ALMA MATER STUDIORUM · UNIVERSITY OF BOLOGNA

School of Science
Department of Physics and Astronomy
Bachelor Degree in Physics

A New Trackster Linking Algorithm Based on Graph Theory for Reconstruction in HGCAL at the CMS Experiment

Supervisor:
Prof. Francesco Giacomini

Co-Supervisors:
Dr. Felice Pantaleo, CERN
Dr. Wahid Redjeb, CERN

Submitted by:
Camilla Lazzati

Academic Year 2023/2024

"[...] *Sempre devi avere in mente Itaca -*
raggiungerla sia il pensiero costante.
Ma non precipitare il tuo viaggio:
fa che duri a lungo, per anni, e che da vecchio
tu metta piede sull'isola, ricco
dei tesori accumulati per strada,
senza aspettarti ricchezze da Itaca.

Itaca ti ha donato il bel viaggio
senza di lei mai ti saresti messo
in cammino: nulla ha da darti più.

E se la trovi povera, Itaca non t'ha illuso
Fatto ormai savio, già tu avrai capito
cosa Itaca vuole significare."

Konstantinos Kavafis, *Iθακη*

A chi mi vuole bene

Abstract

Al termine del Run-3, il Large Hadron Collider verrà sottoposto ad una fase di aggiornamento secondo il progetto HL-LHC (High-Luminosity Large Hadron Collider) che prevede un aumento della luminosità di un fattore 7.5 rispetto a quella del Run-1. Ciò rappresenta una sfida sia per i rivelatori, sia per gli algoritmi di ricostruzione, a causa del forte aumento del numero di collisioni simultanee (*pileup*) e di dati da processare. La collaborazione CMS (Compact Muon Solenoid) ha pertanto progettato lo High Granularity Calorimeter (HGCAL), un nuovo calorimetro ad alta granularità che sostituirà quello attuale. Un nuovo framework per la ricostruzione di HGCAL, The Iterative Clustering Framework (TICL), è in fase di sviluppo. La struttura modulare di TICL prevede diverse fasi, tra cui il *linking*, che ha il compito di unire *cluster* tridimensionali di energia, detti *tracksters*, originati dalla stessa particella primaria. Questa tesi esplora un possibile approccio per effettuare il *linking* basato sulla teoria dei grafi e, in particolare, su un algoritmo detto di Leiden. L'algoritmo è stato implementato in C++ e integrato nel software di CMS. Le prestazioni dell'algoritmo, in termini di qualità della ricostruzione, sono state valutate tramite eventi simulati. L'algoritmo presenta una buona prestazione sia nel caso di singola particella sia nel caso di particelle vicine, riuscendo a distinguere le due particelle anche in caso di sovrapposizione. Tuttavia, mostra una tendenza a collegare troppo poco i trackster. Al momento l'algoritmo non include al suo interno le informazioni fisiche provenienti dai sensori: si ritiene che, con la loro introduzione, si potrà avere un maggiore controllo sul *linking* ed un significativo miglioramento nella qualità della ricostruzione.

Contents

List of Figures

Chapter 1

Introduction

1.1 The Standard Model

The **Standard Model** (SM) is a theory that encompasses the understanding of modern particle physics, describing elementary particles and the fundamental interactions that govern them.

Elementary particles are divided into **bosons** and **fermions** according to an intrinsic property called spin: particles with integer spin are bosons, while those with half-integer spin are fermions. In the SM there are twelve fermions and five bosons.

Bosons are the mediators of the interactions, i.e. an interaction between particles can be viewed as the exchange of a boson. Each boson is responsible for a specific interaction: the electromagnetic one is carried by the photon, the strong nuclear one by the gluons, and the weak nuclear one by the W and Z bosons. Each interaction has an associated charge that particles must have in order to participate in that interaction: electric charge for the electromagnetic one, color charge for the strong one, and weak charge for the weak one. The Higgs boson is the last particle in the Standard Model to have been discovered and it is responsible for the mechanism through which particles acquire mass. It is the only known elementary particle with spin 0. At the present moment the gravitational interaction is not incorporated within the Standard Model.

Fermions are the particles that make up matter. They are categorized into three distinct generations, i.e. groups of particles that exhibit similar behaviors but differ in mass. Each generation comprises two leptons and two quarks, the main difference being that quarks have a color charge, whereas leptons do not. For each of these particles, there exists a corresponding antiparticle with opposite charges. All the elementary particles comprised in the SM are summarised in Figure 1.1 [1].

The SM is a well-tested physics theory, it has successfully explained most experimental results and precisely predicted a wide range of phenomena. The Large Hadron Collider, which is the world's largest and most powerful particle accelerator, hosts exper-

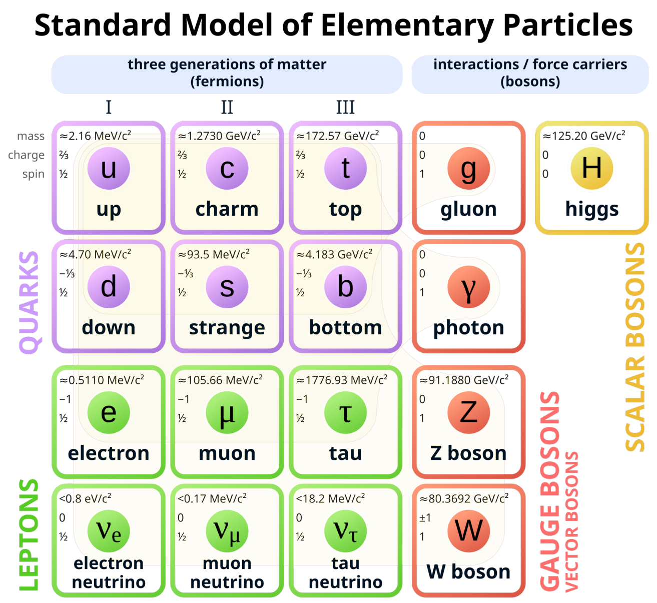

Standard Model of Elementary Particles
three generations of matter (fermions)
interactions / force carriers (bosons)
I
II
III
mass
charge
spin
≈2.16 MeV/c² ⅔ ½ u up
≈1.2730 GeV/c² ⅔ ½ c charm
≈172.57 GeV/c² ⅔ ½ t top
0 0 1 g gluon
≈125.20 GeV/c² 0 0 H higgs
≈4.70 MeV/c² −⅓ ½ d down
≈93.5 MeV/c² −⅓ ½ s strange
≈4.183 GeV/c² −⅓ ½ b bottom
0 0 1 γ photon
≈0.5110 MeV/c² −1 ½ e electron
≈105.66 MeV/c² −1 ½ μ muon
≈1776.93 MeV/c² −1 ½ τ tau
≈91.1880 GeV/c² 0 1 Z Z boson
<0.8 eV/c² 0 ½ νe electron neutrino
<0.17 MeV/c² 0 ½ νμ muon neutrino
<18.2 MeV/c² 0 ½ ντ tau neutrino
≈80.3692 GeV/c² ±1 1 W W boson
QUARKS
LEPTONS
SCALAR BOSONS
GAUGE BOSONS
VECTOR BOSONS

Figure 1.1: Standard model of elementary particles. Brown loops indicate which bosons (red) couple to which fermions (purple and green). Original by Wikimedia Commons, CC-BY-SA 3.0 [2].

iments that are among the main contributors of producing experimental confirmations of the SM. For instance, the predicted existence of the Higgs boson was confirmed by the observations made in 2012 by two detectors at the LHC. However, there are still important aspects left unexplained, such as gravity, dark matter, the asymmetry between matter and antimatter and the great difference in mass among the three generations. The experiments held at LHC play a key role in providing new information to answer these questions and in further testing the SM.

1.2 The LHC and the HL-LHC Project

Research at CERN (Conseil Européen pour la Recherche Nucléaire) focuses on studying the fundamental components of matter and the ways they interact. In order to carry out this research, CERN employs an accelerator complex, consisting of machines that accelerate particles to increasingly higher energies [3]. Each machine boosts the energy of a beam of particles before injecting it into the next one in the sequence, which starts from Linac4 (Linear accelerator 4) and ends with the LHC (Large Hadron Collider), as shown in Figure 1.2.

The LHC is a particle accelerator consisting of a 27-kilometre ring located in an underground tunnel in proximity to Geneva. The LHC accelerates protons in both

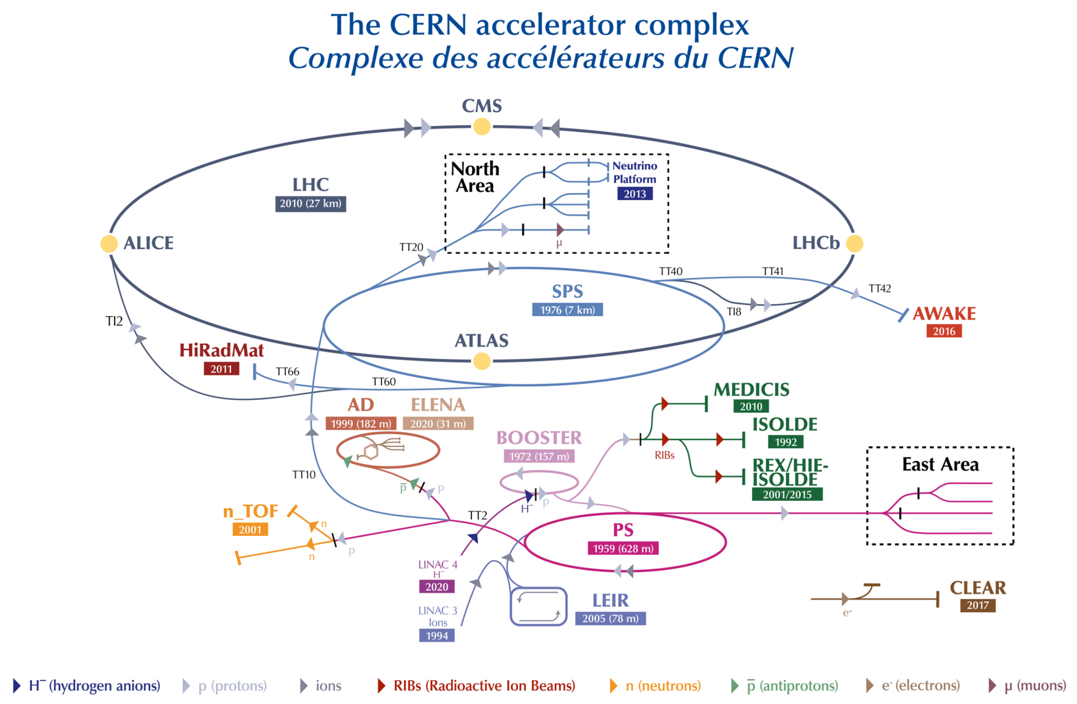


Figure 1.2: CERN accelerator complex. CC-BY CERN [4].

clockwise and anticlockwise directions and these beams are brought into collision inside four different detectors: ATLAS (A Toroidal LHC Apparatus) [5], CMS (Compact Muon Solenoid) [6], ALICE (A Large Ion Collider Experiment) [7], and LHCb (LHC-beauty) [8], all represented in Figure 1.2. The proton beams are structured in bunches, with each bunch containing around 10^{11} protons. From the proton-proton collision secondary particles are originated and spread in all possible directions. Alongside a wide array of low-energy, ordinary particles, heavier, undiscovered particles can also be produced. These massive particles cannot be observed directly: almost immediately they decay into lighter particles, which in turn also decay. The four detectors built around the collision points are needed to count, track, and characterize all the particles produced so as to provide the physical information necessary to reconstruct the event.

Event reconstruction is the process of interpreting the signals produced in a detector to determine which original particles passed through it and their characteristics.

ATLAS and CMS are general-purpose detectors, covering the widest possible range of physics at the LHC, from precision measurements of the Higgs boson to searches for new physics beyond the Standard Model. ALICE studies the properties of quark–gluon plasma, a state of matter where quarks and gluons are no longer confined inside hadrons. LHCb focuses on the study of the asymmetry between matter and antimatter.

