MASTER OF SCIENCE THESIS

A new dipole-based jet clustering algorithm

Home Inst. Advisor:
Ivica PULJAK

Dept. of Mathematics and Physics
FESB, University of Split
Split, Croatia

Visiting Inst. Advisor:
Leif LÖNNBLAD

Dept. of Theoretical Physics
Lund University
Lund, Sweden

Author:
Toni ŠĆULAC

Physics Department
Prirodoslovno Matematički Fakultet
University of Split , Croatia

A thesis written within the ERASMUS exchange programme and submitted in fulfilment of the requirements for the Master's Degree

June 2014

Abstract

A new dipole-based jet clustering algorithm

by Toni ŠĆULAC

Jet production occurs very often in particle physics experiments, and a very good understanding of how partons evolve into jets has been achieved over the last 30 years [1]. The main tool in jet analysis is jet clustering algorithms, and because the problem of clustering particles back to initial partons can not be solved exactly, many different algorithms have been developed [2, 3]. In this work, we propose a new dipole-based jet clustering algorithm, called the dipole-k_t algorithm with two main features; it uses a Lorentz invariant distance measure, and it does $3 \to 2$ recombinations in an attempt to "invert" a dipole-based parton shower. We validate its exclusive version by comparing it to the k_t-algorithm. We then proceed to analyze the dipole-k_t results in W-production events in proton proton collisions, where the W has a large transverse momentum and decays into jets. A simple analysis of the W mass reconstruction strengthens our hope that with future developments this area of work will become dipole-k_t algorithm's main forte.

Acknowledgements

Here I would like to thank my visiting institution supervisor Leif Lönnblad who decided to share his idea for a new algorithm with me. I would mostly like to thank him for being a great supervisor who was always there for me, and who made this work very easy with his sense of humour. I would like to thank my home institution supervisor Ivica Puljak for giving me great advice during this period. Also, I would like to thank Torbjörn Sjöstrand for his, much appreciated, help with PYTHIA and discussions he had with Leif and me. Last but not least I would like to thank my family whose unconditional love and support helped me got this far in my studies.

Contents

Chapter 1

Introduction

A jet is a cluster of hadrons produced by hadronization of a quark or a gluon in a particle physics experiment (eg. proton-proton collisions). We use Quantum Chromo-Dynamics (QCD) to describe jet dynamics, because they consist of hadrons that interact mainly by the strong force. Jets have long been studied in particle physics experiments and a very good understanding of their dynamics has been achieved [1]. However, with the introduction of the Large Hadron Collider (LHC) at the European Organization for Nuclear Research (CERN) which produces jets in much larger numbers than previously, and over a huge kinematic range, a deeper understanding of jets became necessary. Because of the large number of particles produced in every collision, computers are needed to collect and analyze the data in particle physics experiments.

Jet production occurs in most high energy collisions of particles. Quarks and gluons that are produced in collisions cannot live as a stand-alone particle, and thus undergo a process of hadronization that turns them into hadrons. Hadrons are bound states of quarks and gluons that can be detected in the particle physics experiments. Because of this, jet finding algorithms are necessary for basically any kind of a particle collision analysis. This has led to the development of a several jet finding algorithms in the last 30 years [2]. One can think of a jet algorithm as an attempt to rewind from the final state particles in our data to the initial parton (quark or gluon) that produced that particular jet, step by step. This task is very tricky when there are a large number of final state particles, so it is impossible to have an algorithm that will give exact results. That is why several jets clustering algorithms are used these days, and each one of them has some advantages and disadvantages. In reviews [2, 3] of these algorithms one can often find that choosing which algorithm to use depends on what one analyzes.

One active area of the research of the jet clustering algorithms is the so-called "boosted jets analysis". This is the analysis of the jets created from a heavy particle with high transverse momentum (momentum perpendicular to the beam axis). Since we have reached so high energies in proton proton collisions at the LHC, it is possible to create heavy particles, like the Z or W bosons, with transverse momenta of several hundred GeV. This means when they decay further they can produce boosted jets that are almost collinear and therefore hard to differentiate. That is why substructure of boosted jets is a challenge for jet clustering algorithms. Current analysis are done by first finding one fat jet, and then using the algorithm again on that fat jet (but with other parameters) to try to distinguish its substructure (described in detail in refs. [1, 4]).

What jet clustering algorithms used these days have in common, is that they are not Lorentz invariant under transverse boosts and that they do clusterings by clustering two particles back to one ($2 \to 1$). Our aim was to develop a jet clustering algorithm that is different with the respect to that. Firstly, it would use Lorentz invariant variables for calculations, and therefore should do well in a boosted jet case. Secondly, it would cluster three particles to two ($3 \to 2$), as an idea of inverting a dipole-based parton shower evolution. Parton showers are Monte Carlo simulations of the evolution of quarks and gluons produced in particle collisions by successive emissions of further quarks and gluons. In recent times most of them have adopted a color dipole model that does emissions of partons from a dipole rather than from a single particle (does $2 \to 3$ emissions rather than $1 \to 2$)[5]. The main idea of our algorithm is to try and "invert" these emissions.

In this thesis we present the reader with a theoretical background, development and validation of our new algorithm, and finally with a preliminary analysis of the prospects of the algorithm in a boosted jets analysis.

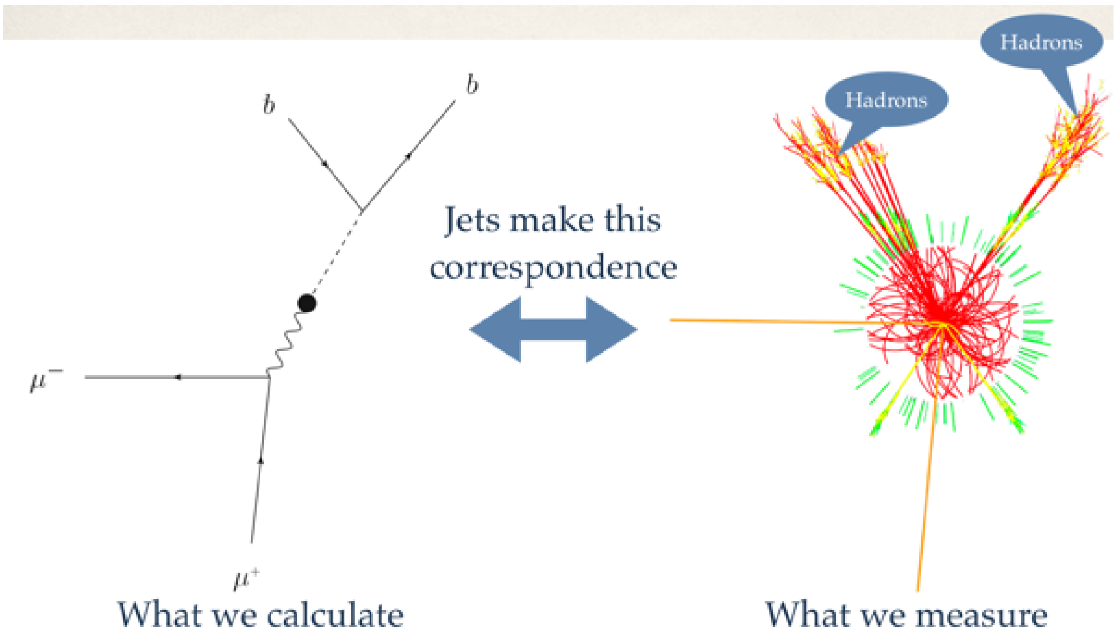


Figure 1.1: Illustration showing a connection between a theoretical view on the Higgs production (shown with a schematic diagram on the left side) and what we actually measure in our detectors (shown with an illustration on a right side).

Chapter 2

Theory

2.1 Quantum Chromodynamics (QCD)

The strong force was first introduced to explain the stability of the atomic nuclei. Now we know that protons and neutrons, which make up the atomic nuclei, are not elementary particles. They are made of three quarks each, and quarks interact by the exchange of gluons, another elementary particle. In total we have found six different flavors (types) of quarks arranged in three generations:

- up (u), down (d)
- charm (c), strange(s)
- top (t), bottom (b).

Everything we see is made only from the first generation quarks: up (u) and down (d). Each quark has spin $\frac{1}{2}$. u, c and t quark have electric charge $+\frac{2}{3}$, d, s and b quark have -$\frac{1}{3}$. A proton is made of two up and one down quark (uud), and neutron from one up and two down quarks (udd). Just like electric charge defines how particles interact via the electromagnetic force, quarks have color (QCD charge) that specifies how they interact via the strong force. Each quark is either "blue", "green" or "red". In addition, each quark has a partner antiquark that has the same mass but opposite charge, and has anti-color. In total there are 36 quarks, that interact by exchange of 8 different types of gluons. Gluons are massless color-charged particles with spin 1.

Just like the electric charge, the color charge is conserved. In addition an interesting phenomenon called color confinement occurs, stating that color-charged particles cannot be isolated, and therefore cannot be directly observed. Because of the color confinement we can only detect quarks as bound inside mesons and baryons. Mesons are particles composed of one quark and one antiquark that have opposite color (eg. "blue" quark and "anti-blue" antiquark) and are therefore colorless. Baryons are composed of three quarks, each with different color, and are also colorless. Baryons and mesons are together called hadrons. Because of the color confinement, quarks and gluons fragment into hadrons and form jets. This process is called hadronization and it occurs frequently in particle collision experiments, such as the LHC. These hadrons that form jets are measured in detectors and studied in order to determine the properties of the original interactions between quarks or gluons.

2.2 Jets

Jets are used for a wide range of physics analyses. One way of classifying their use is according to the different possible origins for the partons that give rise to the jets. In hadron colliders processes, the energy and direction of the jet is closely related to that of the parton in the proton that underwent a hard scattering. So jet spectra contain information on the distribution of partons inside the proton [6–9]. Jets can also originate from partons produced in decays of heavy particles, such as the Higgs or W,Z. The analysis of these jets can then be a way of detecting that particle. The sum of the momenta of the jets should have an invariant mass that is close to the mass of the heavy particle [10]. Jets may also originate radiatively, for example from the emission of a gluon off some other parton in the event. The rate of a production of such jets provides an information on the value of the strong coupling (for example refs. [11, 12]).

Given the variety of these and other related possible uses of jets, it is not surprising that there is no single optimal way of defining jets, and over the 30 years that have passed since the first detailed proposal for measuring jets [13], many jet definitions have been developed and used. In this section reader will be provided with some widely used jet definitions necessary for further discussions.

2.2.1 Jet variables

In order to study jets one must define some variables used to describe them. In proton proton collisions, a natural thing is to define jets transverse momentum ($p_\perp$) and their direction (η,ϕ).

- **Jet $\boldsymbol{p_\perp}$** is the transverse momentum of the jet defined as a sum of the transverse momentum of all particles contained in a jet.

$$\boldsymbol{p_{\perp Jet}} = \sum_{i \in Jet} \boldsymbol{p_{\perp i}} \tag{2.1}$$

- **Jet pseudorapidity** is the coordinate describing the angle of a jet relative to the beam axis. It is defined as:

$$\eta_{Jet} = -ln\left[tan\left(\frac{\theta}{2}\right)\right] \tag{2.2}$$

  where θ is the angle between the jet momentum and the beam axis. Note that, in the approximation that the mass of the particle is nearly zero, the pseudorapidity of the jet is numerically close to the experimantal particle physicist's definition of the rapidity $y = \frac{1}{2} ln \frac{E+p_z}{E-p_z}$.

- **Jet azimuth** is the angle of the jet transverse momentum with the respect to the x-axis:

$$\phi_{Jet} = \begin{cases} arctan\frac{p_y}{p_x} & ; x > 0 \\ arctan\frac{p_y}{p_x} + \pi & ; y \geq 0, x < 0 \\ arctan\frac{p_y}{p_x} - \pi & ; y < 0, x < 0 \\ +\frac{\pi}{2} & ; y > 0, x = 0 \\ -\frac{\pi}{2} & ; y < 0, x = 0 \\ \text{undefined} & ; y = 0, x = 0 \end{cases} \tag{2.3}$$

- **Jet mass** squared is given with

$$m^2 = \left(\sum_{i \in Jet} \boldsymbol{p_i}\right)^2 \tag{2.4}$$

  where $\mathbf{p}_i$ is the 4-momentum of a particle i contained in a jet.

- **Jet radius**. The dimensionless parameter that tells us about the size of the cone in which all of the jet particles are contained. It is defined as the minimum value of R for which the inequality:

$$(\eta_i - \eta_{Jet})^2 + (\phi_i - \phi_{Jet})^2 < R^2 \tag{2.5}$$

  holds for each particle i. Here, η_i and ϕ_i are respectively the pseudorapidity and azimuth of particle i.

2.2.2 Matrix element calculations

In order to understand jets and their production one must first get familiar with basic scattering calculations in particle physics. Particle physics experiments count the number of occurrences of specific events, which is directly related to the "cross section" of the relevant process to happen. We can calculate cross sections from our theories to have a prediction for the experiment.

