DE GRUYTER OPEN Acta Univ. Sapientiae, Informatica **9,** 2 (2017) 195–213

DOI: 10.1515/ausi-2017-0012

Hierarchical k_t jet clustering for parallel architectures

Richárd FORSTER
Eötvös University
email: forceuse@inf.elte.hu

Ágnes FÜLÖP
Eötvös University
email: fulop@caesar.elte.hu

Abstract. The reconstruction and analyze of measured data play important role in the research of high energy particle physics. This leads to new results in both experimental and theoretical physics. This requires algorithm improvements and high computer capacity. Clustering algorithm makes it possible to get to know the jet structure more accurately.

More granular parallelization of the k_t cluster algorithms was explored by combining it with the hierarchical clustering methods used in network evaluations. The k_t method allows to know the development of particles due to the collision of high-energy nucleus-nucleus.

The hierarchical clustering algorithms works on graphs, so the particle information used by the standard k_t algorithm was first transformed into an appropriate graph, representing the network of particles. Testing was done using data samples from the Alice offline library, which contains the required modules to simulate the ALICE detector that is a dedicated Pb-Pb detector. The proposed algorithm was compared to the FastJet toolkit's standard longitudinal invariant k_t implementation. Parallelizing the standard non-optimized version of this algorithm utilizing the available CPU architecture proved to be 1.6 times faster, than the standard implementation, while the proposed solution in this paper was able to achieve a 12 times faster computing performance, also being scalable enough to efficiently run on GPUs.

Computing Classification System 1998: I.1.4
Mathematics Subject Classification 2010: 58A20
Key words and phrases: jet, cluster algorithm, hierarchical clustering, database of experimenthal particle physics parallel computing, multi-core, C++11

1 Introduction

We researched the behaviour of the jet in the high energy physics [20, 25, 30] using the many-core architecture of the modern CPUs [18, 19]. We applied the most important principles of physics and we review the main concepts in this article.

The basic conception is the parton model to study the high energy hadron collisions. Due to the hadronisation process we can measure the final state object. The theory of strong interaction between the quark and gluon is described by Quantum Chromodynamics (QCD) [26]. In this process colored objects are created i.e. quarks and gluons. The scale of this procedure is few fermi 10^{-15} m. The short and long distance physics are fundamentally different. The colored objects are free to move within short range. On the scale of a few centimeter the colored objects become confined into color singlets. The process of quarks and gluon showering is a hadronization. During this procedure many mesons and baryons emerge they decay and form evolve the final state objects which are measured by detector. The spray of hadron is called jet which is a connection between the short scale physics and a final state measured particles [9, 28].

Different type of data requires different approaches to clusterize the input. For real world networks, hierarchical algorithms are used to compute the clusters. By combining some aspects of hierarchical processes and k_t jet clustering, a more efficient process will be made available for generating jets. This proposed algorithm has lower complexity compared to the k_t solution and has faster computation on the same hardware, while also being more scalable, that further enables jet generation on many-core architectures, such as GPUs. The Louvain method, used as the hierarchical clustering algorithm for the basis of the new process already proved to be scalable on both CPUs and GPUs. Combining the two methods provided a final algorithm, that can process 12 times faster, than the standard k_t clustering.

2 Clustering and declustering mechanism

In this Section we discuss the clustering of jets and it is followed by declustering in order to fine the substructures. There are many articles about this question in the literature [1, 16, 32]. We consider the most important algorithms [3] and discuss the physical meaning of these processes.

2.1 Jet

A jet is a narrow cone of hadrons and other particles which are formed by quark and gluon during the high-energy collisions. From the hard subprocess hadronize the created hadrons become collimated around original parton directions and the higher energy parton takes shapes more collimated. These bunches of hadrons are called jets (Figure 1). They can be interpreted as a link to partons and it can yield to understand a deeper level of the important parton interactions. Jets sketch forms a rather simple process, what happened in an event without taking into calculation the multiparticle dynamics. Therefore we use the jets, rather the directly measured hadrons, these can be constructed as infrared-safe observables. The QCD is theoretical background to calculate the predictions of jets with high precision.

2.2 Clustering of jet

The jet is clustered, when we study the jet momentum due to the final state particles in the calorimeter [5]. The results can be made more accurate to consider the other measured quantities as muon systems. All together it means the clusters. By theoretical research we have to mention two questions. These are the infrared (IR) safety and collinear safety[28].