Two key parameters of an accelerator are **beam energy** and **luminosity**. Beam

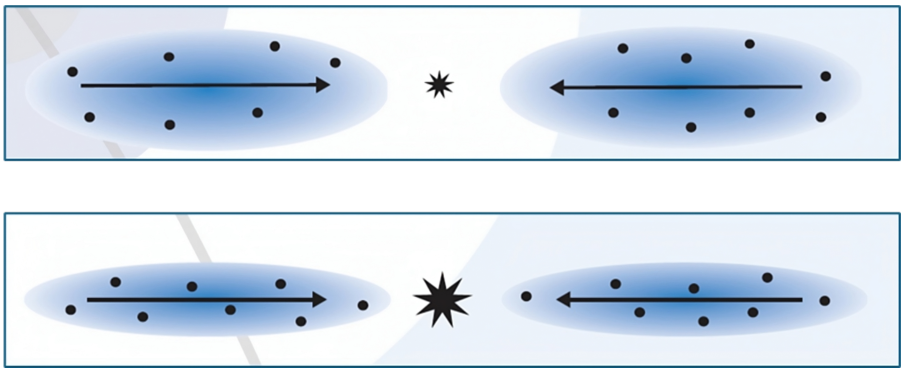


Figure 1.3: The same number of particles are bunched in the beams above and below. The more focused the beam is (below), the higher the number of collisions with particles in the other beam. CC-BY ATLAS Experiment © CERN.

energy determines the energy of the collision in the center of mass reference frame, which in turn establishes the maximum mass that a particle produced by that collision can have. Luminosity $\mathcal{L}$ is defined as the proportionality factor between the number of collisions per second dN/dt and the cross section σ [9]:

$$\frac{dN}{dt} = \mathcal{L} \cdot \sigma \tag{1.1}$$

and its unit of measure is thus cm^{-2}s^{-1}. The luminosity in the LHC is determined completely from the colliding beam properties. For bunched beams with Gaussian density distribution, it can be expressed as [9]:

$$\mathcal{L} = f \frac{N_b N_1 N_2}{4\pi \sigma_x \sigma_y} \tag{1.2}$$

where N_1 and N_2 are the number of particles in the bunches 1 and 2, respectively, N_b is the number of bunches, f is the revolution frequency, σ_x and σ_y are the standard deviations of the Gaussian density profiles $\rho_x(x)$ and $\rho_y(y)$, with x and y Cartesian coordinates such that the z axis is along the beam direction. In order to increase luminosity, one has thus to act on these beam parameters, for example by focusing the beam more tightly so as to diminish the standard deviations σ_x and σ_y, as shown in Figure 1.3. The cross section σ is characteristic of a process. Integrated luminosity is defined as:

$$\mathcal{L}_{\text{int}} = \int_0^T \mathcal{L}(t')\, dt' \tag{1.3}$$

The typical unit of integrated luminosity is the inverse femtobarn[1] fb^{-1}. Very rare processes have a very low cross section. In order to observe them, a high luminosity is therefore necessary.

[1]One barn equals 10^{-24}cm^2.

Pileup (PU) indicates the number of simultaneous collisions during a bunch crossing. The average pileup is defined as:

$$\langle PU \rangle = \frac{\mathcal{L}\sigma}{N_b f} \tag{1.4}$$

Most of these collisions are "soft" or "peripheral" interactions that do not contribute to the search for new physics. In contrast, a relatively small fraction of all collisions are "hard" collisions that produce high transverse momentum[2] particles [10]. Each detector is equipped with a **trigger system**, which selects only the potentially interesting collisions that will then undergo complete reconstruction.

The LHC is currently in Run-3, a new period of data taking began in July 2022 and it is operating at the maximum collision energy of 13.6 TeV. For Run-3, the average pileup is 64 PU and the peak luminosity is approximately 2×10^{34} cm^{-2}s^{-1} [11].

After the conclusion of Run-3, which is projected for 2025, the LHC and its associated detectors will undergo a massive upgrade for the High Luminosity LHC (HL-LHC) project [12]. The HL-LHC project aims to reach a peak luminosity of 7.5×10^{34}cm^{-2}s^{-1} and an integrated luminosity of 250 fb^{-1} per year, enabling the goal of reaching 3000 fb^{-1} twelve years after the upgrade. This integrated luminosity is about ten times the one reached by LHC in its first twelve years of lifetime, since 2011 when Run-1 began [13]. The HL-LHC is predicted to be operational from 2029 and it will allow researchers to test more accurately the Standard Model thanks to the increased volume of data. For instance, HL-LHC will produce at least 15 million Higgs bosons per year, compared to around 3 million from the LHC in 2017, allowing to obtain more precise measurements of the particle [14]. Additionally, it might lead to the observation of new, rare phenomena. However, it will also pose several challenges to the four detectors by creating a high-pileup environment. In fact, HL-LHC is expected to reach the value of 200 PU (see Figure 1.4). Reconstruction in a high PU environment is a difficult task which requires sophisticated algorithms. Additionally, as the data to process will increase drastically, substantial increase in computing requirements, both at the online and offline level, will be required.

Each of the four detectors will undergo hardware upgrades to prepare for the HL-LHC phase. In particular, the CMS detector will replace its current endcap calorimeters with a High-Granularity Calorimeter (HGCAL). In addition to having greater radiation tolerance, HGCAL will have unprecedentedly high granularity to capture fine details of particle showers. Reconstruction in HGCAL will be a particularly challenging task as, due to the high granularity, the reconstruction algorithm will have approximately 10^5 energy deposits as its input.

It is thus clear that the hardware upgrade carried out for HL-LHC needs to be accompanied by a software update to handle the increased complexity and higher PU. In particular, researchers from the CMS Collaboration are developing a new reconstruction

[2]Transverse momentum is the component of particle momentum orthogonal to the beam direction.

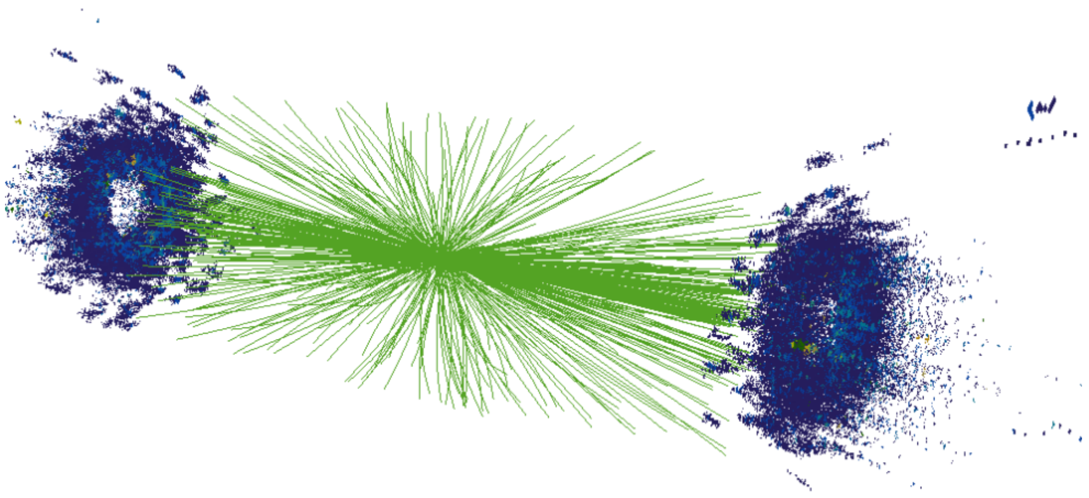


Figure 1.4: Image produced by the CMS Fireworks software, showing the simulation of a 200 PU particle collision event. The tracks are highlighted in green and the energetic showers are shown in blue.

framework for HGCAL.This new framework will need to satisfy the following requirements:

- **High computational performance** due to the large amount of data to be processed both online and offline
- **High quality reconstruction in a high PU environment**, to improve the physics capabilities and reach of the CMS collaboration
- **Performance scaling slowly with the number of inputs** in order to prevent a timing or memory explosion in high-occupancy environments.

Starting from this context, the present thesis explores the possibility of using graph theory to perform reconstruction in HGCAL, in particular by implementing a new algorithm for the linking[3] step using a community detection algorithm called the Leiden algorithm[4].

1.3 Thesis Structure

In Chapter 2, the structure of the CMS detector is presented, alongside the upgrades foreseen by the HL-LHC project. In Chapter 3, the framework under development for reconstruction in HGCAL is described, with a specific focus on the linking step, object

[3]See Chapter 3.

[4]See Chapter 4.

of the thesis work. In Chapter 4 the basics of graph theory and the Leiden algorithm are illustrated. Chapter 5 explains the implementation of the Leiden algorithm within the CMS Software. In Chapter 6, the response plots resulting from the simulations are shown. Lastly, Chapter 7 summarizes the outcomes of this thesis and outlines further research work that can be carried out to continue the present one.

Chapter 2

CMS: The Compact Muon Solenoid

2.1 CMS Detector

The CMS (Compact Muon Solenoid) detector is a general-purpose detector placed in one of the four interaction points at the LHC. It is *compact* compared to the ATLAS one, which is about 1.5 times the diameter of CMS and 2 times as long, but is only half the weight of CMS. The term *muon* is due to its excellent detection of muons, which are important signatures of interesting physics as they can essentially only come from the decay of heavy and thus possibly interesting particles. Lastly, it is called *solenoid* because it contains a big solenoid magnet taking the form of a cylindrical coil of superconducting cable that generates a field of 4T [15].

The CMS detector is nested cylindrically along the beam axis and consists of several sub-detectors, which are shown in Figure 2.1. CMS is built symmetrically around the **interaction point** (IP), where proton-proton collisions occur. The main sub-detectors will be briefly described in the following paragraphs, starting from the sub-detector closest to the IP going outwards.

2.1.1 Tracker

The closest sub-detector to the IP is the **tracker**, comprised of multiple concentric layers of silicon sensors. Charged particles ionize the semiconductor when passing through it, which produces electric signals, called **hits**, in the sensors. From the hits, the position is measured and from these measurements the trajectory, referred to as **track**, is reconstructed. Since charged particles are subject to the solenoid's magnetic field, their trajectory reveals the particle's charge, depending on the bending direction. The tracker is also extremely important for the accurate measurement of the particle momentum, obtained from the curvature of the trajectory.

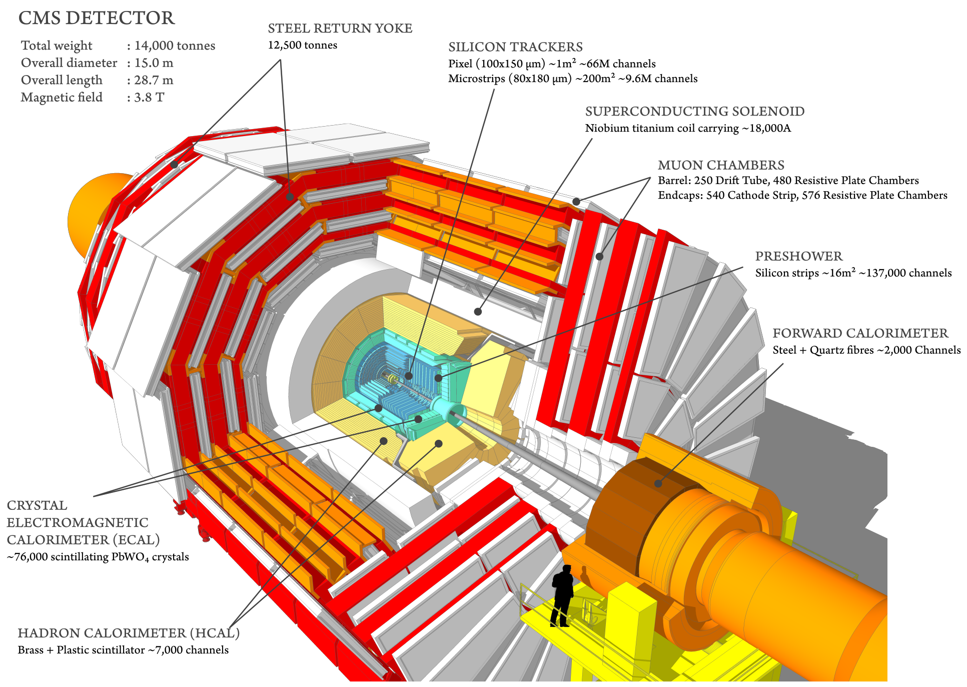

CMS DETECTOR
Total weight : 14,000 tonnes
Overall diameter : 15.0 m
Overall length : 28.7 m
Magnetic field : 3.8 T
STEEL RETURN YOKE
12,500 tonnes
SILICON TRACKERS
Pixel (100x150 μm) ~1m² ~66M channels
Microstrips (80x180 μm) ~200m² ~9.6M channels
SUPERCONDUCTING SOLENOID
Niobium titanium coil carrying ~18,000A
MUON CHAMBERS
Barrel: 250 Drift Tube, 480 Resistive Plate Chambers
Endcaps: 540 Cathode Strip, 576 Resistive Plate Chambers
PRESHOWER
Silicon strips ~16m² ~137,000 channels
FORWARD CALORIMETER
Steel + Quartz fibres ~2,000 Channels
CRYSTAL ELECTROMAGNETIC CALORIMETER (ECAL)
~76,000 scintillating PbWO₄ crystals
HADRON CALORIMETER (HCAL)
Brass + Plastic scintillator ~7,000 channels

Figure 2.1: The CMS detector concentric layers. CC-BY CERN.