The differential cross section of a proton A and proton B to inelastically scatter and produce final state particles $c_1 \ldots c_n$ and proton remnants X, is given by

$$d\sigma(A+B \to c_1 + \cdots + c_n + X) = \sum_{a,b \in \{allowedpartons\}} \int_0^1 dx_b \int_0^1 dx_a f_{a/A}(x_a, \mu_F) f_{b/B}(x_b, \mu_F) d\hat{\sigma}(a+b \to c_1 + \cdots + c_n). \tag{2.6}$$

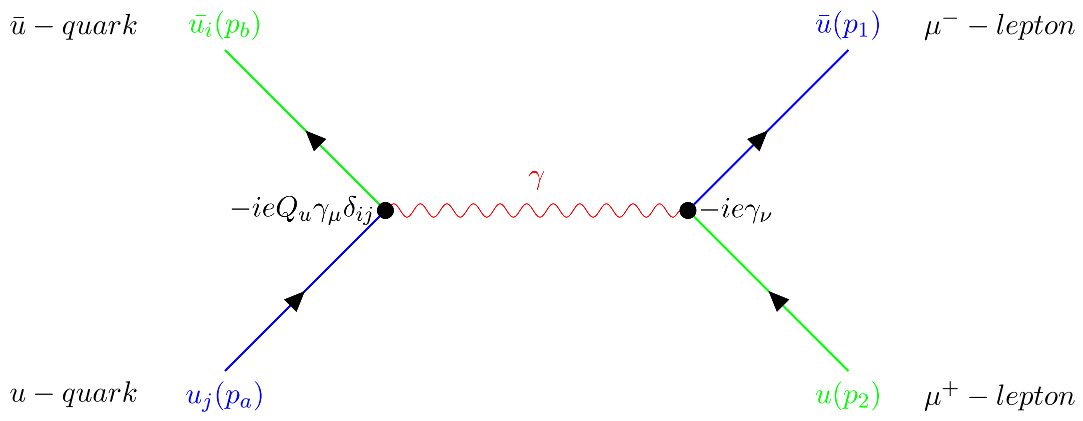
Here $f_{a/A}(x_a, \mu_F)$ and $f_{b/B}(x_b, \mu_F)$ are functions describing the distribution of partons inside the proton, the parton distribution functions (PDFs). They cannot be calculated from theory and are therefore measured in experiments. $d\hat{\sigma}(a+b \to c_1 + \cdots + c_n)$ is the differential partonic cross section of partons a and b scattering and producing final state particles $c_1 \ldots c_n$. It can be calculated from theory and it is given by

$$d\hat{\sigma}(a+b \to c_1 + \cdots + c_n) = \frac{1}{4\sqrt{(p_a p_b)^2 - M_a^2 M_b^2}} |\mathcal{M}(a+b \to c_1 + \cdots + c_n)|^2 d\phi_n(p_a, p_b; p_1, \ldots, p_n). \tag{2.7}$$

The first factor $(1/[4\sqrt{(p_a p_b)^2 - M_a^2 M_b^2}])$ is called a flux factor, and is related to the relative velocities and energies of the incoming partons. The factor $|\mathcal{M}|^2$ is called squared invariant **matrix element** (ME). The flux factor and ME are finally integrated over the phase space differential

$$d\phi_n(p_a, p_b; p_1, \ldots, p_n) = (2\pi)^4 \delta^{(4)} \left(p_a + p_b - \sum_{i=1}^{n} p_i \right) \frac{d^3 p_1}{(2\pi)^3 2E_{p_1}} \cdots \frac{d^3 p_n}{(2\pi)^3 2E_{p_n}}. \tag{2.8}$$

The $\delta^{(4)}$-functions guarantee conservation of the 4-momentum, and the integration measures $d^3 p_i/(2\pi)^3 2E_{p_i}$ are chosen to make the phase space integrations Lorentz invariant. We proceed by describing how to determine the invariant ME for a specific case. The basic building blocks of the invariant ME are *vertex functions, propagators* and *wave functions for external particles.* Vertices describe interactions of particles with the force



$$\mathcal{M} = -e^2 Q_u \bar{u}(p_1)\gamma_\nu u(p2) \frac{-g^{\mu\nu}}{(p_a+p_b)^2+i\epsilon} \bar{u}_i(p_b)\gamma_\mu \delta_{ij} u_j(p_a)$$

FIGURE 2.1: Matrix element for a scattering $\bar{u}(p_b)u(p_a) \rightarrow \mu^+(p_2)\mu^-(p_1)$. The subscripts i and j indicate the color of the quarks. The upper part shows Feynman diagram, with vertices in black, photon propagator in red, and the external wave functions in green and blue. The corresponding factors in the ME are indicated with the same colors. The lower part shows the actual ME expression, which is read off the Feynman diagram by moving from the right to the left side and multiplying all encountered expressions.

field by exchange of virtual particles, while propagators describe the propagation of virtual interacting fields. The perturbative approximation of these building blocks can be derived from the Lagrange density. Once propagators and vertexes have been derived, the calculation of matrix elements can be done using a pictorial representation called Feynman diagrams. As an example the matrix element $\mathcal{M}(\bar{u}u \rightarrow \mu^+\mu^-)$ is shown in fig.2.1. Calculations with Feynman diagrams become inconvenient when many final state particles are involved, since the number of possible graphs grows rapidly. It is possible to derive an approximation of matrix elements with additional final state partons (in particle physics, a parton is a collective name for quarks and gluons). With the introduction of *splitting functions* it is possible to calculate approximations of the invariant ME for an arbitrary number of quark or gluon emissions(in the QCD case).

Knowing those, one can then construct a computer program that provides an approximation of the cross section for any number of collinear partons. Event generators, which are studied in 2.2.3, rely on such approximations.

2.2.3 Event generators

We have found that the cross section for processes containing an additional splitting approximately factorizes into a hard cross section and process-independent splitting functions. Using these splitting functions and multiplying them with a probability that no emission will occur above some chosen $p_\perp$, one can then calculate the probability for the hardest emission to occur (this is described in more detail in section 3.1 where we introduce the PYTHIA parton shower). Further analysis show that such calculations always have a least collinear particle given by the hardest emission off the hard core process. Knowing a way to calculate the probability for the hardest emission provides an insight on how to simulate such process. Since all further emissions cannot be harder, we can for example order them by transverse momentum ($p_\perp$). The hardest emission will be followed by a second hardest emission, and so forth. Now one can derive an algorithm to generate a final state with an arbitrary number of gluons emitted from an initial quark:

Choose a starting scale $p_{\perp max}$, typically given by the $p_\perp$ of the initial quark

1. Choose $p_\perp < p_{\perp 0}$ (=$p_{\perp max}$ in the first step) and an energy fraction z accordingly.
2. Decide if the emission should be generated. If so, construct the emission momentum.
3. Reset the starting scale $p_{\perp 0} \to p_\perp$ and start from 1.
4. Repeat steps 1. - 3. if $p_{\perp 0}$ is above some cut-off $p_\perp$. Otherwise stop.

This algorithm is the foundation of parton shower (PS) programs. Parton showers are a core component of wider simulation frameworks called "Monte Carlo event generators". Event generators aim at describing particle collisions in detail. The main process simulated in the collision is called a hard subprocess. The emission of a particle from the incoming partons is called initial state radiation (ISR). Similarly, the emission of a particle from outgoing partons is called final state radiation (FSR). In a realistic collision the probability to have only hard subprocess is very small, because a lot of other partons interact. We call all of these background interactions multiple parton interactions (MPI) and they are also simulated in the event generator. Event generation is done by interfacing the parton showered hard process with methods that

- Add multiple interactions between the incoming beam particles.
- Decay unstable resonance particles produced in the hard interaction.

- Hadronize color-charged partons into un-colored hadrons.
- Decay unstable hadrons.

The implementation of these tasks varies between the most popular projects HERWIG, PYTHIA and SHERPA, and their current incarnations HERWIG++ [14], PYTHIA8 [15] and SHERPA[16]. A review of current event generators for the LHC physics can be found in [17].

2.3 Jet clustering algorithms

If we know how to simulate emissions and production of new particles and jets, the question arises if it is possible to rewind the action from the final state particles to the initial state partons. More precisely, is it possible to write an algorithm that clusters final state particles backwards to the initial state partons with the exact 4-momentum they had just after the collision (or decay)? Of course, it is not possible to obtain the exact 4-momentum because of the approximations one has to make, as it is often case in physics. Nevertheless, several approximative algorithms have been developed over the years.

A jet algorithm, together with its parameters and a recombination scheme forms a **jet definition**. Some general properties of jet definitions have been set in 1990 [18] by a group of influential theorists and experimental, and reads as follows:

Several important properties that should be met by a jet definition are:

1. *Simple to implement in an experimental analysis.*
2. *Simple to implement in the theoretical calculation.*
3. *Defined at any order of perturbation theory.*
4. *Yields finite cross sections at any order of perturbation theory.*
5. *Yields a cross section that is relatively insensitive to hadronization.*

There are currently many hadron-collider jet algorithms in use, some dating back to the 80's. The purpose of this section is to give a short overview of some main algorithms (for detailed review see ref. [3]).

2.3.1 Cone algorithms

The first ever jet algorithm was developed by Sterman and Weinberg in the 1970's [13]. It was intended for e^+e^- collisions and classified an event as having two jets if at least a fraction $1-\epsilon$ of the event's energy was contained in two cones of opening half-angle δ (and hence is known as cone algorithm). This idea of finding particles within some cone was the basic idea for many algorithms to come. The only differences were how to find the direction of the cone (seed particle) and what parameters should be used. Below, some of these cone algorithms will be briefly described.

Iterative cones (IC) is an algorithm in which a seed particle i sets some initial direction. One then sums the 4-momentum of all particles j such that

$$\Delta R_{ij}^2 = (\eta_i - \eta_j)^2 + (\phi_i - \phi_j)^2 < R^2 \tag{2.9}$$

where η_i and ϕ_i are the pseudorapidity and azimuth angle of particle i, respectively. The direction of the resulting sum is then used as a new seed direction and is iterated until the direction of the resulting cone is stable. There are two problems that arise in this kind of algorithm. Firstly, what should be taken as the seed, and secondly what if two cones overlap.

One solution for the overlap problem was to take as a first seed particle the one that has the largest transverse momentum. Once a stable cone is found it is called a jet and removed from the event. Then a new seed is the hardest particle among the remaining ones. This procedure is repeated until there are no particles left. This avoids any issue of overlapping. A possible name for such algorithm is "Iterative cone with progressive removal of particles". Another approach to the issue of the same particle appearing in many cones is to find all stable cones first and then decide what to do with particles that are part of more than one cone. The split merge approach merges a pair of cones if more than a fraction f of the softer cone's transverse momentum is in particles shared with the harder cone; otherwise the shared particles are assigned to the cone to which they are closer.

With the development of these algorithms question of their **infrared and collinear safety** (IRC) arise. IRC is the property that if one modifies an event by a collinear splitting or the addition of a soft emission, the set of hard jets that are found in the event should remain unchanged. IRC problem gave rise to the **exact seedless cones** (SC) algorithm. This type of algorithm avoids the use of seeds and iterations, and instead finds all stable cones. One takes all subsets of particles and establishes for each one weather it corresponds to a stable cone. The SC algorithm can only cluster few particles because the total time for clustering is $\mathcal{O}\left(2^N\right)$ (it would take about 10^{17}

years to cluster 100 particles). This exponential time behavior made SC impractical for use on events with realistic number of particles. However in 2007 a polynomial-time geometrically based solution was found to the problem of identifying all stable cones [19]. The algorithm is known as **seedless infrared safe cone algorithm**(SISCone), and is described more in the review in ref. [3].

2.3.2 Sequential recombination jet algorithms

Just like cone algorithms, sequential recombination jet algorithms were first considered for e^+e^- experiments. But with small changes, some of the algorithms can be used in hadron collider experiments. A detailed review of this is given in refs. [1, 2]. Many of the ideas underlying today's sequential recombination algorithms appeared first in the LUCLUS algorithm of Sjöstrand [20]. Because of the computational constrains at the time, the algorithm included a preclustering phase and a non-trivial procedure of reassignment of particles between clusters at each recombination step, making it rather complicated. Today's algorithms are all rather simple, as can be seen below.

The Jade algorithm was the first simple sequential recombination algorithm, introduced by the JADE collaboration in the middle of 1980's [21, 22]. For each pair of particles *i, j*, it calculates the distance d_{ij} as a ratio of the squared invariant mass of two particles and the squared invariant mass of the final state. It then finds the minimum d_{min} of all the d_{ij}, and if d_{min} is below some d_{cut} it recombines i and j into a single cluster. This procedure is repeated either until a wanted number of jets is obtained, or all of the obtained jets are above some d_{cut}. It is easy to see that the number of jets that one obtains depends on the value of d_{cut}. So here the number of jets is controlled by a single parameter rather than the two parameters (energy and angle) of cone algorithms.

The Jade algorithm, and many others, were introduced for e^+e^- collisions, so a question arises how can one modify such algorithms for the experiments with the incoming hadrons? There are two problems that need to be considered. Firstly, the total energy is no longer relevant (most of the energy goes to jets that are in the beam direction), so instead of the dimensionless distance one might choose to use a dimensionfull distance. Secondly, QCD branching probabilities are not just between pairs of outgoing particles, but also between an outgoing particle and the incoming beam direction.

The algorithms given in the following were first developed for e^+e^- collisions, but were later modified for hadron collisions.

The $\mathbf{k_t}$-algorithm is formulated as follows:

1. For each pair of particles i, j work out the distance

$$d_{ij} = min(p_{\perp i}^2, p_{\perp j}^2)\frac{\Delta R_{ij}^2}{R^2}, \quad \Delta R_{ij}^2 = (\eta_i - \eta_j)^2 + (\phi_i - \phi_j)^2, \tag{2.10}$$

also define a distance to the beam

$$d_{iB} = p_{\perp i}^2 \tag{2.11}$$

where η_i and ϕ_i are the pseudorapidity and azimuth angle of particle i respectively, and R is usually called jet radius and determines its angular reach.

2. Find the minimum between all d_{ij} and d_{iB}.

3. If this is d_{ij}, recombine i and j into a single new particle and return to step 1.

4. Otherwise, if the minimum distance is a d_{iB}, declare i as jet and remove it from the list of particles. Return to step 1.

5. Stop when no particles remain or $d_{min} > d_{cut}$, or if we have reached wanted number of clusters.

In the collinear limit, the distance measure (eq. 2.10) reduces to squared transverse momentum, this is the origin of the name k_t-algorithm. Since $(\eta_i - \eta_j)$, the ϕ_i and $p_{\perp i}$ are all Lorentz invariant under longitudinal boosts, the d_{ij} and d_{iB} are too. The k_t-algorithm has long been advocated by theorists because it is free of any IRC safety issues. On the other hand it has been criticized by experimenters on the grounds that it was computationally slow, and that it produces geometrically irregular jets. Speed issue has been solved by the FastJet [24] implementation. It was done by reformulating the search for the minimum distance as a geometrical problem. It achieved an $\mathcal{O}\left(NlnN\right)$ timing, meaning clustering of 1000 particles now takes only a few milliseconds.

The Cambridge and Aachen algorithm (C/A) is similar to the k_t-algorithm. The difference is that it does not have a distance to the beam defined. It also uses longitudinally Lorentz invariant variables and introduces an R parameter defined in eq. 2.5. It proceeds by recombining the pair of particles with the smallest ΔR_{ij} in each step, and repeating the procedure until all objects are separated by a $\Delta R_{ij} > R$. Final objects are then declared as jets. This clustering procedure is energy independent, meaning there is no ordering in clustering between hard and soft particles.