An observable is infrared safe, if it does not depend on the low energy physics of the theory. We speak about the collinear(C) safety, when a parton is replaced by a collinear pair of partons, then it should not modulate the jet clustering results. The properties of jet does not change, when one of the particles radiates a very soft objects, or breaks up two collinear particles. Therefore the jet can be determined by perturbative methods to compare with the experiment.

In the theoretical physics the infrared divergence means that situation, when an integral of Feynman diagram diverges because of the constituent objects with very small energy goes to zero. It is important, when the model contains massless particles, as photons. One possibility to deal with it is to apply an infrared cutoff and it approaches zero. The divergence is usually remains finite in all measurable quantities. Therefore the infrared safe and collinear safe jet reconstruction algorithm can be used to the evaluation of the measured data responsing to theoretical condition or it is applied to a given order thanks to the algorithm becomes IRC safe.

The determination of the jets mass and energy depend on the size of jet radius. In the case of larger jet radius these quantities can be calculated more

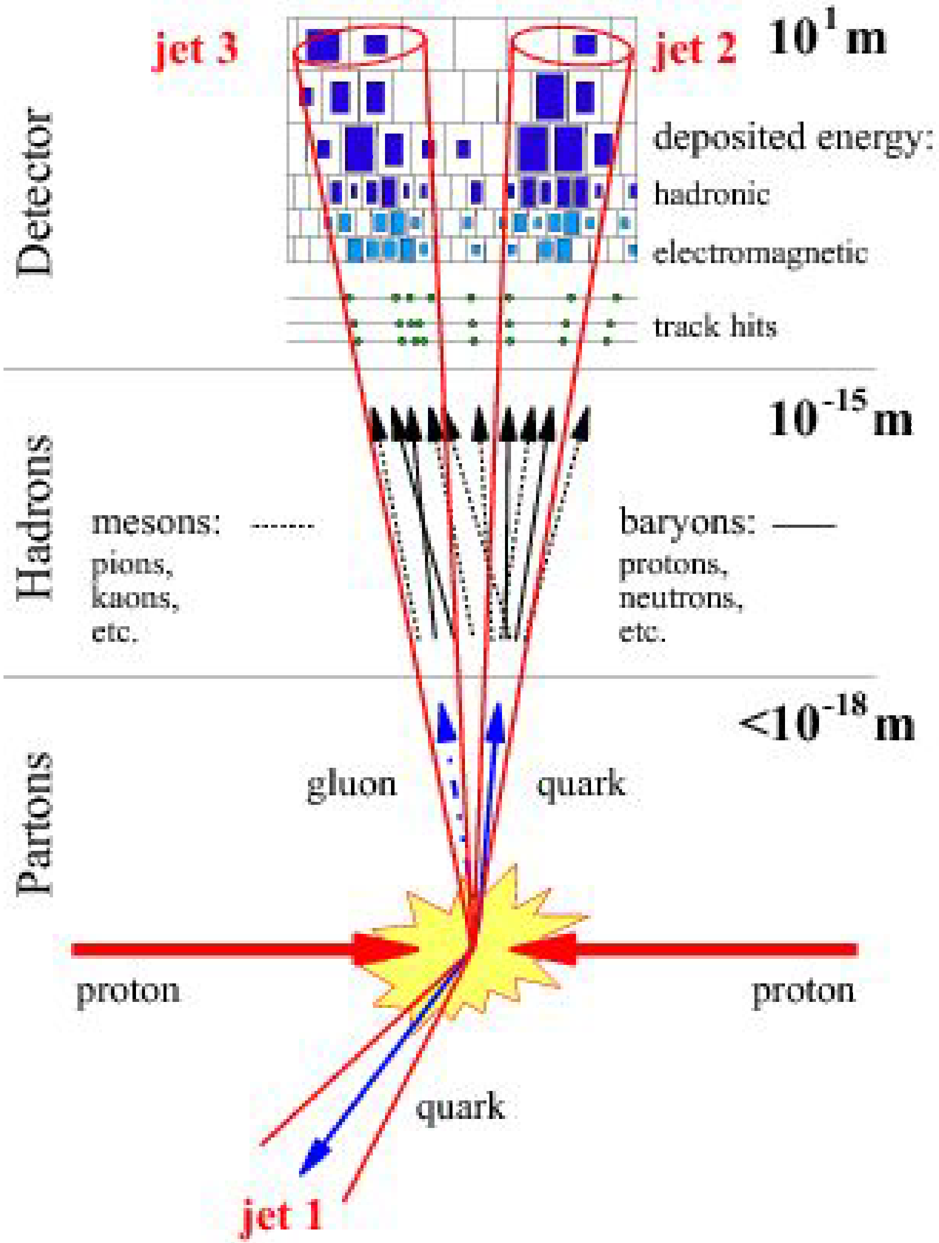


Figure 1: Structure of jet

exactly than smaller distance because the cluster contains more hadronised particles, which includes the underlying event and pile-up.