2.1.2 Calorimeters

The subsequent sub-detector layers are the **calorimeters**, which measure the energy of particles by entirely halting them and absorbing their full energy. They are therefore able to detect neutral particles too, unlike the tracker. However, when high-energy particles interact with the dense calorimeter material, they create lower-energy secondary particles, which in turn create more particles, producing a cascade effect referred to as **particle shower**. There are two types of calorimeters in CMS: **ECAL** (Electronic Calorimeter) and **HCAL** (Hadronic calorimeter).

ECAL

The ECAL fully absorbs electrons and photons and slows down hadrons. It is a homogeneous detector, i.e. it consists of a singular material which plays the role of both absorber and detector. In particular, it consists of approximately 76000 lead tungstate crystals. These crystals scintillate, i.e. produce light, when particles pass through them. The light produced is detected by photodetectors positioned onto the back of each crystal and converted into an electrical signal for analysis. The ECAL consists of the barrel section and two endcaps, as well as a pre-shower system in front of each endcap.

HCAL

The HCAL measures the energy of hadrons. It is a heterogeneous detector, made up of alternating passive absorbing steel layers with active plastic scintillators layers. When particles pass through the active layer, the scintillators emit blue-violet light, in proportion to the particle's energy. The HCAL consists of the barrel and the two endcaps, located inside the superconducting solenoid, and the forward hadronic calorimeter extended beyond the magnet. Muons and neutrinos pass through both calorimeters.

2.1.3 Muon Chambers

After the calorimeters, there are the **muon chambers**, which are able to detect muons. The muon system is comprised of four distinct gaseous ionization detectors, whose functioning relies on the ionization of gas atoms by the passage of a charged particle [16]. Neutrinos escape CMS undetected, but their presence can be inferred by momentum conservation in the transverse plane. The negative sum of the transverse momenta of all the reconstructed particles, known as missing transverse momentum (MET), is in fact interpreted as the total transverse momentum of neutrinos or other hypothetical non-interacting particles.

2.2 The CMS Coordinate System

CMS employs two main coordinate systems, which are shown in Figure 2.2. The right-handed Cartesian coordinate system adopted by convention at CMS has the origin centered at the collision point, the x-axis pointing radially inward toward the center of the LHC, the y-axis pointing vertically upward and the z-axis is along the beam direction, pointing towards the Jura mountains. As an alternative coordinate system, CMS also utilizes spherical coordinates. The azimuthal angle ϕ is measured from the positive x-axis in the xy plane, while the polar angle θ is measured from the z-axis. The transverse momentum p_T of a particle can therefore be written as:

$$p_{\mathrm{T}} = \sqrt{p_x^2 + p_y^2} = p \cdot \sin\theta \tag{2.1}$$

where p is the module of the momentum [16]. In collider physics, instead of the polar angle θ, it is preferred to use the **pseudorapidity** η. This quantity is Lorentz invariant for massless particles and it is defined as:

$$\eta = -\ln\left(\tan\frac{\theta}{2}\right) \tag{2.2}$$

Pseudorapidity varies therefore from 0 at $\theta = \pi/2$ to ∞ at $\theta = 0$.

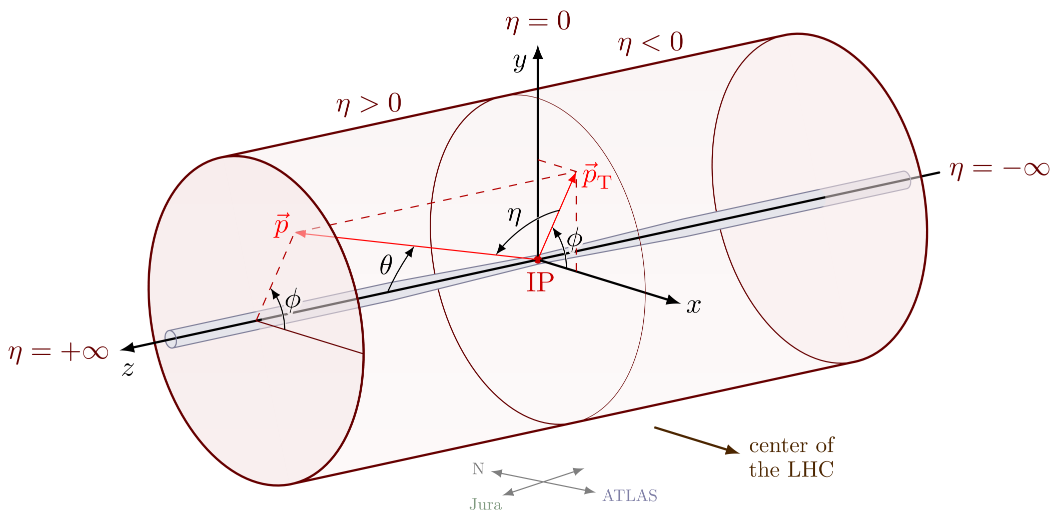


Figure 2.2: The CMS detector Cartesian and spherical coordinate systems. IP stands for interaction point. Programmed in `TikZ` by Izaak Neutelings [17].

The spatial separation between two particles ΔR can be expressed in terms of the azimuthal angle and pseudorapidity as:

$$\Delta R = \sqrt{(\Delta\eta)^2 + (\Delta\phi)^2}. \tag{2.3}$$

2.3 CMS HL-LHC Upgrade

The CMS Experiment is introducing several innovations to prepare for the challenges posed by the HL-LHC project [18]:

1. **Handle the high radiation environment**:
   - Replacement of the tracker and endcap calorimeter systems.
   - Major electronics updates of the barrel calorimeters and muon detectors.
2. **Handle the high PU environment**:
   - Improved granularity wherever possible, e.g. for the barrel and endcap calorimeters.
   - Introduction of a new precision timing detector called the Minimum Ionizing Particle (MIP) Timing Detector (MTD), placed in front of the barrel and endcap calorimeters. Time compatibility can be used as an additional information to improve particle reconstruction.
3. **Handle the high luminosity**:

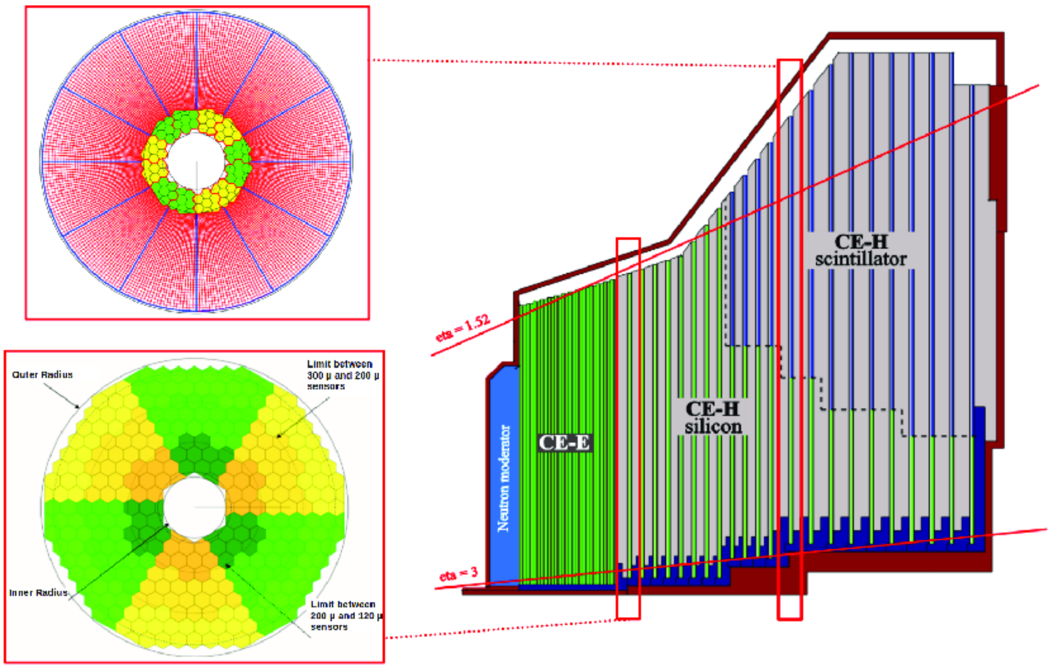


Figure 2.3: Schematic view of one HGCAL endcap slice (on the right), a representative hadronic calorimeter layer (top left), and electromagnetic calorimeter layer (bottom left). Silicon sensors are represented as yellow and green hexagonal cells, while scintillators as red mesh. CC-BY [19].

- A complete overhaul of the Trigger and DAQ systems[1] to deal with the increased data stream.

The endcap calorimeters will be replaced by a High-Granularity Calorimeter (HGCAL), which was designed by keeping in mind the challenges of high radiation tolerance and unprecedented pileup. As the focus of the present work is the reconstruction in HGCAL, a more detailed description of this sub-detector is provided in the following section.

2.3.1 HGCAL: the High Granularity Calorimeter

Two HGCAL endcaps will be mounted on both sides of the CMS detector, covering the forward region in the direction of the incident beams, where most interactions happen, in the range $1.5 < \eta < 3.0$. Each endcap will be divided in two compartments: an electromagnetic one called **CE-E** and a hadronic one called **CE-H**, which are shown in Figure 2.3.

The CE-E comprises 26 active hexagonal silicon layers interleaved with Cu, CuW and Pb absorbers. Each active layer is divided into 8-inch hexagonal sensors. CE-H has 7

[1]See Chapter 3.

full silicon layers and then 14 hybrid layers of scintillators and silicon sensors, where the silicon part becomes progressively smaller going outwards. Silicon and plastic are both radiation-tolerant materials. In total, HGCAL has around 6.5 million detector channels, which justifies the *high granularity* in its name [20].

Chapter 3

Event Reconstruction in CMS

3.1 Online and Offline Reconstruction

The overall collection of software in CMS is referred to as CMS Software (CMSSW). This software contains the reconstruction modules which perform online and offline reconstruction.

Both online and offline event reconstruction in CMS are based on the Particle Flow (PF) algorithm, which uses the information from all sub-detector systems to identify and reconstruct the comprehensive list of final-state particles (photons, electrons, muons, charged and neutral hadrons). The final global event interpretation takes the particle-flow elements within individual sub-detectors, including tracks and calorimeter clusters, as inputs. The tracks and clusters are then connected through a linking process [21].