The anti-k_t algorithm is also similar to the k_t-algorithm. One can deduce it from the k_t-algorithm by replacing $p_{\perp_i}^2$ and $p_{\perp_j}^2$ with $p_{\perp_i}^{-2}$ and $p_{\perp_j}^{-2}$. The only difference is that, because of different measure d_{ij} definition, it clusters hard particles first, rather than soft ones. This ultimately means that the jets grow outwards around hard seeds. The result is an IRC safe algorithm that gives circular shaped jets, making it an attractive replacement for cone-type algorithms.

So far we have not discussed how to combine the momenta when we do the clustering (what is the recombination scheme). The most common recombination scheme used in step 3. of the k_t-algorithm is the so-called E-scheme.[1] This scheme simply adds the 4-momentum of particles, giving massive jets as a result.

As we have seen in the section 2.2.3 parton shower emission simulations can be done doing $1 \to 2$ branchings, and one can see the k_t-algorithm as an attempt to "invert" those branchings. Doing $2 \to 1$ clusterings with an E-scheme results in massive jets, and we know that the initial partons that produce jets are considered to be massless. On the other hand, if we choose some other scheme we could get massless jets, but would then violate the 4-momentum conservation.

The color dipole model was developed at Lund, as an idea that the emissions of additional gluons (or quarks) can be simulated as $2 \to 3$ splittings (firstly implemented in ARIADNE [23]). The idea is that the emitter and recoiler particle form a dipole that then emits a new particle. This has shown good results in simulations, and similar color dipole models are now used in most event generators of the popular projects (HERWIG++, PYTHIA8, SHERPA).

$3 \to 2$ **recombination**. Most sequential recombination algorithms use the idea of inverting successive $1 \to 2$ branchings. When simulating QCD branching it can be also useful to consider dipole branchings, ie. $2 \to 3$ splittings, as in ARIADNE. One can imagine that a $3 \to 2$ recombination scheme could result in massless jets without violating the 4-momentum conservation.

2.3.3 Boosted jets

There are several reasons to be interested in boosted jets. Searching for High mass resonances is the simplest example. For instance, a heavy resonance R_C with mass $m_R \geq 1.5TeV$ which decays to a pair of gauge bosons would yield highly boosted VV pairs (V=Z^0,W^+,W^-) which would then produce boosted (and almost collinear) jets. This is why new jet-reconstruction techniques are developed, and a great interest is shown in understanding the jet substructure. To see how this jet reconstruction techniques work we will analyze the case of a boosted Higgs decaying into $b\bar{b}$ (ref. [4]).

[1]default recombination scheme in FASTJET, but it offers more recombination schemes that can be used instead, such as: E-scheme, p_t-scheme, p_t^2-scheme, E_t-scheme, ...

To be specific, we will consider $pp \to HZ^0$ followed by $H \to b\bar{b}$ and $Z^0 \to l^+l^-$. The approach is to focus on events where the Higgs is produced with $p_{\perp H} > 200GeV$, and cluster these events with a large jet radius (R=1.2), such that all Higgs decay products are swept up in a single fat jet. The Higgs mass then should be the invariant mass of that fat jet. The signal we search for now is a leptonic Z^0 + fat Higgs jet, and the background to this signal is Z^0 + one fat "Higss-like" jet. It can be shown that there are two ways of telling that there was a Higgs boson decay rather than a fat jet from the QCD parton branching[2]. Two jets from the Higgs should be almost symmetric(have approximately the same p_t), and the mass of each jet should be much smaller than the mass of the Higgs. The algorithm for finding boosted Higgs can now be constructed as following:

- Cluster the event on a large angular scale, using C/A algorithm.
- Now, given a hard fat jet, successively unwind the jet by undoing the cluster sequence one branching at the time. At each branching $P \to ij$, check to see weather the splitting looks sufficiently non-QCD-like, by asking that the branching should be both hard,
$$max(m_i, m_j) < \mu m_H \tag{2.12}$$
for some parameter μ, and symmetric,
$$d_{ij} > d_{cut} \tag{2.13}$$
for some choice of d_{cut}.
- If the splitting fails to be sufficiently hard and symmetric, discard the softer of i and j, and continue to unwind
- Continue until either an interesting splitting has been found or there are no more jets left

This procedure is often referred as the "splitting" or "mass-drop" procedure. It identifies an interesting Higgs-like splitting $H \to b\bar{b}$. After the Higgs-like splitting has been identified, if one wishes to investigate the fat jet substructure new jet analysis is needed, this time on a smaller scale. One should have in mind that by obtaining one fat jet a lot of background radiation is included. That is why procedures such as **filtering**, **trimming**, **pruning** and **grooming** were introduced (ref. [3]). They all focus on "cleaning up" fat jets by subtracting the contributions of unassociated radiation.

[2]Note here that there are several ways to determine if jets are produced by the b quarks, with so-called b-tagging methods. But keep in mind that b quarks can appear in jets from the QCD parton branchings as well.

From this example it is easy to see why boosted jet analysis area is so active these days. This led us towards the attempt of constructing the jet finding algorithm that could identify boosted jets without the need of finding a fat jet first.

Chapter 3

Dipole-k_t jet clustering algorithm

With everything said in the previous sections, we were inspired to try and write a new jet clustering algorithm. The key features that we look to include are the $3 \to 2$ dipole-based recombination scheme and the Lorentz invariant distance measure. The main idea of our algorithm was to "invert" the dipole-based PYTHIA parton shower using its invariant transverse momentum as the distance measure, hence the name Dipole-k_t. In this section we first present the reader with the PYTHIA dipole shower framework. We then present our Lorentz invariant distance measure and give the summary of how the algorithm works. In the end algorithm is tested in the comparison with the k_t-algorithm.

3.1 PYTHIA dipole showers

Here, a short overview of the dipole-based PYTHIA parton shower (DPS) will be given. For more details we refer to [25]. Two key features of the DPS, that we are interested in, are $2 \to 3$ dipole splittings and transverse-momentum ordering.

The DPS is ordered in the transverse momentum ($p_\perp$), meaning that the initial-state radiation (ISR) and the final-state radiation (FSR) are both ordered in the $p_\perp$. If we were to look the probability for emission without any ordering it would diverge for the small $p_\perp$. In order to get rid of this divergence, one has to include the Sudakov form factor which gives the probability that no emission above some $p_\perp$ will happen. This now gives the probability distribution for the "next" emission:

$$\begin{aligned}\frac{dP}{dp_\perp} &= \left(\sum \frac{d\mathcal{P}_{ISR}}{dp_\perp} + \sum \frac{d\mathcal{P}_{FSR}}{dp_\perp}\right) \\ &\times exp\left(-\int_{p_\perp}^{p_{\perp max}} \left(\sum \frac{d\mathcal{P}_{ISR}}{dp'_\perp} + \sum \frac{d\mathcal{P}_{FSR}}{dp'_\perp}\right) dp'_\perp\right) \qquad (3.1)\end{aligned}$$

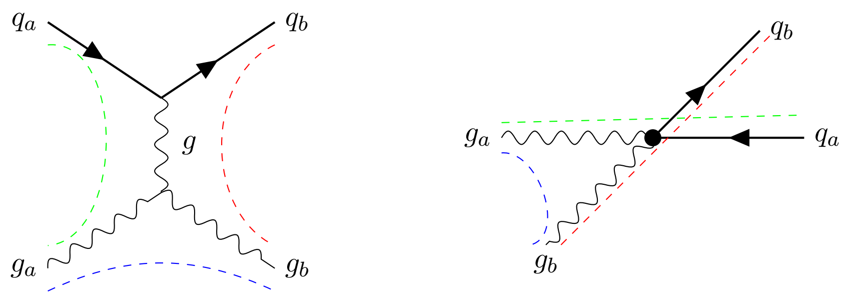


FIGURE 3.1: One possible Feynman graph and color flow for the $qg \to qg$ process (on the left), and the resulting color dipoles between the scattered partons and the beam remnants in the momentum representation(on the right)

Here the sum runs over all incoming (ISR) and outgoing (FSR) partons and over all possible splittings described with the inclusive probability $\mathcal{P}$. $p_{\perp max}$ is the $p_\perp$ of the previous step. Starting from a single hard interaction, eq. 3.1 can be used repeatedly to construct a complete parton-level of an arbitrary complexity. This only gives information on which radiation will occur and at what $p_\perp$, but one still has to define how such radiation will be described and how its kinematics will be done.

To show the motivation for the DPS let us consider a $qg \to qg$ process with one color-anticolor being annihilated, as illustrated on Fig.3.1. One can easily see that there are three color dipoles. By the nature of the endpoints, they can be classified as final-final (FF), final-initial (FI) or initial-initial (II). Because of the color connection between the partons, whenever a quark or a gluon emit a parton the momentum of all dipoles that they belong to will be modified. That means, if for example a parton is emitted from an initial quark, the final-state particles will be boosted. One can now ask oneself how one can distribute the recoil of an emission between the two endpoints of a dipole.

There is no unique prescription on how to do this. One possible choice is to view the emission as being associated with one of the two endpoint partons, the emitter, while the other parton acts as a recoiler. If viewed in the rest frame of an FF dipole, the recoiler does not change direction by the emission, but obtains a reduced absolute momentum such that the overall four-momentum of the dipole is preserved during the emission. The emitter and the emitted parton obtain opposite $p_\perp$ kicks by the branching, but the distribution in azimuthal angle is normally taken to be isotropic. All partons, both before and after the emission, are put on the mass shell. With this strategy, the radiation of a dipole is split into that of its two endpoints. This is done in a smooth manner, such that the radiation close to the either endpoint is associated with that parton radiating, whereas the large-angle radiation is split between the two in reasonable proportions.

The current DPS order emissions in terms of a $p_\perp^2$ evolution variable, with an additional energy-sharing variable z in the branching. For the QCD emissions the DGLAP evolution equation lead to the probability for the splitting of a parton $a \to bc$

$$d\mathcal{P}_a = \frac{dp_\perp^2}{p_\perp^2} \sum_{b,c} \frac{\alpha_s(p_\perp^2)}{2\pi} P_{a\to bc}(z)dz, \tag{3.2}$$

where $P_{a\to bc}$ are the DGLAP splitting kernels. The probability diverges for small values of $p_\perp$, meaning that infinite number of soft radiations is allowed. This inclusive quantity can be turn to an exclusive one by multiplying it with the Sudakov form factor. It gives the probability that there is no emission above the chosen $p_\perp^2$, and therefore secures ordering in the transverse momentum. For the ISR the parton distribution functions, reflecting the contents of the incoming hadron, must be taken into account as well.

$p_\perp^2$ is used both for the ISR and FSR in PYTHIA, but the $p_\perp^2$ definition is different in each case. We continue by giving more details about the ISR and the FSR.

ISR. For the ISR, the radiating dipole is always chosen such that the recoiler is the incoming parton from the other side of the subcollision. Thus all outgoing particles from the hard subprocess share the recoil of a new emission and the momentum of the whole system is boosted (the relative momentum of particles with the respect to each other remains the same). The evolution variable is defined as:

$$p_{\perp evol}^2 = -(1-z)Q^2 \tag{3.3}$$
$$\text{with } m_b^2 = -Q^2 \text{ and } m_a^2 = m_c^2 = 0$$

The z definition in eq. 3.3 is chosen to be

$$z = \frac{m_{br}^2}{\hat{s}}, \tag{3.4}$$

where r is the recoiler, and $\hat{s} = m_{ar}^2$, giving the ratio of the total squared invariant mass before and after the splitting. The actual $p_\perp$ of b and c then becomes

$$p_{\perp b,c}^2 = (1-z)Q^2 - \frac{Q^2}{m_{ar}^2} \approx p_{\perp evol}^2. \tag{3.5}$$

For example, let us consider the ISR radiation of a quark from the incoming parton from the left side (P_b) as illustrated on fig. 3.2. Here we have a $P_a \to p_c P_b$ splitting. If we know the momentum of emitted particle (p_c) and we know the momentum of the incoming parton, we can calculate a new momentum of the incoming parton P_b. This new momentum P_b now boosts all of the outgoing particles in the hard subprocess (even leptons and bosons), but leaving their momentum with the respect to each other intact.

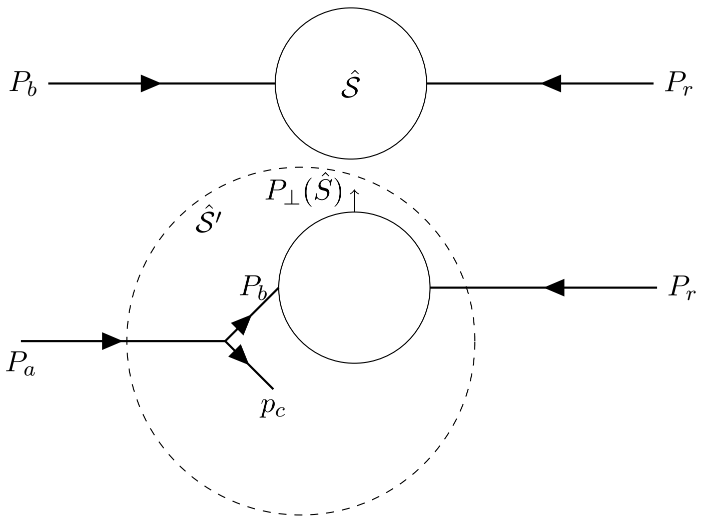


FIGURE 3.2: ISR shown before and after the splitting. First we see two incoming partons with the momentum P_b and P_r and a hard system with the total invariant mass $\hat{S}$. After the ISR momentum of the incoming parton from the left side has changed to P_a when a particle with momentum p_c was emitted, resulting in hard system getting a transverse boost $P_\perp(\hat{S})$. New invariant mass of the hard system is $\hat{S}'$. In this case $z = \frac{\hat{S}}{\hat{S}'}$.