2.3 Cone and sequential recombination algorithms

We discuss two important classes of jet algorithms. The first is the cone algorithm. This methods are the iterative cone with progressive-removal (IC-PR) [2], The iterative cone with split-merge (IC-SM) [7] and seedless infra-red safe cone (SIScone) [31].

The second part of the algorithm is the sequential clustering algorithms. We will introduce the K_t [14], Anti-K_t [12] and the Cambridge/Aachen [34] algorithms.

2.3.1 Cone algorithms

In the case of cone algorithm a conical region contains the particles of jet, so the cluster takes place in the $(\eta - \phi)$ space. Therefore the jet has rigid boundaries. This algorithm was popular in the experimental physics, because it could be easier implemented, but it was not so preferred in the theoretical physics. The cone algorithms are IRC unsafe.

IC-PR The iterative cone algorithm together progressive removal is a collinear unsafe algorithm.

Look for that cell which has the largest p_t i.e. the hardest box. It becomes a center. Let us generated the radius of cone R around the center. We can determine the trial jet axis to sum up the cells inside the cone by four-vectors. If the trial jet axis corresponds to center axis, the behaviour of cone is stable. Each particles which are situated in the stable cone are deleted from the list of particles. This method is repeated by the next hardest cell. But if the trial jet axis does not correspond to the center of axis, thenn the trial jet axis need to look for the another center axis. This method is repeated until convergence of the axes occurs. This process is repeated until there are center above threshold energy E_{cut}.

IC-SM The iterative cone algorithm together with the split merge method is an infra-red unsafe algorithm. For example JetClu, midpoint cone. This process of the jet is as follows:

The first all cells which are above a threshold energy E_{cut} are center. Look for all stable cones together with those center applying the same method as in the IC-PR algorithm, but we do not delete any particles from the list once, a stable cone is discovered. Each stable cones which were recovered are written as ptortojets. A split merge method is run on the protojets until we do not discover each of them.

SIScone That infra-red safe cone method which does not have center is a IRC safe cone algorithm. Because the area is relatively small, therefore it works by the UE and PU well. It has better results than the R for hard radiation, therefore has good resolution, but it is wrong for resolving multijets. The process of SIScone of the jet is written schematically:

1. Choose a particle i
2. Look for each particles j within distance $2R$ of i
3. If there is no more particle j
4. then i is a stable cone and write to the list of protojets
5. Else
6. Look after the circles which is generated by i and j. These are lying on their circumference and determine the momenta of the cones.
7. For each circle
8. All four permutations of the two edge points which are situated on or out of the circle. These four circles mean as current cones.
9. Each current cone, which was not previously looked for.
10. It must be decided that the current cones are situated in or out of the edge particles. This is same as the cone determined by the momentum of the particles in the current cone. If not, then the current cone is unstable.
11. Check each current cones which are not unstable and create an explicit stability one. Write each stable cones to the list of protojets.
12. Run a split merge method on the protojets.

2.3.2 Sequential clustering algorithms

The sequential clustering algorithms [3] can be used when the particles are situated in the jets and there are small differences in the transverse momenta. Therefore so called groups particles can be written on the momentum space in jets, which contain fluctuating areas in $(\eta - \phi)$ space. The sequential clustering algorithms were preferred by theorists. This method do not have been favoured by experimentalists, because it has slow implementation. The FastJet program [13] provides such clustering algorithms which are speed enough for experimental research. Sequential clustering algorithms are also IRC safe.

We introduce the basic idea of sequential clustering algorithms. The first we mention the distance variable between two particles $d_{ij} = min(p^{a}_{ti}, p^{a}_{tj}) \times \frac{R^2_{ij}}{R}$, where a is an exponent corresponding to a particular clustering algorithm. We define distance $R^2_{ij} = (\eta_i - \eta_j)^2 + (\phi_i - \phi_j)^2$ between two particles in the $(\eta - \phi)$

space and R is the radius parameter which determines the final size of the jet. The value is changing in the range [0.4,0.7]. The second distance variable is $d_{iB} = p_{ti}^{a}$ which is the momentum space distance between the beam axis and the measured particle.

In the sequential clustering algorithms the first step is to look for the minimum of the entire set $\{d_{ij}, d_{iB}\}$. If d_{ij} is the minimum distance then the particle i and j are integrated in one particle (ij) we need to take the summation four-vectors and i and j are deleted from the list of particles.