Online Reconstruction Saving and analysing all the events generated at CMS is not possible because 40 million collisions take place every second. However, only a few of these collisions are considered physically interesting. Thus, a two-level trigger system has been introduced to select only significant events [22]. The trigger system plays a crucial role for the experiment, as it has to select which events to keep very quickly, but also carefully: if an event is discarded by the trigger, the data relative to it are lost forever. The first level trigger is called **Level 1 Trigger** (L1T) and the second one **High Level Trigger** (HLT).

The L1T is a fast, hardware-based trigger that reduces the data flow from 40 MHz to 100 kHz.

The HLT is a software-based trigger, after which the data rate is lowered to 7.5 kHz. The HLT performs a more accurate reconstruction compared to the L1T: it applies event reconstruction algorithms, similar to those used offline, but with a faster processing time and a lower reconstruction quality. Then, on the reconstructed objects, a set of filters called Trigger Paths is applied to select only the potentially interesting events.

Offline Reconstruction The events selected by the trigger are destined to long-term storage and offline analysis. The offline reconstruction is subject to less restrictive time constraints than the online one. However, even after the two trigger steps, the number of events to be analysed remains substantial. It is therefore important to have a high computational performance for the offline reconstruction as well. The following section will focus in more depth on the reconstruction in HGCAL, as this constitutes the context for the work carried out in the thesis.

3.2 Reconstruction in HGCAL

When a particle interacts with the calorimeter layers a shower of secondary particles is produced. Reconstruction in HGCAL in the high PU environment of the HL-LHC run is a difficult task due to the overlapping of particle showers, as can be seen in Figure 3.1, and to the high granularity which inherently brings higher computational demands. In order to tackle the HGCAL reconstruction task, a reconstruction framework, called The Iterative Clustering Framework (TICL), is currently under development [23].

3.2.1 TICL: The Iterative Clustering Framework

When a particle showers, the deposited energy is collected by the sensors on the layers that the shower traverses. Energy deposition in the detector is reconstructed by the local reconstruction, building the so-called hits, identified by the spatial coordinates, their energy, and the time assignment provided by HGCAL. These five-dimensional data points are called **recHits**. TICL takes the recHits as inputs and outputs a list of particle candidates, alongside their identification probabilities and kinematic properties [23]. The core steps of the framework are represented in Figure 3.2.

2D Pattern recognition: CLUE

TICL starts from the calibrated recHits and first clusters them within their respective detector layers using an energy-density-based algorithm known as CLUE, which stands for Clustering of Energy. CLUE takes as its input the n recHits and returns as its output k clusters, reducing the problem complexity by an order of magnitude. CLUE addresses one of the main challenges of HGCAL, i.e. the fact that the large number of channels results in an escalation of the computing load when clustering hits. CLUE is fully parallelizable and optimized for high occupancy scenarios. The CLUE algorithm aims to cluster points by detecting continuous high-density regions and consists of the following steps [24]:

a) each sensor cell on a layer is taken as a 2D point with an associated weight equal to its energy deposit value.

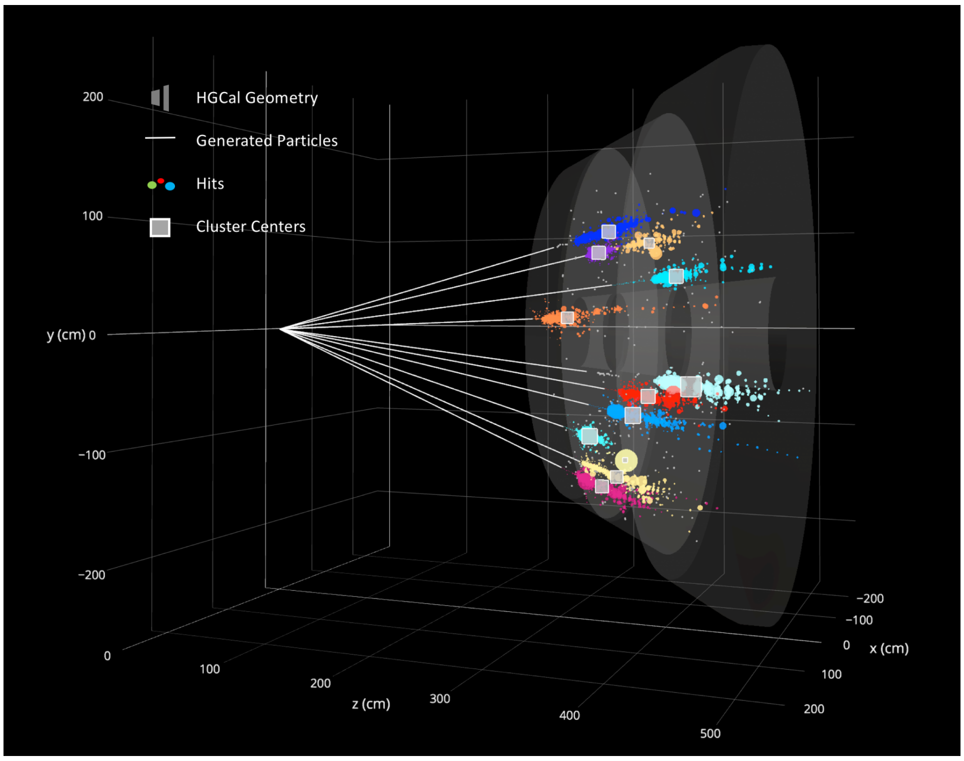


Figure 3.1: Colliding protons at the collision point generate secondary particles that travel towards the endcaps. When interacting with the calorimeter, these particles produce showers, which can be very close to each other and even overlap. Image courtesy of Ziheng Chen, Northwestern University.

b) for each point, the local density ρ is computed using the following definition:

$$\rho_i = \sum_{j \in N_{d_c}(i)} w_j \chi\left(d_{ij}\right)$$

where $N_{d_c}(i)$ indicates the neighbors of the hit i at the cut-off distance d_c, $\chi\left(d_{ij}\right)$ is the convolutional kernel (either a flat, Gaussian, or exponential function), d_{ij} is the distance between the hits i and j, and w_j the weight of the hit j given by its energy. The kernel plays the role of the weight in the sum and it decreases as d_{ij} increases.

c) for each point, the separation δ, which is the distance to the nearest point with higher energy density (nearest-higher), is calculated. If a point has no neighbours with higher density, then $\delta_i = +\infty$.

d) points with density $\rho > \rho_c$ and large separation $\delta > \delta_c$ are promoted to cluster seeds, while points with density $\rho < \rho_c$ and large separation $\delta > \delta_c$ are demoted to outliers, where $\rho_c, \delta_c, \delta_o$ are pre-defined cut-off thresholds.

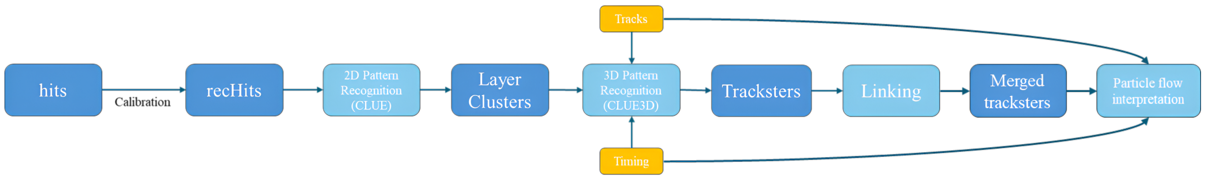

Tracks
hits
Calibration
recHits
2D Pattern Recognition (CLUE)
Layer Clusters
3D Pattern Recognition (CLUE3D)
Tracksters
Linking
Merged tracksters
Particle flow interpretation
Timing

Figure 3.2: The main steps in the TICL framework. Reconstruction algorithms correspond to the light blue boxes, while input/output objects correspond to the dark blue ones. Tracks and Timing (yellow blocks) are information provided by other CMS subdetectors. TICL starts from recHits and clusters them in **layer clusters** (LCs) through the clustering algorithm CLUE [24]. Then, **tracksters**, i.e. 3D clusters, are produced by applying CLUE3D [25] to the LCs. Tracksters produced by CLUE3D are usually only a fraction of the entire shower produced by a particle. This **fragmentation** is due to several factors, illustrated in Section 3.2.1. Tracksters thus need to be merged together through the linking step. From here, individual particle probabilities and properties are identified and are then fed to the rest of the Particle Flow chain.

e) For each point i, a list of followers is built containing the points which are neither seeds nor outliers and have the point i as their nearest-higher.

f) Each seed and its followers are iteratively grouped into separate clusters. Outliers and their descendant followers are excluded from forming clusters: this acts as a noise rejection by removing low-density deposits.

3D Pattern recognition: CLUE3D

CLUE3D takes as inputs the k layer clusters generated by CLUE and clusters them together across layers producing m 3D clusters, called **tracksters**, reducing the problem complexity by another order of magnitude. The algorithm follows a very similar procedure to CLUE [23]:

a) for each LC i, the k layers behind and in front of the layer where the LC i is are considered.

b) the local density ρ_i is computed by selecting all the LCs in these layers whose distance from the LC i is lower than a certain threshold Δ and then summing their energies multiplied by a kernel function. This kernel function acts as the weight in the sum, which decreases as the distance between layer i and the other LC considered increases.

c) for each LC, the separation δ, which is the distance to the nearest-higher, is calculated. If a point has no neighbours with higher density, then $\delta_i = +\infty$.

d) Based on ρ and δ, the algorithm identifies seeds, followers, and outliers in an analogous way to CLUE.

e) tracksters are yielded by aggregating seeds and their followers.

We define the **raw energy** of a trackster as the sum of the energies of all the LCs. To address inefficiencies in the clustering and mis-calibration, a Convolutional Neural Network is adopted to correct the final energy of clusters. The CNN input are constructed by building images starting from the LCs in the trackster, and it outputs the *regressed energy*, representing the best energy estimate obtainable for a reconstructed Trackster. The LCs discarded by CLUE3D as outliers are transformed in individual tracksters, called **Recovery** tracksters, containing one LC each.

Linking

Tracksters reconstructed by CLUE3D, in general, represent only a fragment of a larger shower. This **fragmentation** is due to several factors [26]:

- **Particle shower secondary components:** The CLUE3D algorithm tends to produce aligned showers. However, several processes can cause the formation of misaligned showers with secondary components. For instance, for electrons, showers initiated by the *bremsstrahlung*[1] can lead to the formation of secondary components. The same can happen for photons due to the process of *pair production*[2]. CLUE3D is able to reconstruct the individual electromagnetic components, but a linking step associating the components from the same initial shower (e.g. the electron and positron showers for the photon) is required. For hadronic showers, Minimum Ionizing Particles[3] (MIP) may be produced, which results in hadronic showers being much more spread out than the electromagnetic ones. This leads CLUE3D to split them into smaller tracksters.

- **Showers split at the border between CE-E and CE-H:** Hadrons leave part of their energy in the CE-E and the remaining of it in the CE-H. In this case CLUE3D often yields two separate tracksters (one in the CE-E and one in the CE-H) surrounded by small tracksters, especially in the CE-H part.

- **Deviating tracks:** The compartments of HGCAL have different material compositions for the absorbers as well as different sensors. If a secondary particle has a deviating track, this results in different energy densities, causing tracksters to be split at the boundary between the compartments.

- **High PU rejection tuning:** CLUE3D has been tuned for high pileup rejection, which means that the clustering parameters, such as Δ or ρ_c, are tight. The

[1]Emission of electromagnetic radiation caused by the deceleration of a charged particle passing through the electric field of another particle, typically an atomic nucleus.

[2]Pair production of a photon indicates the production of an electron and a positron.

[3]Particles whose mean energy loss rate through matter is close to the minimum.

fragmentation might be reduced by loosening the parameters, but this would also result in more PU being included in the tracksters, which could not be removed by any following step.

It is, therefore, necessary to introduce a **linking** step, which merges together tracksters belonging to the same particle shower, taking as its input both the CLUE3D tracksters and the Recovery ones.