FSR. For the FSR, the recoil of emissions is assumed to be taken entirely by the other end of the color dipole. Unlike the ISR case, in the FSR the only momenta of the emitter, recoiler and emitted particle are changed. Let us now study the kinematics in the **FF** case. The evolution variable is defined as:

$$p^2_{\perp evol} = z(1-z)Q^2 \tag{3.6}$$
$$\text{with } m_a^2 = Q^2 \text{ and } m_b^2 = m_c^2 = 0.$$

In the rest frame of a dipole ar (see fig. 3.3), the branching $a \to bc$ at an evolution scale $p_{\perp evol}$ implies that a parton a acquires a virtuality

$$Q^2 = \frac{p^2_{\perp evol}}{z(1-z)}. \tag{3.7}$$

Its energy is thereby increased from the original, and the recoiler energy is reduced accordingly. Parton b then takes a fraction z of the increased emitter energy, and c takes 1-z.

The kinematics of an **FI** branching has some differences relative to a FF one. In the dipole rest frame a fraction Q^2/m^2_{ar} of the recoiler energy is given from the recoiler to the emitter, as above. But the recoiler is not a final-state particle, so the increase of a momentum is not compensated anywhere in the final state. Instead, the incoming parton that the recoiler represents must have its momentum increased by the same amount as

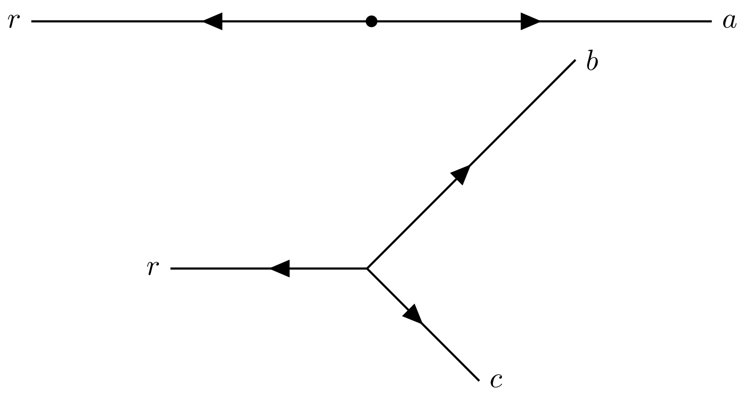


FIGURE 3.3: FSR shown before and after the splitting. First we see two particles a and r. After FSR ($a \to bc$ with r as a recoiler), momentum of the recoiler (r) has changed.

the emitter. The dipole mass m_{ar} and the squared subcollision mass $\hat{s}$ are increased in the process (instead of being unchanged as in the FF case). The ISR in FI dipoles is done as if it was an II dipole. The issue remains on how to subdivide the FSR and ISR in an FI dipole. One can naively think that boosting to the rest frame of the emitter and recoiler and dividing the radiation accordingly would do the trick. If we imagine a situation where a soft final-state parton forms a dipole with a highly energetic initial-state parton, then this will result in partons sometimes being radiated far away (in the η) from the final-state parton. In order to suppress this, one has to introduce some kind of a suppression weight. As one can read in more detail in ref [25], the DPS in PYTHIA introduces the weight

$$\mathcal{W} = \frac{p_\perp p_{\perp b,c}}{p_\perp p_{\perp b,c} + Q^2}, \tag{3.8}$$

where $p_\perp = p_{\perp evol}$ and $p_{\perp b,c} = p_{\perp b} + p_{\perp c}$.

3.2 Distance formulas

What we want to do now, is to "invert" the DPS, from the smallest $p_\perp$-emission to the largest one. This is the way k_t-algorithm does it as well, looking for two particles with the smallest k_t ($p_\perp$) and clustering them to a single cluster (as if "inverting" a $1 \to 2$ parton showers). Of course, this can never be exact because of many other effects coming from the hadronization, multiparton interactions, etc. The dipole-k_t algorithm is based on inverting $2 \to 3$ emissions. So, for each particle we want to think of how it could have been emitted. It could have been radiated from the final-state particles as a FSR. In order to check it, we need to calculate the $p_{\perp evol}$ between all the triplets of particles. It could have been radiated from the FI dipole as FSR. In order to check that case, we need to calculate the $p_{\perp evol}$ between all the pairs of particles with the respect to

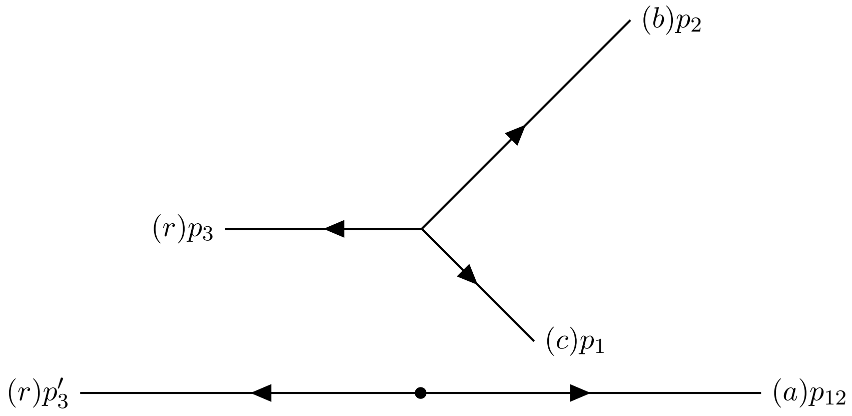


FIGURE 3.4: FSR shown in reverse. First we see three particles b,c and r with respective momentum p_2, p_1 and p'_3. After the FSR clustering($3 \to 2$), b and c have been clustered to a, and the momentum of the recoiler(r) has changed.

the incoming partons and construct the condition from 3.8 in order to determine what kind of radiation it was. It could have been radiated from the incoming parton as an ISR. We need to calculate the $p_{\perp evol}$ between each particle and the incoming partons in order to check this last possible case.

Let us start by looking at a FF-emission. A particle is emitted from the emitter with some recoiler in their rest frame (see Figure 3.4). The variables in the splitting functions are defined in the dipole rest frame, but we can rewrite them in terms of Lorentz invariant quantities. Since we are in the rest frame $p_{z3} = p_{z(1,2)}$. With the assumption that all particles are massless one gets, in the rest frame:

$$\begin{aligned}
E_3 &= |p_{z3}| = \sqrt{(E_1+E_2)^2 - S_{12}} \\
E_3^2 &= (E_1+E_2)^2 - S_{12} \\
E_3^2 &= (\sqrt{S} - E_3)^2 - S_{12} \\
2\frac{E_3}{\sqrt{S}} &= 1 - \frac{S_{12}}{S} \qquad (3.9)\\
z &= \frac{E_1}{E_1+E_2} = \frac{S-S_{23}}{2S - S_{23} - S_{13}} \qquad (3.10)\\
Q^2 &= S_{12} \qquad (3.11)
\end{aligned}$$

where S_{ij} is the invariant mass of particles i and j, and S is the total invariant mass of the 3 particles.

Using the DPS formula for the FSR (eq. 3.6) and eq. 3.9 - 3.11, one can define a distance measure for the a FF clustering in the dipole-k_t algorithm as:

$$p_{\perp evol}^2 = \frac{(S-S_{13})(S-S_{23})}{(2S-S_{13}-S_{23})^2} S_{12}. \qquad (3.12)$$

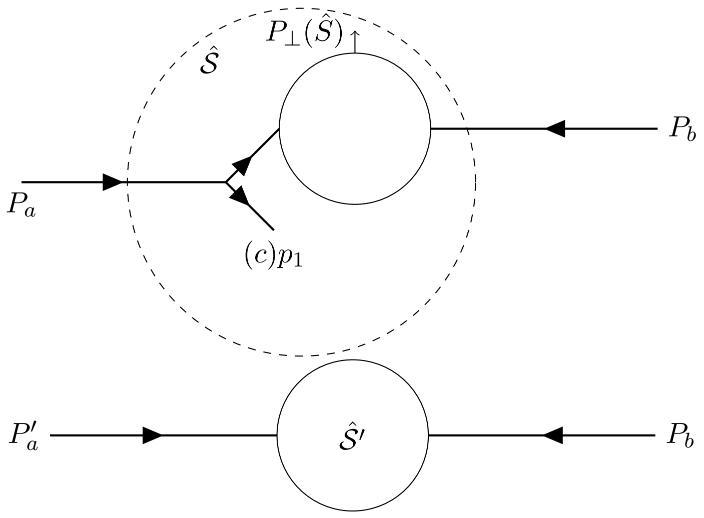


FIGURE 3.5: ISR shown in reverse. First we see a hard system with an invariant mass $\hat{S}$ and a particle (c) with momentum p_1 that we want to cluster. After the clustering is done, the momentum of the incoming parton P_a has changed and a hard system with a new total invariant mass $\hat{S}'$ has been boosted by $-P_\perp(\hat{S})$. In this case $z = \frac{\hat{S}'}{\hat{S}}$.

If we now look at the ISR emission, situation is very similar as the one above. We are trying to do a reconstruction shown on fig. 3.5. Using the DPS formula for the ISR (eq. 3.3) and eq. 3.9 - 3.11 one can define a distance measure for II clusterings in the dipole-k_t algorithm as:

$$p^2_{\perp evol} = -\left(1 - \frac{S_{1P_a}}{\hat{S}}\right) S_{P'_a}, \tag{3.13}$$

where P_a stands for the momentum of one incoming parton, P'_a for that momentum after emission, c for the emitted particle, and $\hat{S}$ for the invariant squared mass of all the particles from the collision (even the ones that are not clustered to jets, such as leptons).

The final-initial (FI) distance is calculated using eq. 3.12 where the recoiler is represented by the one of the incoming partons. If this was done with no condition, too much FSR clusterings would occur and hence result in too hard jets. Because of the suppression weight used in the DPS (eq. 3.8), one has to have some sort of a condition. We do the FI clusterings only when

$$p_{\perp evol} p_{\perp b,c} > m^2_{bc} \tag{3.14}$$

is satisfied. By doing this, we have replaced the suppression weight function in the DPS with a step function in order to get rid of an unwanted excess of the momentum.
One can now construct the dipole-k_t algorithm using eq. 3.12 - 3.14 for the $3 \to 2$ dipole-based recombination scheme.

3.3 Algorithm summary

The idea of this section is to give the reader an overview of the algorithm, with a more detailed description of how to use the code and its implementation given in the Appendix A. General algorithm construction idea is fairly simple and similar to the one of the k_t-algorithm described in the section 2.3.2. We take all the particles in the event and consider them as initial "clusters". The dipole-k_t algorithm then is formulated as follows:

1. For each cluster (c):

   - For each incoming momentum work out the distance corresponding to the ISR:
   $$d_{c,P_a} = -\left(1 - \frac{S_{cP_a}}{\hat{s}}\right) S_{P'_a}. \tag{3.15}$$
   - For every other cluster (e) work out the distance corresponding to the FSR from the FI-dipole for each of the incoming partons (but only if eq. 3.14 is satisfied):
   $$d_{e,c,P_a} = \frac{(S - S_{cP_a})(S - S_{eP_a})}{(2S - S_{cP_a} - S_{eP_a})^2} S_{ce}, \tag{3.16}$$
   $$\tag{3.17}$$
     - For every other cluster (r) work out the distance corresponding to the FSR from the FF-dipole:
     $$d_{e,c,r} = \frac{(S - S_{cr})(S - S_{er})}{(2S - S_{cr} - S_{er})^2} S_{ce}. \tag{3.18}$$

   Where S is the invariant mass squared of all the clusters used in the distance calculation, S_{ij} is the invariant mass squared of the clusters i and j, and $\hat{s}$ is the invariant mass squared of all the final-state particles.

2. Find the minimum between all $d_{e,c,r}$, d_{e,c,P_a} and d_{c,P_a}. If the minimum is above some d_{cut}, stop.

3. If the minimum distance is $d_{e,c,r}$ recombine $e,c,r \to e',r'$.
   If the minimum distance is d_{e,c,P_a} recombine $e,c,P_a \to e',P'_a$.
   If the minimum distance is d_{c,P_a} recombine $c,P_a \to P'_a$, and boost all other clusters.

4. If the number of clusters left is larger than the number of wanted jets ($n_{clusters} > n_{min}$) return to step 1. Otherwise stop.

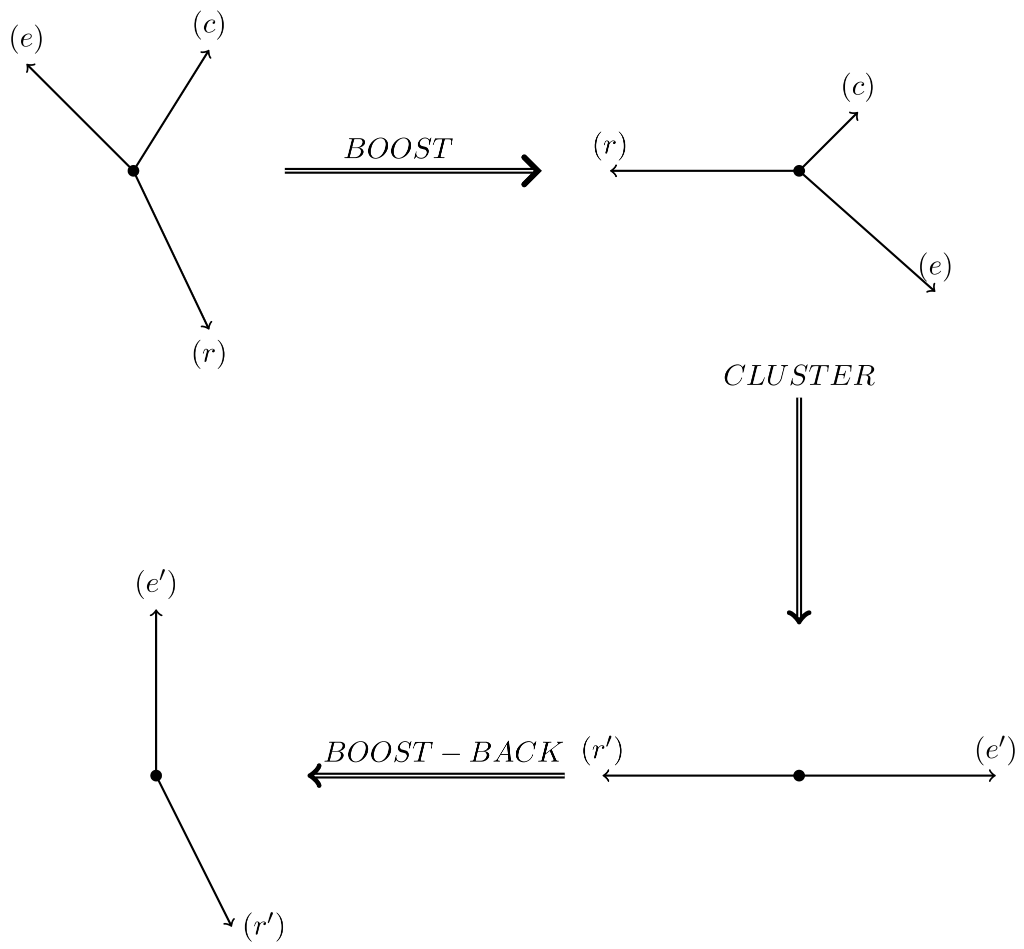


FIGURE 3.6: Recombination scheme for dipole-k_t algorithm. The scheme shows how clusterings are done, starting with three clusters that are boosted to their rest frame, recombined and boosted back to produce final result of the clustering step.