If d_{iB} is the minimum value, then i becomes a final jet and it is deleted from the list of particles.

We need to continue this method until either each particles are part of jet, where the distance between the jet axes R_{ij} larger than R, this process is the inclusive clustering. Or until a designed the total of jets have been looked for. We call it exclusive clustering.

K_t method In the case of K_t algorithm [12, 14] the a value equals to 2, then the equations follows this form:

$$d_{ij} = \min\left(p_{ti}^2, p_{tj}^2\right) \times \frac{R_{ij}^2}{R} \tag{1}$$

$$d_{iB} = p_{ti}^2 \tag{2}$$

The K_t algorithm [15] is applied mainly to cluster the soft particles, because the particles have low p_t particularly effecting the fluctuates in area appreciably and a method that is susceptible to the UE and PU [23]. Because the method of clustering is efficient process, therefore K_t algorithm works well at resolving subjets.

Anti-K_t method The value a equals to -2, then we speak about Anti-K_t. The form of equation is the following:

$$d_{ij} = \min\left(\frac{1}{p_{ij}^2}, \frac{1}{p_{ij}^2}\right) \times \frac{R_{ij}^2}{R} \tag{3}$$

$$d_{iB} = \frac{1}{p_{ti}^2} \tag{4}$$

The equation (3) is dominated by high p_t, therefore this method can be applied to cluster hard particles mainly. So the area fluctuates slightly and the process is slightly susceptible to the UE and PU. The Anti-K_t's clustering preference results in a method, namely it has the optimum at resolving jets, but therefore it has poor de-clustering.

Cambridge/Aachen The value a equals to 0 which yields the C/A algorithm, which provides the following equations:

$$d_{ij} = \frac{R_{ij}^2}{R}$$

$$d_{iB} = 1$$

Both of the distance variables do not depend on the momentum, therefore its area fluctuates poorly and slightly susceptible to the UE and PU. Because the spatial property of the distance variable is little, therefore C/A de-clusters is optimum for studying jet substructure, but it is poorly more complicated to de-cluster than the K_t algorithm.

2.3.3 Sequential algorithms: the FastJet package

FastJet [13] is a software package which is used to determine the cluster jets. It is an open source program.

The basic reconstruction algorithm [11] has been further developed to faster software including the array structure for the distance between the objects.

The original program implementation provides the next calculation of the demand:

First we determine the d_{ij} distance between all the particles and the d_{iB} between all the objects. This calculation needs $O(N^2)$.

Second we search the closest particles and that objects which are nearest neighbour, i.e. the minimal value of the distance d_{ij} and d_{iB}. This calculation disposes $O(N^2)$ and it is done N times.

Then we can perform the jet reconstruction to apply the advanced pairs and cluster of measured data set. The calculation of the algorithm in particular the cluster provides $O(N^3)$.

FastJet algorithm employs two arrays to solve the original processes. One of them is applied for the distance between the closest objects, another contains the distance of the beamline. The calculation demand is $O(N^2)$.

Further development of the FastJet software achieves $O(N \ln N)$ calculation to use another applicable metric instead of the array structure for the distance of closest particles.

That clusters and jets, which were produced by FastJet, are saved in pseudojets. It plays important role, because it allows to reconstruct the structure of the clusters. The pseudojets include the four-momentum and the hierarchical time dependent events inside the cluster, this results all objects in the jet and it plays important role for the declustering mechanism.

2.4 Substructure of jet

Substructure of jets can be one jet containing more than one group of gaussian-distributed clusters. Substructure can also be a non gaussian component, which is corresponds to an offset. It can as well consist of another gaussian group of clusters, namely second hard jet.

Three different types which is able to define:

I: Subjet from uncorrelated sources, overlapping the hard jet considered or clustered together with it. It is soft process, originating from proton-leftovers, initial state radiation, beam-rests and/or scatterings, e.g. pileup (PU) and underlying event (UE).

II: Subjet from correlated sources, clustered together with the hard jet considered, originating from the same primary vertex, but another branch of the Feynman diagram.

III: Subjet from correlated sources, originating from the decay of a single boosted particle, clustered together into a single jet.