The linking phase presents several criticalities, such as [27]:

- **Linking tracksters over long distances**
- **Reconstructing showers with highly irregular shapes**
- **Overlapping particle merging:** in a high PU environment, it is very common to have distinct particles producing showers that are very close, or even overlapping. The linking algorithm might, therefore, merge two showers into a single final trackster. This mistake has very negative consequences for reconstruction, especially for a neutral and a charged particle close to each other. The trackster will be in fact later matched with the track of the charged particle, losing the neutral one.

Chapter 4

Graph Theory and Community Detection Algorithms

4.1 Basic definitions

Graph theory is a branch of discrete mathematics that has become a fundamental tool in a wide and heterogeneous range of fields, from physics and biology to social sciences and linguistics [28, 29]. A graph can be formally defined as follows [28]:

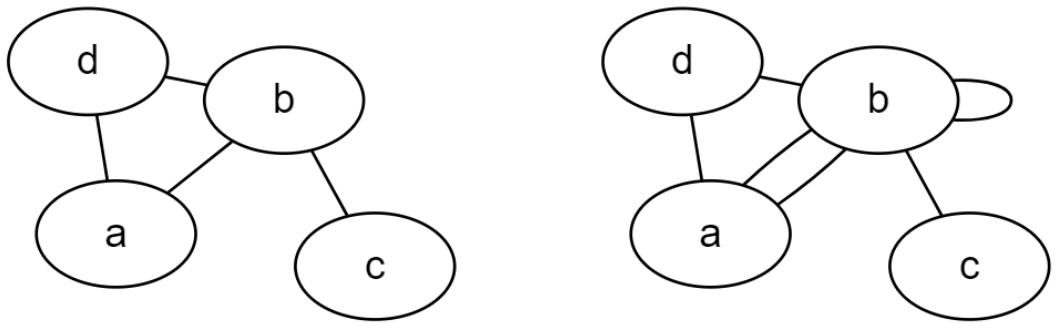
> A **graph** G is a pair $(V,\, E)$ where V is a non-empty finite set of elements called **vertices** and E is a finite family of unordered pairs of (not necessarily distinct) elements of V called **edges**. We call V the **vertex set** and E the **edge family** of the graph G.

Note that the previous definition allows the existence of multiple edges between the same pair of endpoints. In fact, the term **family** indicates a collection of elements, some of which may occur several times, as opposed to **set**, in which each element appears only once. Additionally, edges joining a vertex to itself, the so-called **loops**, are also allowed, as the elements in a pair are not necessarily distinct. A graph that does not contain multiple edges or loops is said to be a **simple graph** and can be defined as follows [30]:

> A **simple graph** G is a pair $(V,\, E)$ where V is a non-empty finite set of elements called **vertices**, E is a E is a subset of $\mathcal{P}_2(V)$ whose elements are called **edges** and $\mathcal{P}_2(V)$ is the set of all the 2-element subsets of V.

From the definition, it follows that every simple graph is a graph, but not every graph is a simple graph.

Two vertices u and v of G are said to be **adjacent** (to each other) if $\{u, v\} \in E$. The



(a) simple graph (b) graph with a multiple edge and a loop

Figure 4.1: Planar representation of two graphs.

vertices u and v are called the **endpoints** of this edge. If $v \in V$, the **neighbours** of v are the vertices of G that are adjacent to v [30]. The **degree** of a vertex is the number of edges with that vertex as an endpoint. A **walk** is a finite sequence $(v_0, v_1, ..., v_k)$ of vertices (with $k \geq 0$) such that $\{v_0, v_1\}, \{v_1, v_2\}, \{v_2, v_3\}, \ldots, \{v_{k-1}, v_k\}$ are edges of G. A **path** is a walk with no repeating vertices [30]. For two graphs $G_1 = (V(G_1), E(G_1))$ and $G_2 = (V(G_2), E(G_2))$, where $V(G_1)$ and $V(G_2)$ are disjoint, the **union** $G_1 \cup G_2$ is the graph with vertex set $V(G_1) \cup V(G_2)$ and edge family $E(G_1) \cup E(G_2)$ [28].

A graph G can be visually represented by drawing it on the plane. Vertices are usually represented by an oval (in which we put the name of the vertex) and edges by a curve that connects the two endpoints. The points' positions and the curves' shapes can be chosen freely, as long as they allow the reader to unambiguously reconstruct the graph from the picture [30]. Figure 4.1 shows the representation of two graphs, the one on the left being a simple graph.

4.1.1 Types of Graphs

In this section, the main types of graphs are presented [28, 30]:

- **Complete graph:** a simple graph in which every two distinct vertices are adjacent.
- **Empty graph:** a simple graph with no edges.
- **connected graph:** a graph that cannot be expressed as the union of two graphs.
- **Weighted graph:** a connected graph in which a non-negative number, called weight, is assigned to each edge.
- **n-th cycle graph:** for each $n > 1$, we define the n-th cycle graph C_n to be the simple graph $(\{1, 2, \ldots, n\}, \{\{i, i+1\} \mid 1 \leq i < n\} \cup \{\{n, 1\}\})$.

- **Regular graph:** a graph in which all vertices have the same degree. Note that a loop by convention increases the degree by 2.

4.2 Community Detection Algorithms

Graphs representing real systems often display big inhomogeneities. Globally, we see vertices with a high number of neighbors coexisting with vertices with very few neighbors. The distribution of edges is also locally inhomogeneous, with high concentrations of edges within special groups of vertices and low concentrations between these groups. This characteristic is called **community structure**. Vertices within the same community probably share common features and/or have similar roles within the graph [29]. In Figure 4.2, a graph with a community structure is represented.

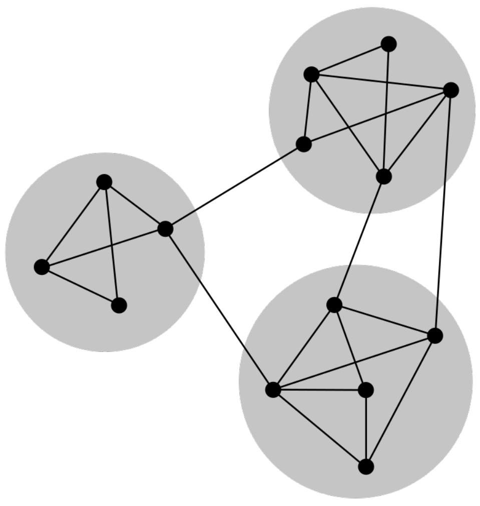


Figure 4.2: Representation of a graph with a community structure. Communities are represented by grey circles. Nodes within a community are densely connected, while nodes belonging to different communities are only sparsely connected. Original by Wikimedia Commons, CC-BY-SA 3.0 [31].

Community detection algorithms aim to identify the communities within a graph. These algorithms produce a **partition** of the initial graph as their output. A partition $\mathcal{P} = \{C_1, \ldots, C_r\}$ is a set of communities in which each community $C_i \subseteq V$ consists of a set of nodes such that $V = \bigcup_i C_i$ and $C_i \cap C_j = \emptyset$ for all $i \neq j$.

In the literature, a wide variety of community detection algorithms can be found. The following sections examine two algorithms relevant to the present work: the **Louvain Algorithm** and the **Leiden Algorithm**. The former is one of the most popular and well-known community detection algorithms, whilst the latter was developed more recently in an attempt to solve some of Louvain's criticalities and is the one that has been applied to the reconstruction problem in the present study. Both algorithms are based on the optimization of a **quality function** called **modularity**, which measures the quality of a partition.

4.2.1 Modularity

The **modularity** of a partition is a scalar value indicating the density of edges within communities as compared to edges between communities. For weighted graphs, it is defined as:

$$Q = \frac{1}{2m} \sum_{i,j} \left[A_{ij} - \frac{k_i k_j}{2m} \right] \delta\left(c_i, c_j\right) \tag{4.1}$$

where A_{ij} is the weight of the edge between i and j, $k_i = \sum_j A_{ij}$ is the sum of the weights of the edges having vertex i as an endpoint, $m = \frac{1}{2} \sum_{ij} A_{ij}$, c_i is the community to which vertex i belongs and $\delta\left(c_i, c_j\right)$ is 1 if the two vertices i and j belong to the same community and 0 otherwise. For unweighted graphs, where the weight is 1 for every edge, Equation 4.1 can be written as [32]:

$$Q = \frac{1}{2m} \sum_{c} \left(2e_c - \frac{K_c^2}{2m} \right), \tag{4.2}$$

where e_c is the number of edges within community c, K_c is the sum of the degrees of the nodes in the community c, and m is the total number of edges in the network. For unweighted graphs the modularity value lies in the range $[-\frac{1}{2}, 1)$. The modularity of a graph is 0 for the trivial case in which all vertices are in the same community. The first term in Equation 4.2 represents the fraction of the edges in the network that connects vertices within the same community. The second term represents the expected value of the same quantity in a network with the same community divisions but random connections between the vertices. Therefore, a positive modularity indicates that the number of within-community edges is greater than random. In practice, values higher than 0.7 are rare [33].

4.2.2 Louvain Algorithm

The Louvain Algorithm was proposed in 2008 by Blondel et al. [34] from the University of Louvain (thus the algorithm's name). The algorithm is based on the optimization of the modularity of a partition. Note that the algorithm was originally proposed with modularity optimization, but it can also be used with other quality functions.

The algorithm, illustrated in Figure 4.3 consists in two steps which are repeated iteratively [34]:

- **Moving nodes:** The initial partition is the **singleton partition** of the graph. This means that each node is assigned to a different community so that each community contains a single node. For each node i, its neighbors j are considered, and the change in modularity resulting from moving i from c_i to c_j is calculated for every neighbor j. The node i is moved to the community producing the greatest change, provided that this change is positive. Nodes are then re-visited sequentially and repeatedly until no more node movements increasing the quality function are possible.

- **Aggregation:** The communities obtained in the first step become the nodes in the aggregate graph. For a weighted graph, the weights of the edges between the aggregate nodes are given by the sum of the weights of the edges between nodes in the two corresponding communities. For an unweighted graph, the number of edges between two aggregate nodes equals the number of edges between nodes in the two corresponding communities. Edges between nodes of the same community become self-loops for this community in the aggregate graph. For a community of aggregated nodes, the size of the aggregated community is calculated recursively as the sum of the sizes of the communities that became aggregated nodes.

After completing the second step, the process is repeated for the aggregate graph. The iteration stops when no further improvements can be made, i.e., after the moving phase, each community consists of only one node [34].

Badly connected communities

The Louvain algorithm, while having several advantages such as being particularly fast on large networks compared to other community detection algorithms [34], might produce arbitrarily **badly connected** communities and even **internally disconnected** ones [35]. A community is defined as badly connected in [35] if, when running the Leiden algorithm (see Section 4.2.3) on the sub-network of the nodes belonging to that community, multiple sub-communities are found. A community is defined as internally disconnected if one part of that community can reach another part only through a path going outside the community. Disconnected communities can be seen as the extreme manifestation of the problem of arbitrarily badly connected communities [35]. Figure 4.4 shows an example of how the Louvain algorithm might produce a disconnected community. This problem occurs quite often in practice: for instance, after the first iteration of the Louvain algorithm, the average percentage of badly connected communities on large empirical networks ranges from 14% to 23% [35].