Note that the dipole-k_t algorithm can be stopped in two ways. It can be stopped when all the distances are above some cut-off giving the inclusive version of the algorithm, or when wanted number of jets was reconstructed giving the exclusive version of the algorithm. The recombination scheme used in step 3 (illustrated in fig.3.6) is the inversion of the DPS and is done as follows. For the FSR, particles are boosted to their rest frame, with r lying in the -z direction. Then total momentum of e,c and r is calculated and with that the new particles $e\prime$ and $r\prime$ are constructed along the $+z$ and -z direction respectively. After this has been done, $e\prime$ and $r\prime$ are boosted back with an inverse Lorentz transformation.

In the FI dipole FSR case, particles are boosted to the rest frame, with P_a lying in the -z direction. Then total momentum of e,c and P_a is calculated and used to construct e' and P'_a. Afterwards e' and P'_a are boosted back with an inverse Lorentz transformation. Finally, for the ISR momentum of c is simply subtracted from the incoming parton P_a

to create P'_a. Afterwards all of the remaining particles are boosted.

Note here, that this recombination scheme gives massless jets as a result (contrary to the k_t recombination scheme). Some other details need to be mentioned here, such as the option of the dipole-k_t algorithm to have all the particles of an event as input, rather than just ones that are to be clustered. They are separated to "tracks" and "others". Because we need to know $\hat{s}$ of the system, we need to include all particles, even the ones we do not want to cluster to jets. We define these particles as "others" and boost them when the ISR recombination occurred, as described above. We know that detectors have some limited acceptance in the angle, so it is highly unlikely to have the momenta of all particles when analyzing the date from an experiment. Also, the presence of neutrinos is taken into account. When we generate the data with the MC simulation we know the momentum of every neutrino, but in a real life experiment that is not the case. The unbalance in the $p_\perp$ is calculated and assigned to a single neutrino with $p_z = 0$ (since we do not know the total energy missing p_z cannot be calculated) which is then added to the "others".

3.4 The speed issue

One of the main aspects of every jet clustering algorithm is the time needed to cluster N particles. With the need to check all the triplets of particles in each step, the dipole-k_t algorithm has higher computational complexity than the k_t-algorithm. One should note that, since we are using Lorentz invariant variables we are speeding things up. If we were not to use Lorentz invariant variables we would have to boost the particle to the rest frames in order to calculate their distance measure, which would slow the algorithm even more. Because we are dealing with the triplets, rather than pairs (k_t-algorithm), total run time goes as $\mathcal{O}(N^3)$ (one could naively think that the run time goes as $\mathcal{O}(N^4)$, but with a trick that can be lowered, as described in Appendix.A). It takes ∼1 minute to cluster a realistic LHC event with ∼1000 particles. It is not trivial to speed this up, and one should maybe look for a geometrical solution to this problem (as was done in the FASTJET implementation of the k_t-algorithm [24]).

3.5 Algorithm validation

With the idea of inverting the DPS processes implemented in the dipole-k_t algorithm, it is left to see what such an algorithm can do when competing with the k_t-algorithm. In order to validate the algorithm we have used Monte Carlo (MC) simulations, because they are easy to generate, manipulate and reproduce. We have have set up a PYTHIA8 code that generates some processes of interest ($q\bar{q} \to Wg$, 2-jet production, and 3-jet production). We would also want to check that the algorithm does sensible things for the other generators, but this will not be done here. In PYTHIA it is possible to stop each of these processes after an arbitrary number of steps[1](in emissions), and store the data (4-momentum) of the current particles and therefore study the algorithm from the most simple clustering problem, such as clustering two particles back to one. Also, it is possible to find out what the original state looked like before the parton shower. With this option it is possible to know what the exact solution of a jet clustering algorithm should be and how well we can approximate it. This is why we have decided to study the exclusive version of the algorithms, in order to get the number of jets that corresponds to the number of particles in the original state. We have decided to compare our algorithm with the k_t-algorithm from FASTJET. We would like to emphasize that only the exclusive version of the k_t-algorithm was used in order to compare with the exclusive version of the dipole-k_t algorithm. So, the results presented in this work do not reflect the normal use of the k_t-algorithm. This was done because we believe that the exclusive variant of the dipole-k_t algorithm should be its best feature.

In order to compare the two algorithms, one has to decide which variables to use. We have decided to reconstruct the $p_\perp$ and the rapidity (y) of jets and compare them with the values obtained in the original hard interaction. For example, when we reconstruct the two jets $p_\perp$ the value we should get is the $p_\perp$ of the two partons in the initial-state, before any emission was done. We then express the results as $p_\perp/p_{\perp 0}$, $y{-}y_0$ (where 0

[1]One can easily do this with the help of User Hooks class in PYTHIA

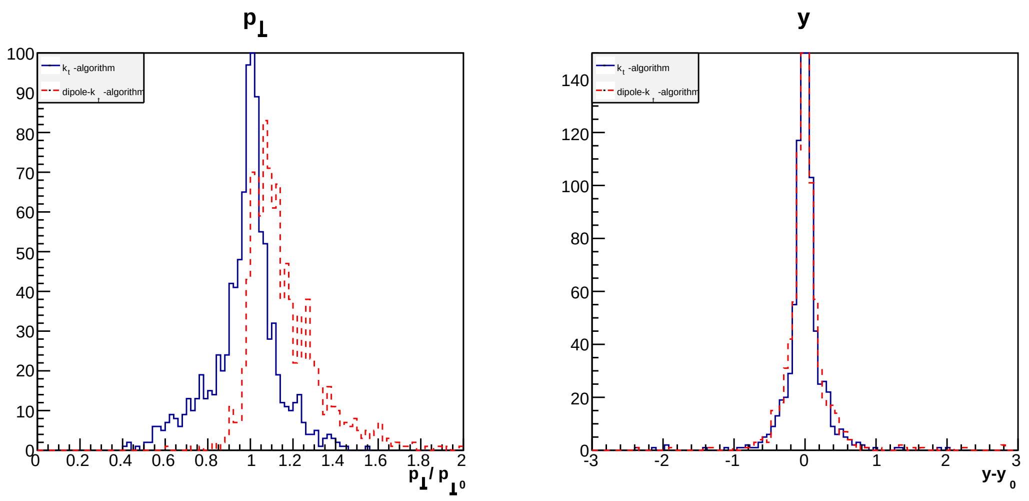


Figure 3.7: Histograms comparing dipole-k_t results (red line) with the k_t results (blue line) for $p_\perp$ (on the left) and y (on the right) in $q\bar{q} \to Wg$ case. Clustering was done on events with full parton shower, hadronization and multi-parton interactions.

$\mathbf{p_\perp}$	$\mathbf{Dipole-k_t}$		$\mathbf{k_t}$	
	Mean	**RMS**	**Mean**	**RMS**
1 Step	1,04	0,10	0,98	0,13
3 Steps	1,06	0,12	0,96	0,15
10 Steps	1,06	0,12	0,95	0,15
END	1,06	0,12	0,95	0,15
END+H	1,06	0,11	0,95	0,14
END+H+MPI	1,16	0,17	0,97	0,16

TABLE 3.1: Mean and RMS values of p_t in $q\bar{q} \to Wg$ analysis. Red text indicates results closer to the expected value. END represents end of parton shower, +H included hadronization, and +MPI included multi-parton interactions.

y	$\mathbf{Dipole-k_t}$		$\mathbf{k_t}$	
	Mean	**RMS**	**Mean**	**RMS**
1 Step	0,01	0,29	0,01	0,26
3 Steps	0,01	0,30	0,01	0,28
10 Steps	0,00	0,31	0,02	0,29
END	0,01	0,29	0,01	0,28
END+H	0,00	0,25	-0,01	0,27
END+H+MPI	0,01	0,31	-0,01	0,27

TABLE 3.2: Mean and RMS values of y in $q\bar{q} \to Wg$ analysis. Red text indicates results closer to the expected value. END represents end of parton shower, +H included hadronization, and +MPI included multi-parton interactions.

indicates the initial value) and plot the corresponding histograms. It is obvious that the closer the mean value of the $p_\perp$ histogram is to 1, and the y histogram to 0, the better are the results. Additionally, one can also express the spread of the results using the root mean square (RMS). Smaller RMS means less spread distribution and is something we would want in our results.

We have done the analysis after 1, 3 and 10 steps of emissions (with the MPI and the hadronization turned off). Furthermore we have analyzed the results after the full parton shower with the hadronization turned off and on, and at the end we have analyzed a realistic example with the full parton show, hadronization and MPI. In all of the cases minimum possible $p_\perp$ of particles from the hard subprocess is set to 100 GeV, and the R value used by the k_t-algorithm is set to 0,7.

We start with the results from the $q\bar{q} \to Wg$ analysis. Note here, that we do not allow the decay of the W, and we add its momentum to the "others" for the dipole-k_t clustering. We search for the one jet produced by a gluon. Results for the $p_\perp$ and y are given in tables 3.1 and 3.2. Furthermore, the histograms showing the results for the final event with hadronization and MPI (realistic event) are shown in fig. 3.7.

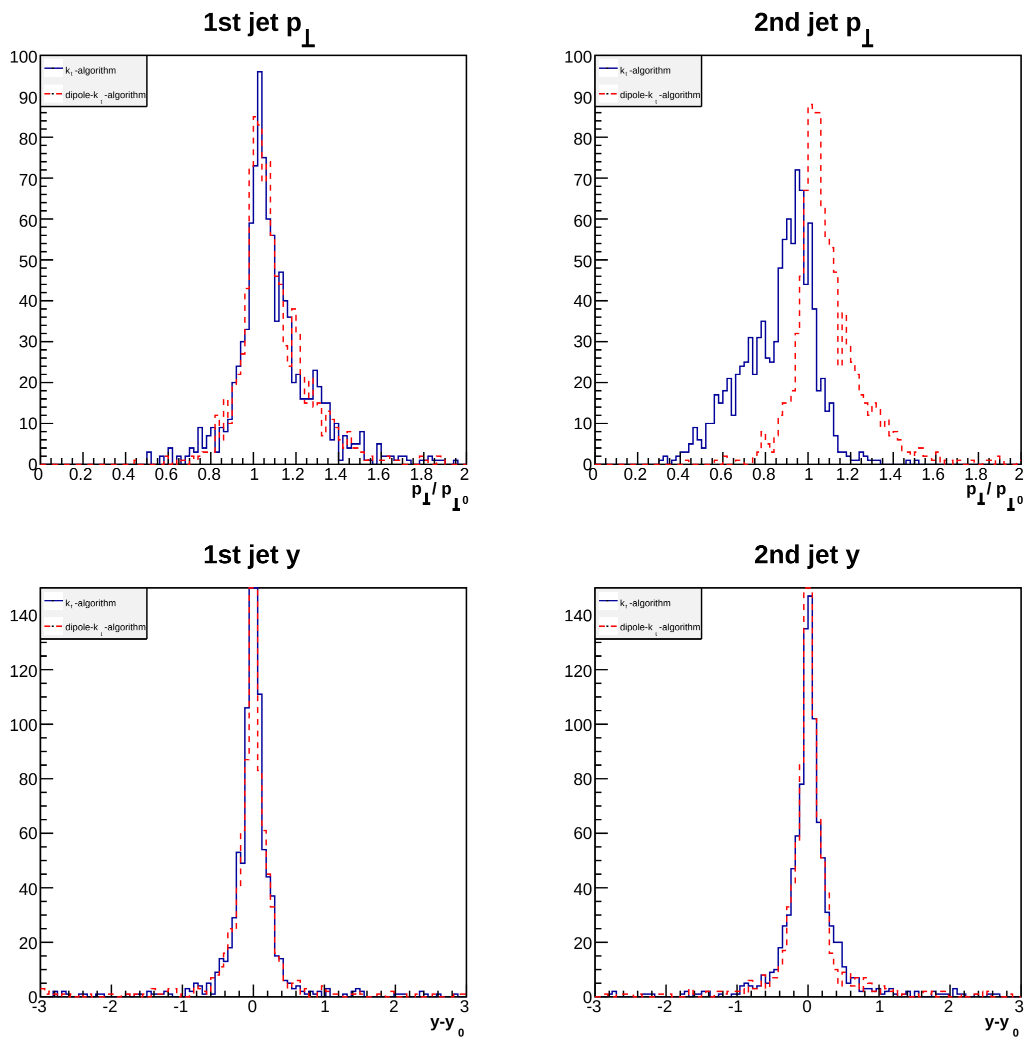


FIGURE 3.8: Histograms comparing dipole-k_t results (red line) with k_t results (blue line) in clustering to two jets case. Clustering is done on event with full parton shower, hadronization and multi-parton interactions.

In the most simple task of reconstructing one jet, the dipole-k_t and the k_t algorithm share more or less similar results, they have same success when comparing the results to the expected values. Clustering of realistic event (3.7) shows that the k_t-algorithm tends to cluster softer jets, and the dipole-k_t algorithm harder jets, but both give decent results. Also, both give almost the same results for the rapidity.

