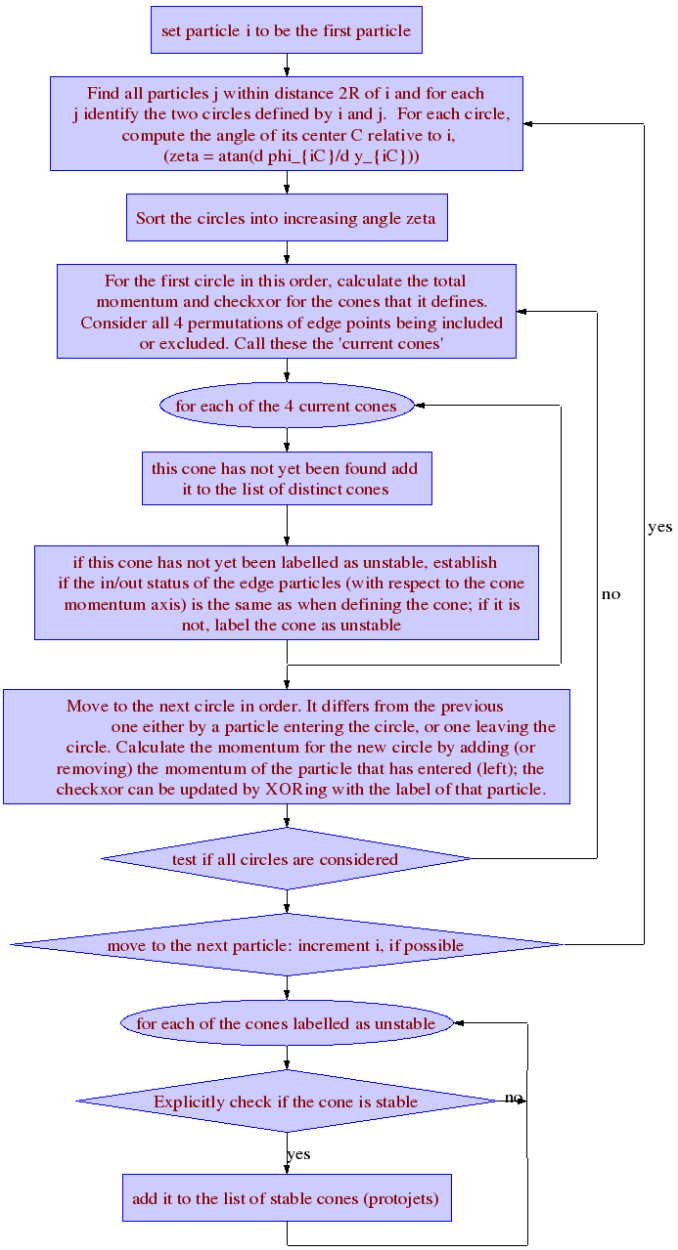


Figure 5.8: Sheme of the stable protojet determination (protocones.cpp)

Calling the program, Input and Output

To calculate the jets from a given set of particles the algorithm is called through the following member:
int Csisscone::compute_jets(vector<Cmomentum> &particles, double R, double f, int n_pass_max, double ptmin, split_merge_scale);
The returned integer value is the number of found jets.

The input of the algorithm is a list of particles stored in the form of a STL vector of the type Cmomentum. A particle of momentum p_x, p_y, p_z and energy E can be created by calling: **Cmomentum particle = Cmomentum(px, py, pz, E);** To update the particle list with all particles in the event one can just append one 4-momentum vector after another using the STL-methode: particles.push_back(Cmomentum(px, py, pz, E)); There are several methods of Cmomentum to calculate properties of the 4-momentum such as ϕ, η, ... (see momentum.h for details)

The result of calling compute_jets(...) or recompute_jets(...) is stored in the vector:
vector<Cjet> Csiscone::jets;
The elements of that vector contain all necessary information concerning the jets:

- Cjet::v: 4-momentum of the jet, again of the type Cmomentum
- Cjet::n: number of particles in the jet
- Cjet::contents: jet contents. It is stored as a vector<int> listing the indices of the particles contained in the jet according to the input particles

The second member recompute_jets allows to rerun the split merge algorithm with a different overlap parameter f with the following command:
int Csisscone::recompute_jets(double f, int n_pass_max, double ptmin, split_merge_scale);
Thus the list of protocones is not recalculated, only the split merge procedure is rerun which needs much less computation time.