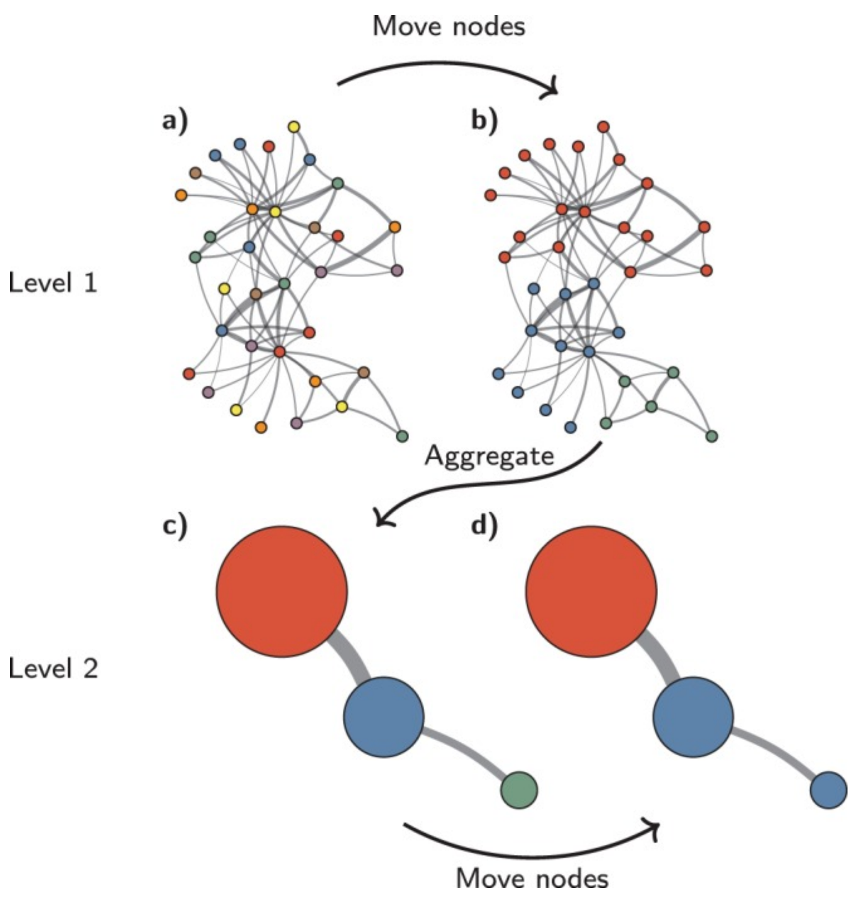


Figure 4.3: Illustration of the Louvain algorithm. The algorithm starts from a singleton partition, represented by the fact that each node has a different color (a), and then moves nodes to obtain the maximum gain in modularity (b). The communities become nodes in the aggregation step (c). The process then starts over, with the nodes in the aggregate graph being moved (d). CC-BY [35].

4.2.3 Leiden Algorithm

An algorithm addressing the issue of badly connected communities was developed in 2019 by Traag et al. [35] from the University of Leiden and was thus called the Leiden Algorithm. This algorithm is guaranteed to produce communities that are internally connected and it is faster than the Louvain algorithm. Firstly, the operations performed by the algorithm will be explained; then, the advantages it provides compared to the Louvain algorithm will be illustrated.

How the algorithm works

The Leiden algorithm can be divided into three steps:

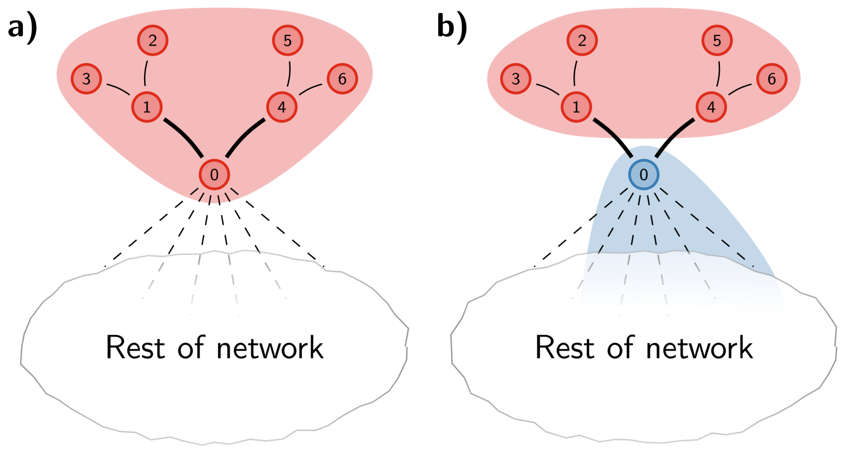


Figure 4.4: Example of how the Louvain algorithm may yield a disconnected community. The thickness of an edge is directly proportional to the weight assigned to that edge. Nodes 0–6 are initially in the same community. Nodes 1–6 have edges only within the community, while node 0 has many external edges as well (a). When node 0 is considered for moving, if there are sufficient neighbors of node 0 forming a community in the rest of the network, node 0 might be moved to this external community (b). Nodes 2, 3, 5 and 6 have only internal connections, and so will remain in the red community. Nodes 1 and 4 now also have external edges, but depending on the weights, they may still remain in the red community. In this case, a disconnected community is produced. CC-BY [35].

- **Fast node move:** The initial partition is the singleton partition of the graph. All the nodes in the network are added to the queue in random order. The node from the front of the queue is considered, and for every community, including the empty one, the variation of the quality function happening when the node is moved to that community is calculated. The node is then moved to the community producing the maximum increase. If the node has been moved, all the neighbors of the node that are not in the node's new community nor in the queue are added to the rear of the queue. This step continues until the queue becomes empty, and a partition P is produced as a result.

- **Refinement:** This step produces a **refined partition** P_{ref} based on the partition obtained from the previous step. P_{ref} is initially a singleton partition. For each community C in P, nodes are visited in random order. If the node is in a singleton community in P_{ref}, the other communities C' in P_{ref} which are contained in C are considered. The node is moved to one of the communities for which the move produces a positive change in the quality function. Note that in this step nodes are not merged with the community producing the largest increase: the community is selected randomly following a probability distribution in which the probability is higher for communities producing a larger increase in the quality function. In

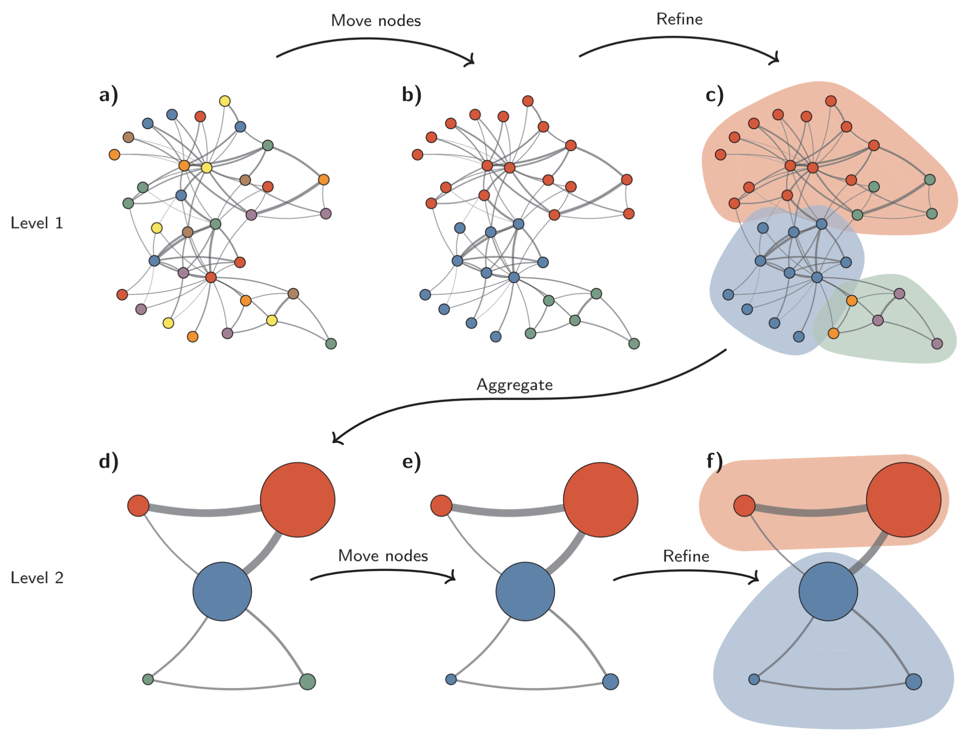


Figure 4.5: Illustration of the Leiden algorithm. The algorithm starts from a singleton partition (a). The fast move of nodes is then performed (b), and the partition obtained is then refined (c). Each community of the refined partition becomes a node in the aggregate graph network, which maintains the non-refined partition as its initial partition (d). These steps are then repeated on the aggregated graph (e-f). Note that in f), the refined partition is equal to the non-refined one. CC-BY [35].

particular, the probability of the community i is calculated as w_i/S, where S is a normalisation constant and w_i is the weight defined as:

$$w_i = \begin{cases} \exp\left(\frac{1}{\theta}\Delta Q_{\mathcal{P}}(v \mapsto C_i)\right) & \text{if } \Delta Q_{\mathcal{P}}(v \mapsto C_i) \geq 0 \\ 0 & \text{otherwise} \end{cases} \tag{4.3}$$

where $\Delta Q_{\mathcal{P}}(v \mapsto C_i)$ is the variation in modularity resulting by moving node v to community C_i and θ is a parameter indicating the degree of randomness. At the end of this step, communities in P may be split into several sub-communities in P_{ref}.

- **Aggregation:** the communities of P_{ref} become nodes in the aggregate graph. Note that the partition from which the local move of aggregate nodes starts will be P.

These steps are illustrated in Figure 4.5.

Advantages

The Leiden algorithm improves the Louvain one in two ways: it is faster and it produces communities guaranteed to be connected [35].

The algorithm is faster thanks to the introduction of a fast local move phase. Initially, all nodes are visited once, the same as in the Louvain algorithm. After this, however, Louvain revisits all the nodes, while Leiden revisits only the nodes whose neighborhoods have changed.

The other main difference is the introduction in the Leiden algorithm of randomness instead of **greedy move sequences**. With greedy move sequences, we indicate that the moving of nodes to the community produces the largest increase in the quality function. This is what is done by the Louvain algorithm. During the refinement step, the Leiden algorithm performs a **non-decreasing move sequence**, i.e., assigns the node randomly among the communities that yield a positive increase in the quality function, as described in the previous paragraph. As demonstrated in [35], reaching some optimal partitions with a greedy move sequence is not possible, while by using a non-decreasing move sequence it is always possible to reach the optimal partition. This can be intuitively explained by the fact that randomness allows the partition space to be explored more broadly.

Thanks to the randomness embedded in the algorithm, Leiden provides a series of additional guarantees, for instance [35]:

- **γ-connectivity**, which ensures that all communities are internally connected.
- **subset optimality**, which is the strongest guarantee provided. It means that each subset contained in the community is **locally optimally assigned**, i.e. moving it to any other community (including the empty one) would never result in an increase in the quality function.

Chapter 5

Implementing Leiden Algorithm for the Linking Phase

This chapter aims to illustrate how, in this thesis, graph theory and, in particular, the Leiden algorithm described in Chapter 4 were applied to perform the linking phase presented in Chapter 3.

The linking problem consists in finding an assignment in which tracksters coming from the same particle are connected, while the ones coming from different showers are kept separate. The key idea of the approach explored in this thesis is that **the trackster linking problem can be viewed as a community detection problem** for a graph in which each trackster is a node.

5.1 Graph Construction and Implementation

In the implementation, a trackster is described by the user-defined type `Trackster` containing as data members the coordinates of the trackster's barycenter and its raw energy. The function `TICLGraphProducer`, implemented in CMSSW, creates a graph by associating each trackster to a node and by creating the initial edges in the following way:

- The transversal section of HGCAL is divided into tiles, which are represented by a user-defined type `TICLLayerTileT`. Tiles are obtained by dividing into bins the continuous range of values the coordinates η and ϕ can assume.
- Each tile is filled with all the tracksters whose barycenter falls within it.
- For each trackster t, a search window, defined by four bin indexes, indicating the minimum and maximum bin for η and ϕ, is opened. The dimension of the window can be regulated through a parameter.

- All the tracksters who belong to a tile in the search window are added to the neighbors list of the node corresponding to t.

The steps defined above are repeated separately for the two endcaps, distinguished by the sign of the coordinate η.

In order to build the graph, *ad hoc* data structures were implemented in `C++` into CMSSW, such as the classes `TICLGraph`, `Community`, `Elementary`. The latter represents a single trackster and contains the trackster index in the collection. A `Node` is defined as an `std::variant`, i.e. a discriminated union, of an `Elementary` or a `Community`. A `Community` contains an `std::vector<Node>`. The choice of using `std::variant` is motivated by the intention of making the data structure suitable for the aggregation step described in Chapter 4.