Let us now consider a bit more complicated process, where we have a production of two jets. Because the transverse momenta of two jets is the same, we need to define how to differ the two jets. In the $p_\perp$ analysis we declare a jet with higher reconstructed $p_\perp$ as the 1st jet. In the rapidity analysis we declare a jet with smaller absolute value of reconstructed rapidity as the 1st jet. Results from tables 3.3 and 3.4 show that the

$\mathbf{p_\perp}$	**1st JET**				**2nd JET**			
	Dipole − $\mathbf{k_t}$		$\mathbf{k_t}$		**Dipole − $\mathbf{k_t}$**		$\mathbf{k_t}$	
Step	**Mean**	**RMS**	**Mean**	**RMS**	**Mean**	**RMS**	**Mean**	**RMS**
1 Step	1,03	0,12	1,10	0,16	1,03	0,12	0,91	0,13
3 Steps	1,05	0,14	1,09	0,18	1,04	0,14	0,87	0,15
10 Steps	1,05	0,14	1,09	0,18	1,05	0,14	0,85	0,16
END	1,05	0,14	1,08	0,18	1,05	0,14	0,85	0,16
END+H	1,05	0,15	1,07	0,19	1,05	0,15	0,85	0,16
END+H+MPI	1,11	0,15	1,11	0,19	1,11	0,15	0,87	0,17

TABLE 3.3: Mean and RMS values of p_t in two jet production analysis. Red text indicates results closer to the expected value. END represents end of parton shower, +H included hadronization, and +MPI included multi-parton interactions.

$\mathbf{y}$	**1st JET**				**2nd JET**			
	Dipole − $\mathbf{k_t}$		$\mathbf{k_t}$		**Dipole − $\mathbf{k_t}$**		$\mathbf{k_t}$	
Step	**Mean**	**RMS**	**Mean**	**RMS**	**Mean**	**RMS**	**Mean**	**RMS**
1 Step	-0,02	0,46	0,02	0,35	0,21	0,89	0,36	0,80
3 Steps	0,01	0,47	0,01	0,45	-0,31	0,83	-0,35	0,81
10 Steps	0,04	0,47	0,01	0,47	-0,35	0,82	-0,34	0,79
END	0,06	0,43	0,01	0,48	-0,36	0,82	-0,35	0,81
END+H	0,06	0,45	0,01	0,49	-0,38	0,80	-0,37	0,81
END+H+MPI	0,01	0,41	-0,03	0,38	-0,03	0,40	0,01	0,44

TABLE 3.4: Mean and RMS values of y in two jet production analysis. Red text indicates results closer to the expected value. END represents end of parton shower, +H included hadronization, and +MPI included multi-parton interactions.

dipole-k_t algorithm reconstructs the $p_\perp$ rather better than the k_t-algorithm. Clear difference is that the dipole-k_t algorithm shows symmetric results for two jets (as expected, because the dipole-k_t algorithm preserves momenta in the recombination, so the two jets will always have symmetrical transverse momenta), while the k_t-algorithm tends to get asymmetrical results. In the rapidity analysis the k_t-algorithm has a few more values closer to the expected ones, but the difference between the two algorithms is neglectable. After full parton shower, hadronization and included MPI fig. 3.8 shows that the only noticeable difference between the algorithms is that the k_t-algorithm tends to get the second jet a bit too soft.

Finally we will look at the production of three jets. The results of clusterings are shown in tables 3.5 and 3.6, and the histogram of the realistic event clustering is shown in fig.3.9. From the quick view on the tables one can see that again the dipole-k_t algorithm has results closer to the expected values. Finally from fig.3.9 one can say that the results are very similar, with the dipole-k_t algorithm giving somewhat better transverse momentum reconstruction. What we still have not discussed, and is important aspect of a jet clustering algorithm, is the influence of the hadronization and the MPI on the results. If one thinks about the hadronization process in terms of the string fragmentation model (as it is done in PYTHIA), with the strings formed between the color dipoles in their rest frames and then fragmented similarly to the emission of particles from an color dipole. Because of that, there should be no big difference in the results with or without hadronization. As presented in our data, this is the case with the dipole-k_t algorithm, with neglectable differences in the results with the included or excluded hadronization process. On the other hand, the MPI present a bigger challenge. What we hope, since the probability for the MPI parton to have a high $p_\perp$ is rather small, is to have most of the MPI clustered as the ISR to the incoming parton. This way we would get rid of the unwanted extra momentum, and hence be insensitive to the MPI. As one can see from the results, the inclusion of the MPI does change the results, but not by much. This is because, in some cases it can happen that the $p_\perp$ of the MPI is close enough to a jet from the hard subprocess, and therefore added to the momentum of the jets. What one needs to know is that this problem varies with the $p_\perp$ of the produced jets. For smaller values of $p_\perp$ the dipole-k_t algorithm should tend to give too hard jets, while for larger values of p_t it should be more or less insensitive to the MPI.

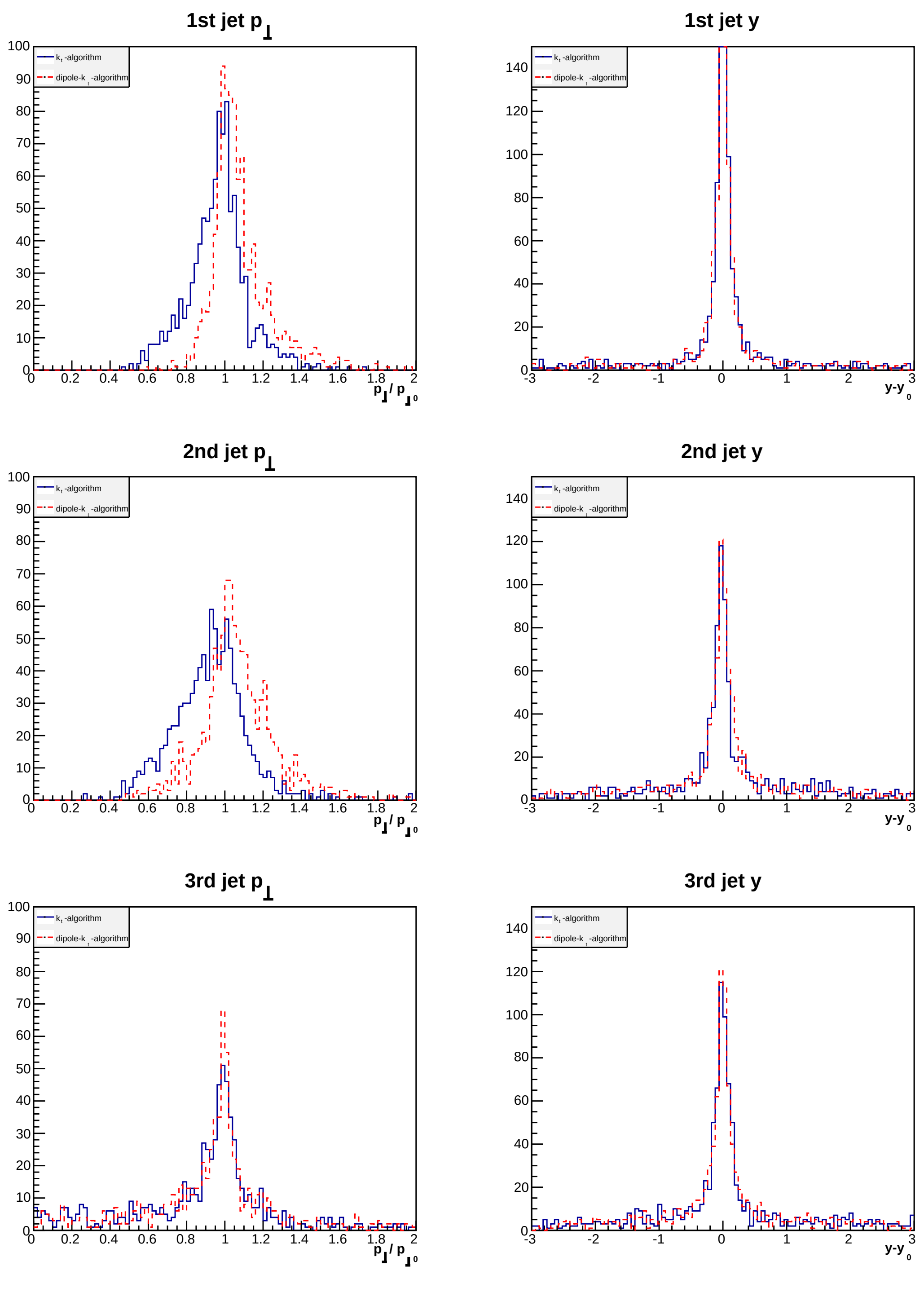


FIGURE 3.9: Histograms comparing dipole-k_t results (red line) with k_t results (red line) in clustering to three jets case. Clustering is done on event with full parton shower, hadronization and multi-parton interactions.

$\mathbf{p_\perp}$	**1st JET**				**2nd JET**			
	$\mathbf{Dipole - k_t}$		$\mathbf{k_t}$		$\mathbf{Dipole - k_t}$		$\mathbf{k_t}$	
Step	**Mean**	**RMS**	**Mean**	**RMS**	**Mean**	**RMS**	**Mean**	**RMS**
1 Step	1,03	0,12	0,99	0,13	1,05	0,15	0,99	0,14
3 Steps	1,04	0,14	0,97	0,15	1,07	0,17	0,95	0,16
10 Steps	1,05	0,14	0,95	0,16	1,07	0,18	0,93	0,16
END	1,05	0,14	0,94	0,15	1,07	0,18	0,93	0,16
END+H	1,04	0,14	0,94	0,15	1,07	0,18	0,92	0,16
END+H+MPI	1,09	0,16	0,96	0,16	1,12	0,19	0,94	0,17
	3rd JET							
1 Step	1,00	0,17	0,96	0,16				
3 Steps	1,00	0,18	0,93	0,19				
10 Steps	1,01	0,19	0,89	0,21				
END	1,01	0,19	0,89	0,20				
END+H	1,02	0,19	0,89	0,21				
END+H+MPI	1,07	0,20	0,92	0,19				

TABLE 3.5: Mean and RMS values of p_t in three jet production analysis. Red text indicates results closer to the expected value. END represents end of parton shower, +H included hadronization, and +MPI included multi-parton interactions.

$\mathbf{y}$	**1st JET**				**2nd JET**			
	$\mathbf{Dipole - k_t}$		$\mathbf{k_t}$		$\mathbf{Dipole - k_t}$		$\mathbf{k_t}$	
Step	**Mean**	**RMS**	**Mean**	**RMS**	**Mean**	**RMS**	**Mean**	**RMS**
1 Step	-0,01	0,56	0,00	0,69	-0,00	0,84	-0,03	0,90
3 Steps	0,00	0,68	0,01	0,73	-0,01	0,94	-0,02	0,98
10 Steps	0,00	0,68	0,00	0,74	-0,04	0,92	-0,05	1,01
END	-0,01	0,69	0,00	0,74	-0,03	0,92	-0,03	1,00
END+H	0,00	0,69	0,02	0,73	-0,04	0,92	-0,04	0,98
END+H+MPI	0,00	0,73	0,01	0,77	0,04	0,97	0,00	1,04
	3rd JET							
1 Step	-0,03	0,81	0,00	0,85				
3 Steps	-0,04	0,92	0,00	0,96				
10 Steps	-0,03	0,92	-0,01	0,97				
END	-0,01	0,92	0,01	0,98				
END+H	-0,02	0,91	0,03	0,96				
END+H+MPI	-0,01	0,99	0,03	1,00				

TABLE 3.6: Mean and RMS values of y in three jet production analysis. Red text indicates results closer to the expected value. END represents end of parton shower, +H included hadronization, and +MPI included multi-parton interactions.

In the k_t-algorithm case, one can see the hadronization process as a "splash-out" of particles, and the MPI as a "splash-in" of particles. The goal then is to try and find the value of R that will include roughly the same amount of "splash-in" and "splash-out" in order to cancel out their effects.

In order to investigate the dependence of the results on the $p_\perp$ of the event as well as on the chosen value for the R parameter, we will study the two jet production in more detail. Also, we have taken only particles with rapidity $|\eta| < 5$ in order to investigate the effects of the detector acceptance which can occur in an experimental enviroment. Even though the dipole-k_t algorithm does not have a free parameter, one can see condition from eq.3.14 as "dynamic R" value (since it controls the separation between the ISR and FSR clusterings). We have done the analysis for different minimum $p_\perp$ values, and the results are given in table 3.7. One can clearly see, that for the dipole-k_t algorithm results are getting better as $p_{\perp min}$ grows. This is because of the previously discussed MPI problem, that tends to disappear for high $p_\perp$ values of jets. Also, because we do not have the information about all particles from the collision, the results from the dipole-k_t algorithm are not symmetrical anymore. For the future development of the algorithm, we plan to investigate in detail how does this cut in the pseudorapidity effect the results of the clusterings. For the k_t-algorithm, one can not establish such a connection between the $p_{\perp min}$ value and the results. Also, it is hard to say for which value of the parameter R it gives the best results. For a complete k_t analysis of such events, one should play around with different values of R in order to obtain an optimal result.

With all the data presented in this section, it is clear that the dipole-k_t algorithm does clusterings with similar precision as the k_t-algorithm (even slightly better in some cases). Even though two algorithms share the same main concept there are couple of big difference between them. Dipole-k_t algorithm uses dipole-based clustering procedure, has Lorentz invariant distance measure and does not have a free radius parameter[2]. The k_t-algorithm uses the radius parameter in order to decide what should be clustered to a jet and what to the incoming parton, and clustering results depend on what value of R is choose. It is encouraging that we can reproduce the k_t-algorithm results with the dipole-k_t algorithm, even with those differences.

[2]More precisely the dipole-k_t algorithm does use some kind of R parameter as discussed before, but it is not a free user-defined parameter.