5.2 Leiden Algorithm Implementation and Outputs

The Leiden algorithm was implemented into CMSSW by following the pseudo-code shown in [35]. In order to conform to the CMSSW established interface for linking, it produces as its outputs:

- `std::vector<Trackster>`: contains the merged tracksters, i.e. objects of type `Trackster` which are the output of a function `mergeTracksters`. This function receives as input all the tracksters belonging to the same community and creates a new trackster from the LCs of all the input tracksters. The raw energy of the merged trackster is the sum of all the raw energies of the input tracksters.

- `std::vector<std::vector<TracksterID>>`: each element contains the trackster identification indexes of the input tracksters that were merged together.

Chapter 6

Simulation and Results

6.1 CMS Simulation Software

The performance of the linking algorithm was tested by running the algorithm on simulated events. The simulated events were generated using the CMS Simulation Software [36], which in turn employs Pythia8 for hadronization [37] and GEANT4 to simulate the interaction of particles with matter [38]. The main parameters of the simulation are:

- **Number of events**
- **Number of particles** for each event.
- **Particle ID** which identifies the type of particle according to the Monte Carlo Particle Numbering Scheme [39].
- **Energy interval**, i.e. the minimum and maximum energy that a particle can have.
- η **interval**, i.e. the minimum and maximum value of the η coordinate that a particle can have, where η indicates the pseudorapidity[1].
- **Minimum distance** at which two different particles can be generated, expressed adimensionally as a function of η and ϕ, as defined in Chapter 2.
- **Overlapping**, which, if set to true, allows the showers of two particles to overlap.

From the simulated events, **RecoTracksters** and **SimTracksters** are obtained. A SimTrackster represents the best possible reconstruction that can be made starting from the 2D Layer clusters as the input. It is a hybrid Monte Carlo Truth information since

[1]See Chapter 2 for the definition of the coordinate system.

it exploits the true information together with reconstructed objects (Layer Clusters). A RecoTrackster is a trackster obtained by running CLUE3D on the layer clusters and subsequently applying the linking algorithm. Regressed energy for RecoTracksters and raw energy have already been defined in Chapter 3. For SimTracksters, regressed energy is simply the value of the energy parameter used when generating that simulated event. The reason why SimTracksters are created is that they convey the (hybrid) truth information in the same data format as the RecoTrackster, thus allowing to perform a direct comparison.

For each event, the SimTrackster is associated to the RecoTrackster which best matches it. The best match is calculated based on an association score called **Sim-To-Reco** [40], which measures the degree of overlap between hits in the detector belonging to a specific SimTrackster and the corresponding RecoTrackster. As seen in Chapter 3, hits are clustered into LCs; therefore, in order to calculate the Sim-To-Reco score for tracksters, one first has to compute the following:

$$fr_i^s = \frac{\sum_{h \in i} fr_h^s \cdot E_h}{E_i} \tag{6.1}$$

where fr_i^s is the shared energy between each LC i and each SimTrackster s, E_h is the energy associated to the hit h, fr_h^s is the fraction of the hit energy deposited by the SimTrackster s and E_i is the total energy of the LC i.

The fraction fr_i^s can go from 0 to 1, where 0 indicates that there is no overlap of LC i hits with the SimTrackster s. This is then used to compute the Sim-to-Reco score between the RecoTrackster t and the SimTrackster s:

$$\text{score}_{s,t} = \frac{\sum_{i \in s} \min\left(\left(fr_i^t - fr_i^s\right)^2, \left(fr_i^s\right)^2\right) \cdot E_i^2}{\sum_{i \in s} \left(fr_i^s\right)^2 \cdot E_i^2} \tag{6.2}$$

where i indexes the LCs of the SimTrackster s, fr_i^t and fr_i^s are the energy fractions of LC that have been assigned to the RecoTrackster t and to the SimTrackster s, respectively and $\sum_{i \in s} \left(fr_i^s\right)^2 \cdot E_i^2$ represents the square of the total energy of the SimTrackster s. This score measures how accurately a SimTrackster s is represented by the RecoTrackster t, with 0 indicating a perfect match and 1 indicating no matching [26].

6.2 Response Plots

Response plots are a useful tool for assessing the quality of the reconstruction performed by algorithms. If t is the RecoTrackster associated by the best score to the SimTrackster s, the response is calculated as:

$$r = \frac{E_t}{E_s} \tag{6.3}$$

where E_t is the raw energy of the selected RecoTrackster t and E_s is the generated energy of the Particle s. The raw energy of a final trackster is calculated as the sum of the raw energies of the original tracksters which have been linked. For a SimTrackster, r is defined analogously. Response plots are obtained by graphing the number of entries as a function of the response value r.

6.3 Results

The algorithm was run on two different types of samples: a single pion and two close pions with the possibility of overlapping. The pion was chosen as the particle type because hadron reconstruction is the one who needs the liking step the most. In fact, hadrons tend to produce very fragmented showers, which makes them very difficult to reconstruct. For both events, response plots were produced to analyse the algorithm behaviour.

For each of the two kinds of events, 2000 samples were generated. The chosen values of the algorithm parameters were $\gamma = 1$ and $\theta = 0.01$. The single pion events were generated using the following values of the parameters:

- **Energy interval:** $[25, 400]$ GeV
- **η interval:** $[1.7, 2.7]$

The response plot obtained from the RecoTracksters is shown as the red line in Figure 6.1. The SimTrackster response plot and the CLUE3D one are also shown for comparison purposes. The CLUE3D plot, alongside the Recovery tracksters defined in Chapter 3, represents the input of the Leiden linking algorithm.

The double-pion events were generated using the following values of the parameters:

- **Energy interval:** $[25, 400]$ GeV
- **η interval:** $[1.7, 2.7]$
- **Minimum distance:** 30
- **Overlapping:** true

The response plot obtained from the RecoTracksters is shown as the red line in Figure 6.2, alongside the SimTrackster and the CLUE3D ones.

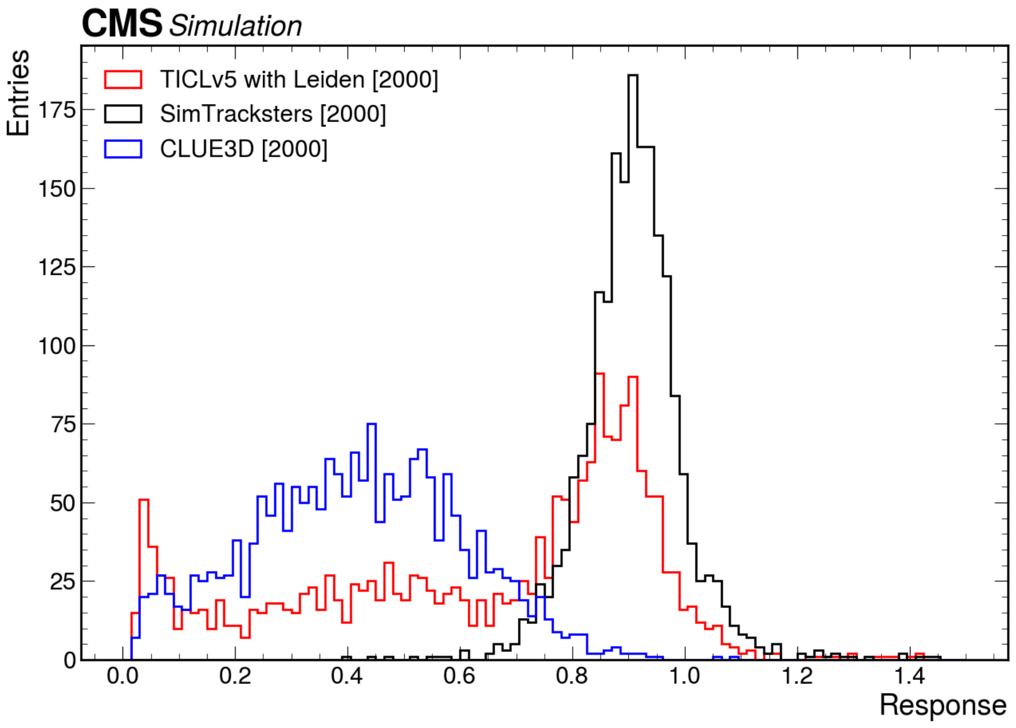


Figure 6.1: Response plot obtained for a single pion. The red line represents the response plot obtained from the RecoTracksters, the black line is the one obtained from the SimTracksters, and the blue line is the one obtained from CLUE3D without performing any linking afterwards.

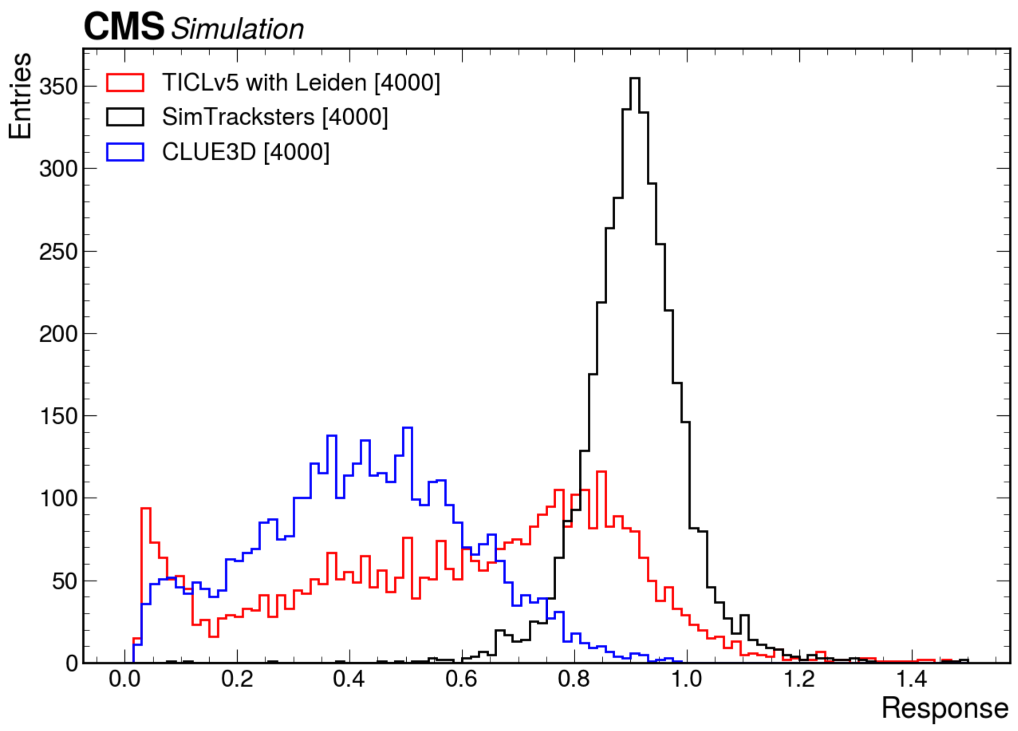


Figure 6.2: Response plot obtained for two pions with possibility of overlapping. The red line represents the response plot obtained from the RecoTracksters, the black line the one obtained from the SimTracksters and the blue line is the one obtained from CLUE3D without performing any linking afterwards.

6.3.1 Considerations

Several observations can be made from the response plots obtained. The CLUE3D plots confirm the necessity of introducing a linking step, as the response values obtained are too low. The SimTrackster peak shows the best achievable reconstruction that can be obtained starting from the Layer Clusters. The response of the SimTrackster is not centered around 1 because of missing calibration of the detector, leakage of shower out of the sector, lost energy contribution from the recHits that has not been clustered in the Layer Clustering step. The RecoTracksters plots reveal the performance of the Leiden algorithm. In order to assess it, the desired response for a linking algorithm will first be clarified.

The best possible performance theoretically obtainable is represented by a response r peaked around 1. However, while linking can certainly improve it, additional steps after linking will be needed to obtain an ideal response. For instance, a regression performed by a neural network should be applied to the raw energy at the numerator of r in order to obtain a better response. Therefore, a reconstruction is considered of high quality if the resulting response plot presents a peak close to the SimTrackster one. From the RecoTracksters plot we can see that, by introducing the linking phase using Leiden, a significant improvement is obtained. In both plots, a peak close to the SimTrackster one is clearly visible. Additionally, for the double pion plots, the tracksters of the two particles are not merged, and this is certainly a strength of the algorithm as incorrect merging is a critical risk of the linking phase. However, both for the single and the double pion, the plot presents a smaller peak at low response values, below $r = 0.2$. This shows that the algorithm is not linking the tracksters enough, in particular the smaller ones, i.e. the **Recovery** tracksters defined in Chapter 3. This results in having a non-negligible number of output tracksters that contain only a small fraction (below 20%) of the total energy corresponding to the particle.

The fact that Leiden is not linking enough tracksters can be understood considering that the present implementation does not provide to the algorithm any physical data coming from the detector sensors and from other reconstruction phases. These include, for instance, the time associated with each input trackster, the physical distance between tracksters, the energetic compatibility with and the distance from the reconstructed tracks (for charged particles). It is, therefore, plausible to assume that, when introducing the physical information into the algorithm, one will have better control over which tracksters will be linked, and an improvement in the linking performed by the algorithm is expected.

Chapter 7

Conclusions and Future Work

The HL-LHC project will entail a massive hardware upgrade of the collider and of the experiments' detectors. The CMS detector upgrade will include the construction of a High-Granularity Calorimeter (HGCAL) for its endcaps. This hardware upgrade needs to be accompanied by a software upgrade, to face the increased data rate and to perform reconstruction in a high PU environment. One of the key steps necessary to improve reconstruction is the linking step, performed after the three-dimensional clustering phase.

The present work explored the possibility of using community detection algorithms from graph theory to perform the linking step. In particular, the Leiden algorithm was implemented into CMSSW and applied to a graph constructed from the input tracksters. The performance of the algorithm was then tested by running it on simulated events of a single pion and of two close pions. The response plots were significantly improved after applying the Leiden algorithm. Additionally, when applied to the reconstruction of two close pions, the algorithm succeeded in avoiding merging the two showers. However, significant improvements can still be made.

First of all, the algorithm tended to perform less linking than necessary, resulting in the response plots having a smaller peak for $r < 0.2$. This issue can be addressed by introducing physical data available from the detector or from other reconstruction steps into the algorithm. There are two ways in which this can be done:

a) **Weighted graph:** a weight taking into account the physical compatibility can be defined and used when building the graph. For instance, the weight of an edge will be higher if the two tracksters linked have a very small physical distance or have very similar times associated to them. Then, the Leiden algorithm should be applied, optimizing modularity for the weighted graph.

b) **New quality function:** a new quality function taking into account the physical information can be defined and optimized instead of modularity, which is based purely on the number of nodes and edges.

Furthermore, the quality of the graph given in input to the Leiden algorithm needs to be assessed. If two tracksters that are supposed to be merged in the simulated event are not connected in the initial graph, then the Leiden algorithm will not be able to link them properly.

Additionally, the algorithm relies on the parameters γ and θ, which govern the refinement step. For this thesis, the values suggested in [35] have been used. However, a tuning of the parameters should be performed to obtain the best response possible. This tuning should be performed in a high PU environment (200 PU), which correctly reproduces the operating conditions of HGCAL in the HL-LHC upgrade.

Lastly, the computational performance of the algorithm needs to be analyzed. As seen in Chapter 4, the Leiden algorithm has proven to be faster than other famous community detection algorithms such as the Louvain. However, a detailed analysis of its performance within CMSSW should be carried out, and if necessary, strategies to improve such performance should be experimented with and introduced.

Acknowledgements

I would like to thank my supervisor, professor Francesco Giacomini, for making me passionate about coding through his lessons, for following me actively during my thesis, providing a lot of insightful observations, and for introducing me to the Patatrack research group. I would also like to thank my co-supervisors from the Patatrack team, Dr. Felice Pantaleo and Dr. Wahid Redjeb, for their patience, genuine interest and constant support. I am very grateful for the people I met during this three-year journey through the Physics Bachelor and the Collegio Superiore, for creating a collaborative and stimulating environment. Lastly, I would like to thank my family, for always doing everything they can to support me in reaching my goals.

Bibliography

[1] CERN. *The Standard Model.* https://home.cern/science/physics/standard-model. Accessed: 09/06/2024.

[2] Original by User:Cush.

[3] CERN. *CERN's accelerator complex.* https://home.cern/science/accelerators/accelerator-complex. Accessed: 10/06/2024.

[4] E. Lopienska. *The CERN accelerator complex, layout in 2022.* General Photo. 2022. URL: https://cds.cern.ch/record/2800984.

[5] The ATLAS Collaboration et al. "The ATLAS Experiment at the CERN Large Hadron Collider". In: *Journal of Instrumentation* 3.08 (Aug. 2008), S08003. DOI: 10.1088/1748-0221/3/08/S08003.

[6] S. Chatrchyan et al. "The CMS experiment at the CERN LHC. The Compact Muon Solenoid experiment". In: *JINST* 3 (2008). Also published by CERN Geneva in 2010, S08004. DOI: 10.1088/1748-0221/3/08/S08004.

[7] K. Aamodt et al. "The ALICE experiment at the CERN LHC". In: *Journal of Instrumentation* 3.8 (2008). ISSN: 1748-0221. DOI: 10.1088/1748-0221/3/08/S08002.

[8] A. A. Alves Jr. et al. "The LHCb Detector at the LHC". In: *JINST* 3 (2008), S08005. DOI: 10.1088/1748-0221/3/08/S08005.

[9] W. Herr and B. Muratori. "Concept of luminosity". In: *CERN Accelerator School and DESY Zeuthen: Accelerator Physics.* Sept. 2003, pp. 361–377.

[10] B. Schmidt. "The High-Luminosity upgrade of the LHC: Physics and Technology Challenges for the Accelerator and the Experiments". In: *J. Phys. Conf. Ser.* 706.2 (2016), p. 022002. DOI: 10.1088/1742-6596/706/2/022002.

[11] S. Morović. *CMS detector: Run 3 status and plans for Phase-2.* 2023. arXiv: 2309.02256 [hep-ex]. URL: https://arxiv.org/abs/2309.02256.

[12] O. Brüning and L. Rossi. *The High Luminosity Large Hadron Collider – HL-LHC.* 2024. DOI: 10.1142/9789811278952_0001.

[13] O. Brüning and L. Rossi. *The High Luminosity Large Hadron Collider*. WORLD SCIENTIFIC, 2015. DOI: `10.1142/9581`.

[14] CERN. *High-Luminosity LHC*. `https://home.cern/science/accelerators/high-luminosity-lhc`. Accessed: 29/06/2024.

[15] D. Barney. "An overview of the CMS experiment for CERN guides". 2003. URL: `https://cds.cern.ch/record/2629323/files/CMSdocumentforGuides.pdf`.

[16] G. L. Bayatian et al. "CMS Physics: Technical Design Report Volume 1: Detector Performance and Software". 2006.

[17] I. Neutelings. *CMS coordinate system*. Accessed: 17/06/2024. 2017.

[18] M. Mannelli. "CMS HL-LHC Upgrade: Selected Highlights". Presented at the LHCP 2020 on behalf of the CMS Collaboration. 2020.

[19] L. Portalès. "L1 Triggering on High-Granularity Information at the HL-LHC". In: *Instruments* 6.4 (2022). ISSN: 2410-390X. DOI: `10.3390/instruments6040071`.

[20] A.M. Magnan. "HGCAL: a High-Granularity Calorimeter for the endcaps of CMS at HL-LHC". In: *Journal of Instrumentation* 12.01 (Jan. 2017), p. C01042. DOI: `10.1088/1748-0221/12/01/C01042`.

[21] M. Dordevic. "The CMS Particle Flow Algorithm". In: *EPJ Web of Conferences* 191 (2018). Ed. by V.E. Volkova et al., p. 02016. ISSN: 2100-014X. DOI: `10.1051/epjconf/201819102016`.

[22] V. Khachatryan, A. M. Sirunyan, and A. Tumasyan. "The CMS trigger system." In: *JINST* 12.01 (2017), P01020. DOI: `10.1088/1748-0221/12/01/P01020`. arXiv: `1609.02366`.

[23] F. Pantaleo and M. Rovere. "The Iterative Clustering framework for the CMS HGCAL Reconstruction". In: *Journal of Physics: Conference Series* 2438.1 (Feb. 2023), p. 012096. ISSN: 1742-6588, 1742-6596. DOI: `10.1088/1742-6596/2438/1/012096`.

[24] M. Rovere et al. "CLUE: A Fast Parallel Clustering Algorithm for High Granularity Calorimeters in High-Energy Physics". In: *Frontiers in Big Data* 3 (2020). ISSN: 2624-909X. DOI: `10.3389/fdata.2020.591315`.

[25] E. Brondolin. "CLUE: a clustering algorithm for current and future experiments". In: *Journal of Physics: Conference Series* 2438.1 (Feb. 2023), p. 012074. DOI: `10.1088/1742-6596/2438/1/012074`.

[26] J. Jaroslavceva. "A New Trackster Linking Algorithm Based on Graph Neural Networks for the CMS Experiment at the Large Hadron Collider at CERN". Presented 14 Jul 2023. Prague, Tech. U. URL: `https://cds.cern.ch/record/2865866`.

[27] A. Nandi. “New Techniques for Reconstruction in the CMS High Granularity Calorimeter”. Presented 16 Jan 2023. RWTH Aachen U., 2022. URL: `https://cds.cern.ch/record/2854616`.

[28] R. J. Wilson. *Graph Theory.* Longman Group Ltd, 1996. ISBN: 0-582-24993-7.

[29] S. Fortunato. “Community detection in graphs”. In: *Physics Reports* 486.3 (Feb. 2010), pp. 75–174. ISSN: 0370-1573. DOI: `10.1016/j.physrep.2009.11.002`.

[30] D. Grinberg. *An introduction to graph theory.* Aug. 2023. DOI: `10.48550/arXiv.2308.04512`.

[31] Original by User:Nog33, converted to SVG by User:j_ham3.

[32] A. Clauset, M. E. J. Newman, and C. Moore. “Finding community structure in very large networks”. In: *Phys. Rev. E* 70 (6 Dec. 2004), p. 066111. DOI: `10.1103/PhysRevE.70.066111`.

[33] M. E. J. Newman and M. Girvan. “Finding and evaluating community structure in networks”. In: *Phys. Rev. E* 69 (2 Feb. 2004), p. 026113. DOI: `10.1103/PhysRevE.69.026113`.

[34] V. D. Blondel et al. “Fast unfolding of communities in large networks”. In: *Journal of Statistical Mechanics: Theory and Experiment* 2008.10 (Oct. 2008), P10008. ISSN: 1742-5468. DOI: `10.1088/1742-5468/2008/10/P10008`.

[35] V. A. Traag, L. Waltman, and N. J. van Eck. “From Louvain to Leiden: guaranteeing well-connected communities”. In: *Scientific Reports* 9.1 (Mar. 2019). Publisher: Nature Publishing Group, p. 5233. ISSN: 2045-2322. DOI: `10.1038/s41598-019-41695-z`.

[36] S. Abdouline et al. “The CMS Simulation Software”. In: Oct. 2006, pp. 1655–1659. ISBN: 978-1-4244-0561-9.

[37] C. Bierlich et al. *A comprehensive guide to the physics and usage of PYTHIA 8.3.* 2022. arXiv: `2203.11601`. URL: `https://arxiv.org/abs/2203.11601`.

[38] S. Agostinelli et al. “Geant4—a simulation toolkit”. In: *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment* 506.3 (2003), pp. 250–303. ISSN: 0168-9002. DOI: `https://doi.org/10.1016/S0168-9002(03)01368-8`.

[39] L. Garren et al. “Monte Carlo particle numbering scheme”. In: *European Physical Journal C* 15 (Mar. 2000), pp. 205–206. DOI: `10.1007/BF02683426`.

[40] L. Cristella. “Validation is automation”. Presented at the CMS HGCAl chat. 2022.