$\mathbf{p_\perp}$	**1st JET**							
	$\mathbf{Dipole - k_t}$		$\mathbf{k_t(R = 0.4)}$		$\mathbf{k_t(R = 0.7)}$		$\mathbf{k_t(R = 1.0)}$	
$\mathbf{p_{\perp min}(GeV)}$	**Mean**	**RMS**	**Mean**	**RMS**	**Mean**	**RMS**	**Mean**	**RMS**
20	1,38	0,29	1,01	0,28	1,22	0,28	1,34	0,29
50	1,20	0,22	1,02	0,23	1,13	0,22	1,22	0,23
100	1,11	0,15	1,02	0,19	1,11	0,19	1,15	0,19
200	1,08	0,14	1,03	0,18	1,08	0,17	1,12	0,16
	2nd JET							
20	1,29	0,33	0,68	0,22	0,89	0,24	1,06	0,27
50	1,14	0,23	0,74	0,19	0,87	0,19	0,96	0,19
100	1,11	0,15	0,77	0,18	0,87	0,17	0,84	0,17
200	1,07	0,14	0,80	0,17	0,88	0,15	0,94	0,14

TABLE 3.7: Mean and RMS values of the p_t in the two jet production analysis. Red text indicates results closest to the expected value. Only particles with rapidity $|\eta| < 5$ are included in the analysis.

Chapter 4

Prospect in boosted jets

4.1 Introduction

With the increase in energy LHC is exploring phenomena at energies well above the electroweak scale. Because of this, techniques developed for lower energies where the weak vector bosons could be considered slow-moving, have to be reconsidered. In particular, in the context of jet analysis, the large boost of electroweak bosons and top quarks causes their hadronic decays to became collimated inside a single jet. A new research field has emerged in recent years, investigating how best to identify the substructure that appears in single fat jets, as reviewed in refs. [26][27].

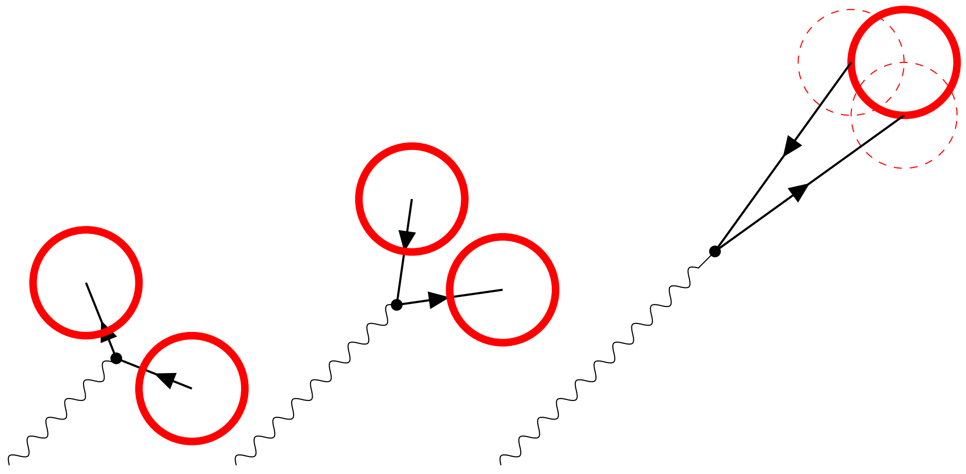


FIGURE 4.1: An illustration of a boson decaying into two quarks with three different values of transverse momentum (longer boson line indicates higher $p_{\perp}(boson)$). One can see that for high transverse boosts of the boson (on the right) it is no longer easy to differ the two jets produced by the quarks.

Larger the boost of the boson we get more collinear jets as result (shown in fig.4.1). This has been a problem for most of the known jet clustering algorithms. One can think that just taking smaller R values for clustering sequence would solve this problem, but then one looses a lot of particles. It is done the opposite way, larger R values are taken to find collinear jets in a single fat jet which is then run trough clustering sequence again to find substructure and get rid of the background (as already mentioned in section 2.3.3). Knowing that the dipole-k_t algorithm uses "dynamic cone size" and Lorentz invariant distance measure we think it could do well in boosted jets case.

In this section we present the reader with an analysis of such events. What one has to have in mind is that this analysis does not represent a realistic case (which is far more complicated) and is done only to verify our belief that the dipole-k_t algorithm should be more or less insensitive to the boost of decaying particles. Furthermore, we compare the dipole-k_t results with an exclusive variant of the k_t-algorithm, and in todays experiments only inclusive variants are used for such purposes. Bearing this in mind, one can proceed to analyze the performance of the dipole-k_t algorithm in such an environment.

4.2 Boosted $W \to q\bar{q}$

We have chosen to study a $q\bar{q} \to Wg$ process, where the W decays further to a quark-antiquark pair. The final products of this process are three jets, and the higher we set the $p_\perp$ of the W, the more collinear the jets produced in its decay will be. Our goal was to reconstruct the mass of the W boson. One first has to find two jets that come from the decay of the W and then reconstruct its mass as invariant mass of those two jets. Since we are using exclusive variant of the k_t and dipole-k_t algorithm one could think

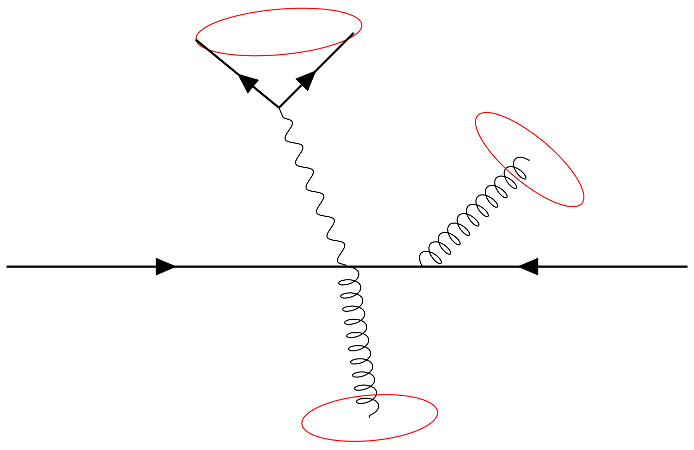


FIGURE 4.2: Possible three reconstructed jets in $q\bar{q} \to Wg$. High $p_\perp$ of a gluon from the ISR resulted in clustering of W decays into a single fat jet.

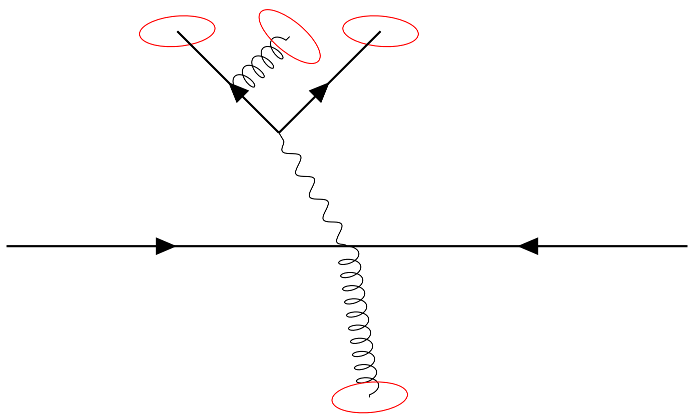


FIGURE 4.3: Possible four reconstructed jets in $q\bar{q} \to Wg$. Absence of high $p_\perp$ parton from the ISR resulted in clustering a FSR inside a "fat jet" as a stand-alone jet.

that searching for three final jets and then checking which combination of two jets is the closest one to the W mass should be sufficient. However, let us consider a situation illustrated on fig.4.2. Here, a gluon from the ISR happened to have high $p_\perp$, so among three found jets the W products are swept in a single fat jet. In this case it would be better to reconstruct four final jets, as the last step of the clustering should give two jets from a W fat jet.

On the other hand, if one always reconstructs four final jets, the situation illustrated in fig. 4.3 could occur fairly often. If there are no high-$p_\perp$ partons from the ISR, we could get a part of the FSR from the W-jets a stand-alone jet. This again would give wrong results when reconstructing the mass of the W.

In order to avoid these two problems we have decided to do as follows:

1. Find four jets.
2. Find the invariant mass M_1 of two jets that is the one closest to the known W mass.
3. Do another step in clustering in order to obtain three jets.
4. Find the invariant mass M_2 of two jets that is the one closest to the known W mass.
5. Chose the minimum of M_1 and M_2 as the reconstructed W mass.

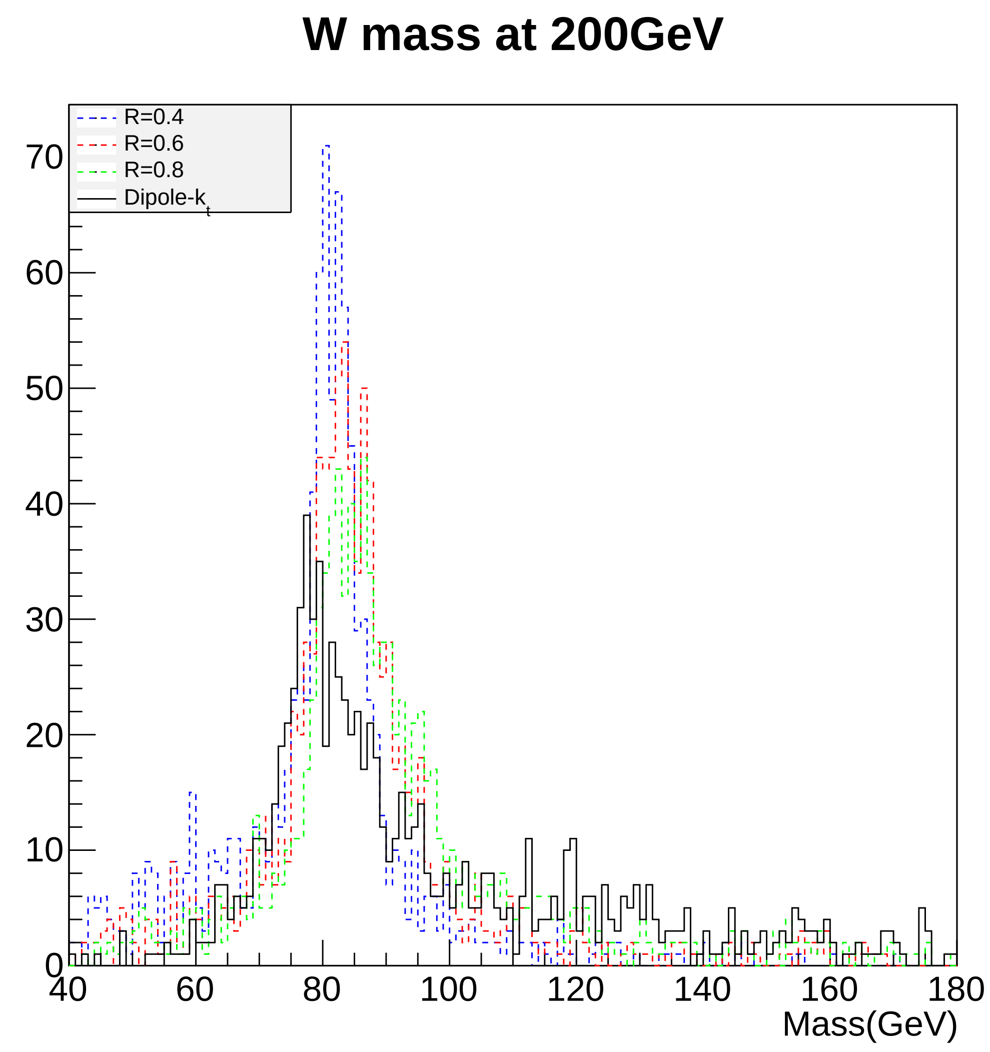


FIGURE 4.4: Histogram with the reconstructed W mass for $q\bar{q} \to Wg$ events with the dipole-k_t and k_t-algorithm ($\mathbf{p_\perp(W) \geq 200GeV}$). The k_t results are given for three different values of R.

This procedure requires the knowledge of the W mass, and thus is not relevant in search for particles with unknown mass.

Results of the analysis with the minimum $p_\perp$ of the W set to 200 GeV in the generation are shown in fig. 4.4. The dipole-k_t algorithm reconstructed a nice peek, which when fitted to the Breit-Wigner[1] function gives $M_W = (80.93 \pm 0.51) GeV$ (with known W mass of 80.38 GeV inside the 2σ). In the k_t-algorithm analysis, one can easily see that the position of the peek is strongly correlated with the chosen value of R. A more detail study of the problem is needed to determine the best value of R to use, and most likely this value would be different for another process.

In order to check how sensitive clustering algorithms are to the change in boost of the W we have chosen to repeat the analysis, now with the minimum $p_\perp$ of the W set to 500 GeV. Results are given in fig. 4.5.

Again, one can see a clear peak in the dipole-k_t case (and notice that the distribution remained almost the same), which when fitted gives $M_W = (78.93 \pm 0.45) GeV$ (with known W mass inside the 4σ). Results have changed, but not by much taking into consideration that now we are dealing with highly boosted W. The k_t-algorithm shows even higher correlation between chosen value for the R parameter and the results. As one can see, choosing too large value for R results in a very broaden distribution (even without a clear peak for the resonance).

One important part of a realistic analysis, that we have not taken into a consideration, is the background. For example if we wish to study such an event, we would have a group of events that produce three jets with some $p_{\perp,min}$ as a cut-off. In such a group, many events would not be a W production but rather some other processes that end up having three high $p_\perp$ jets. In our case, the main background would be a production of three jets from pure QCD events studied in section 3.5. It is rather complicated, and time consuming, to simulate such realistic data. Preliminary investigations indicate that the k_t-algorithm has a stronger tendency to create a mass peak in background events than the dipole-k_t algorithm, but further studies are needed to clarify this issue.

[1] $f(E) = {}^{k}/_{(E^2 - M^2)^2 + M^2\Gamma^2}$, where k is a constant, M the mass of the resonance and Γ the width of the resonance.

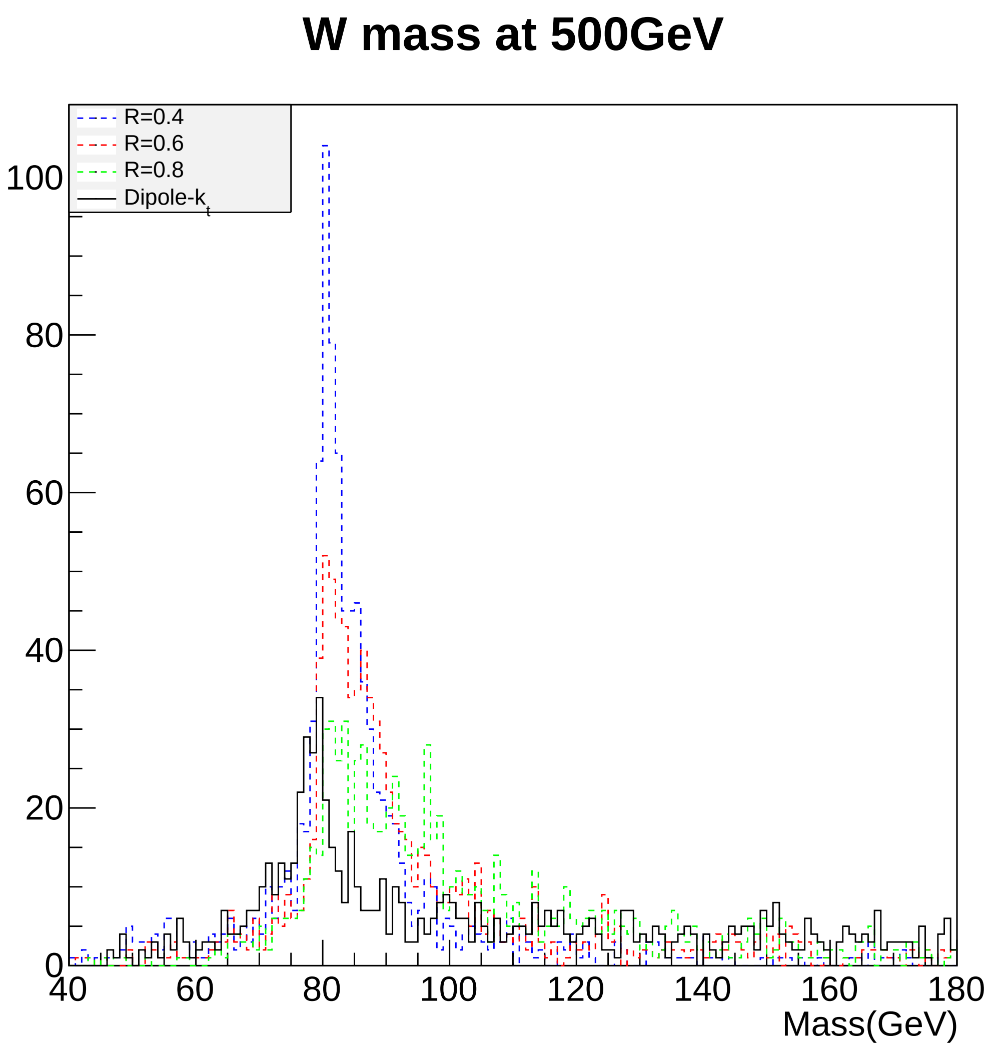


FIGURE 4.5: Histogram with the reconstructed W mass for $q\bar{q} \to Wg$ events with the dipole-k_t and k_t-algorithm ($\mathbf{p}_\perp(\mathbf{W}) \geq \mathbf{500GeV}$). The k_t results are given for three different values of R.

Chapter 5

Conclusion

Let us start with a quick review of the thesis work. The idea of the thesis was to develop a new Lorentz invariant jet clustering algorithm that does $3 \to 2$ clusterings rather than $2 \to 1$ (that almost all other clustering algorithms do). This was done by "inverting" the PYTHIA dipole-based shower framework. We have analyzed how PYTHIA does simulations of parton showers, and used it to build our recombination scheme and define a distance measure. We implemented this in a code, and dipole-k_t algorithm was born. We have decided that, for the purpose of this thesis, we are only going to analyze the exclusive version (finds requested number of jets) rather than inclusive version (finds all jets above some cut-off $p_\perp$). The k_t-algorithm was our choice for comparison in the validation process. Validation results showed that the dipole-k_t does good clusterings in three simple processes (even slightly better than the k_t-algorithm in some cases). That was a promising sign that we could do well also in a boosted jets analysis. Because of the time limitations, we have only done a very simple analysis that showed good prospects. Having a dynamic cone size (unlike a fixed cone size R, as in the k_t-algorithm) is an advantage of the dipole-k_t algorithm in such analysis compared to the k_t-algorithm where results were highly dependent on the choice of the R.

One can clearly see that, although the dipole-k_t algorithm gives good results and shows prospect for boosted jets analysis, a lot of development work as well as analysis are left to be done. Here we will mention only a couple, that we think are of the most importance, and that we plan to investigate in the near future. First of all, the dipole-k_t algorithm has to be speeded up. It is now ~ 100 times slower than the k_t-algorithm. In order to speed it up to, one should consider using Computational Geometry Algorithms Library (CGAL) [28] which is used to speed up the k_t-algorithm. Secondly, a detailed analysis of boosted jet production together with realistic background simulation is needed. One should generate as realistic as possible data, and then proceed to analyze it with the dipole-k_t algorithm. Afterwards, these results should be compared with the state-of-the art techniques used in boosted jets analysis, such as: mass tagger, trimming, pruning, etc. We are planning to investigate booth features in order to hopefully prove that the dipole-k_t algorithm can be of help in future analysis of LHC data. Other interesting point would be to analyze what results would other distance measures give (for example the distance measure used in ARIADNE).

Two main ideas prolonged trough this thesis, so we wish to finish by stating them once again. We think that usage of a Lorentz invariant distance measure is an obvious way towards development of an algorithm that has no problems when dealing with highly boosted jets. Also, since the color dipole model has shown improvements in the evolution of parton shower algorithms, $3 \rightarrow 2$ recombination schemes may very well become an important feature of new jet clustering algorithms.

Appendix A

Algorithm in detail

A.1 How to use it

Here, a more detailed overwiev of the dipole-k_t algorithm is given. The code was writen in C++. The code does not supply its own handeling of Lorentz vectors, but relies ona a so-called traits-class system based on C++ templates for interfacing to any external Lorentz vector code.

The default constructor is given as:

```
Clustering ( nmin=2, cutkt=0 )
```

where **nmin** is minimum number of jets to be constructed and **cutkt** is the cut-off in transverse momentum (only clusters with invariant transvers momentum higher than cutkt will be considered jets).

There are only two public functions that can be called by the user.

```
int cluster( tracks, others );
vector getJets( );
```

For the **cluster()** function the needed input is a vector of 4-momentum of all particles to be clustered, and optionally a vector of 4-momentum of other particles in the event. When called, this function does jet clustering, and depending on the parameters nmin and cutkt it returns the number of jets. **getJets()** returns 4-momentum of all final jets when called, after the clustering.

To summarize the usage of this algorithm:

- Define a variable of type Clustering. Set the value of **nmin** and **cutkt**.
- Call the function cluster() with all the particles one wishis to cluster as input (and optionally the other particles from the event).
- Call the function getJets() to get all the jets with their 4-momentum.

A.2 Algorithm implemetation

Here a more detailed implementation of the algorithm will be given, describing all the essential aspects of the code. When the user calls function *cluster()* it invokes following actions:

- **Initialization**
  - Frist we copy the 4-momentum of "tracks" and "others" so we can manipulate them.
  - The total invariant mass of the hard subprocess ($\hat{s}$) and momenta of the incoming partons (P_a and P_b) are calculaded from the 4-momentum of "tracks" and "others" in the previous step. Unbalance in $p_\perp$ is assigned to a neutrino with $p_z = 0$ which is now part of the "others".
  - An $(N+1)\times(N+1)$ matrix S is initialized. The matrix is filled with squared invariant masses of all pairs of particles, and of particles with respect to the incoming partons.
  - For each cluster, we calculate the minimum invariant k_t with the respect to:
    - ∗ all other pairs of clusters (corresponding to the FSR)
    - ∗ all other clusters and both incoming partons (corresponding to the FI dipole)
    - ∗ the both incoming partons (corresponding to the ISR)
  - For the FSR (which is the most time-consuming part) we can use S-matrix elements to get the invariant k_t ($S = S_{ce} + S_{cr} + S_{er}$ from eg. 3.16).
  - The first search for the minimum k_t is done with an $\mathcal{O}(N^3)$ complexity
  - Note that we do not consider FSR from FI dipoles if a pair does not pass the 3.14 cut.
- **Each step**
  - If the minimum k_t found in a previous step indicates that:

    * there was a FSR: Boost to the rest frame where the recoiler momentum is in the -z direction. Calculate the total momentum in the rest frame and construct 2 new clusters whose momentum lies in the -z and $+z$ direction. Boost these particles back. Afterwards update the relevant elements of S-matrix.
    * there was a FSR from the FI dipole: Boost to the rest frame where the incoming parton momentum is in the -z direction. Calculate the total momentum in the rest frame and construct new clusters whose momentum lies in the $+z$ and the incoming parton momentum lies in the $+z$ direction. Afterwards update the relevant elements of S-matrix.
    * there was a ISR: Simply subtract the momentum of the cluster from the incoming parton momentum. Afterwards boost all of the remaining clusters and particles (”others”). Update the relevant elements of S-matrix.
  - Note that updates of the S matrix in the step above differ:
    * If there was a FSR, then only couple of matrix elements containing at least one of those three clusters will be changed.
    * If there was a FSR from a FI dipole, then all matrix elements containing at leas one of the two clusters will be changed. Additionaly, since the momentum of the beam has changed, each element containing the beam momentum has to be updated.
    * If there was an ISR, then all the elements containing the beam momentum have to be updated. Note here, that although all the clusters have been boosted, their invariant masses remain the same.
  - Note that we save the information of the minimum k_t for each cluster, so in the next step we do not need to recalculate everything.
  - After each clustering the update of the k_t's is on $\mathcal{O}(N^2)$, because of the saved information from the last clustering.
  - Note that in a ”worst case scenatio” where all clusters have a minimum k_t with the respect to the clusters that were clustered or changed in the last step (because then, for all clusters we have to compute a new k_t), the complexity is $\mathcal{O}(N^3)$.

- **End**
  - If the user defined `cutkt` and at some point all k_t's are larger than cutkt stop the clustering process. Declare all of the remaining clusters as jets.
  - If the user defined an nmin and at some point we have exactly nmin jets stop the clustering process. Declare all of the remaining clusters as jets.

Bibliography

[1] M. Dasgupta, A. Fregoso, S. Marzani, and G. P. Salam, *JHEP* **1309** (2013) 029, arXiv:1307.0007 [hep-ph].

[2] S. Moretti, L. Lonnblad, and T. Sjostrand, *JHEP* **9808** (1998) 001, arXiv:hep-ph/9804296 [hep-ph].

[3] G. P. Salam, *Eur.Phys.J.* **C67** (2010) 637–686, arXiv:0906.1833 [hep-ph].

[4] J. Shelton, arXiv:1302.0260 [hep-ph].

[5] L. Lonnblad, *Comput.Phys.Commun.* **71** (1992) 15–31.

[6] **CDF Collaboration** Collaboration, A. Abulencia *et al.*, *Phys.Rev.Lett.* **96** (2006) 122001, arXiv:hep-ex/0512062 [hep-ex].

[7] **D0 Collaboration** Collaboration, V. M. Abazov *et al.*, *Phys.Rev.* **D85** (2012) 052006, arXiv:1110.3771 [hep-ex].

[8] **H1 Collaboration** Collaboration, I. Abt *et al.*, *Phys.Lett.* **B314** (1993) 436–444.

[9] **ZEUS Collaboration** Collaboration, J. Breitweg *et al.*, *Eur.Phys.J.* **C4** (1998) 591–606, arXiv:hep-ex/9802012 [hep-ex].

[10] **D0 Collaboration** Collaboration, V. M. Abazov *et al.*, *Phys.Rev.* **D80** (2009) 092006, arXiv:0904.3195 [hep-ex].

[11] **H1 Collaboration** Collaboration, F. Aaron *et al.*, *Eur.Phys.J.* **C65** (2010) 363–383, arXiv:0904.3870 [hep-ex].

[12] **ZEUS Collaboration** Collaboration, S. Chekanov *et al.*, *Phys.Lett.* **B547** (2002) 164–180, arXiv:hep-ex/0208037 [hep-ex].

[13] G. F. Sterman and S. Weinberg, *Phys.Rev.Lett.* **39** (1977) 1436.

[14] M. Bahr, S. Gieseke, M. Gigg, D. Grellscheid, K. Hamilton, *et al.*, *Eur.Phys.J.* **C58** (2008) 639–707, arXiv:0803.0883 [hep-ph].

[15] T. Sjostrand, S. Mrenna, and P. Z. Skands, *Comput.Phys.Commun.* **178** (2008) 852–867, `arXiv:0710.3820 [hep-ph]`.

[16] T. Gleisberg, S. Hoeche, F. Krauss, M. Schonherr, S. Schumann, *et al.*, *JHEP* **0902** (2009) 007, `arXiv:0811.4622 [hep-ph]`.

[17] A. Buckley, J. Butterworth, S. Gieseke, D. Grellscheid, S. Hoche, *et al.*, *Phys.Rept.* **504** (2011) 145–233, `arXiv:1101.2599 [hep-ph]`.

[18] J. E. Huth, N. Wainer, K. Meier, N. Hadley, F. Aversa, *et al.*

[19] G. P. Salam, `arXiv:0705.2696 [hep-ph]`.

[20] T. Sjostrand, *Comput.Phys.Commun.* **28** (1983) 229.

[21] **JADE Collaboration** Collaboration, W. Bartel *et al.*, *Z.Phys.* **C33** (1986) 23.

[22] **JADE Collaboration** Collaboration, S. Bethke *et al.*, *Phys.Lett.* **B213** (1988) 235.

[23] L. Lonnblad, `arXiv:hep-ph/9908368 [hep-ph]`.

[24] M. Cacciari, G. P. Salam, and G. Soyez, *Eur.Phys.J.* **C72** (2012) 1896, `arXiv:1111.6097 [hep-ph]`.

[25] R. Corke and T. Sjostrand, *JHEP* **1103** (2011) 032, `arXiv:1011.1759 [hep-ph]`.

[26] A. Abdesselam, E. B. Kuutmann, U. Bitenc, G. Brooijmans, J. Butterworth, *et al.*, *Eur.Phys.J.* **C71** (2011) 1661, `arXiv:1012.5412 [hep-ph]`.

[27] A. Altheimer, A. Arce, L. Asquith, J. Backus Mayes, E. Bergeaas Kuutmann, *et al.*, `arXiv:1311.2708 [hep-ex]`.

[28] "CGAL, Computational Geometry Algorithms Library.". http://www.cgal.org.