3 Problem statement and notation

Let $G(V, E, \omega)$ be an undirected weighted graph, with V representing the set of vertices, E the set of edges and ω a weighting function that assigns a positive weight to every edge from E. If the input graph is unweighted, then the weight of the edges is considered to be 0. The graph is allowed to have loops, so edges like (i, i) are valid, while multiple edges between the same nodes should not be present. The following will be the adjacency list of vertex i: $\Gamma(i) = j|(i, j) \in E$. Let δ_i denote the weighted degree of vertex i, such as $\delta_i = \sum_{j\in\Gamma(i)}\{\omega(i, j)\}$. Let N denote the number of vertices in graph G, M the number of edges, and W the sum of all edge weights, such as $M = \frac{1}{2}\sum_{i\in V}\delta_i$. By computing the communities, the vertex set V will be partitioned into an arbitrary number of disjoint subsets, each with size n, where $0 < n \leq N$. $C(i)$ will denote the community containing vertex i. $E_{i\to C}$ is the set of all edges connecting vertex i into community C. Consequently let $e_{i\to C}$ hold the sum of the edge weights in $E_{i\to C}$.

$$e_{i\to C} = \sum_{(i,j)\in E_{i\to C}} \omega(i,j) \tag{5}$$

The sum of all the vertices in community C shall be denoted by deg_C, which will represent the degree of the whole community.

$$\mathsf{deg}_C = \sum_{i\in C} \delta_i \tag{6}$$

3.1 Modularity

Let $S = C_1, C_2, ..., C_k$ be the set of every community in a given partitioning of V, where $\{1 \leq k \leq N\}$. Modularity Q of partitioning S is given by the following [27]:

$$Q = \frac{1}{2W} \sum_{i\in V} e_{i\to C(i)} - \sum_{C\in S} \left(\frac{\mathsf{deg}_C}{2W} \cdot \frac{\mathsf{deg}_C}{2W} \right) \tag{7}$$

Modularity calculation is a common solution to measure the quality of the process and also to define a termination function. Still it's not without some drawbacks, like the resolution limit [21, 33]. Definition can be given in multiple forms as are described in [33, 4, 6]. In the literature the more widespread version is defined in Eq. 7, also this is used in the Louvain method [8].

3.2 Community detection

Given a $G(V, E, \omega)$ graph as input, the expected result is partitioning S of communities that leads to the greatest modularity. This problem is known to be NP-complete [10]. The major difference compared to other partitioning solutions is the number and size of the clusters, while also in specific cases some initial distribution is given [22].

4 The Louvain algorithm

The Louvain method [8], gives an iterative, greedy algorithm, that produces the communities in multiple phases. Each phase runs for many iterations until the system is converging. As the initialization of the first phase each vertex will belong to a set containing only that node. As the process goes on, through the iterations the gain in modularity becomes lesser and the iterating stops when a predefined threshold is reached. In every round the vertices are checked in an arbitrary, but predefined order. Visiting vertex i, the neighbors are explored, searching for a new community with the highest modularity gain. Once this calculation is done, the selected neighbors community will be selected and vertex i will be moved there. If this search yields no suitable community, no changes are made to the current node. One iteration lasts until all vertices are examined. Because of this the modularity is a monotonically increasing. By reaching convergence in a given phase, the system is getting reduced, by assigning a single "meta-vertex"[24] in the place of all the nodes belonging to the same community. The new nodes can have loops and the weight of these new edges will be the sum of the weights of all the edges that are connecting the inner nodes of the group. For an edge pointing into another cluster, the weight is calculated by summing the weights of all the edges between the connected sets. The result will be a reduced graph $G'(V', E', \omega')$, which becomes the input for the next phase. At any given iteration, $\Delta Q_{i\to C(j)}$ holds the modularity gain resulting from the reassignment of vertex i from its current community $C(i)$ to a neighboring $C(j)$. This is given by:

$$\Delta Q_{i\to C(j)} = \frac{e_{i\to C(j)}}{W} + \frac{2 \cdot \delta_i \cdot \mathrm{mod}_{C(i)/i} - 2 \cdot \delta_i \cdot \mathrm{mod}_{C(j)}}{(2W)^2} \tag{8}$$

For any vertex i the new community will be computed based on the following. For $j \in \Gamma(i) \cup \{i\}$:

$$C(i) = arg \max_{C}(j) \Delta Q_{i\to C(j)} \tag{9}$$

Because the modularity is monotonically increasing it guarantees termination. By running only for a dozen iterations during a few phases, this method can find the clusters in real world datasets.

4.1 Parallel heuristics

The challenges to parallelize the Louvain method were explored in [24]. To solve those issues multiple heuristics were introduced, that can be used to leverage the performance of the parallel systems in a basically sequential algorithm. From the proposed heuristics two is going to be detailed in this paper. Lets assume the communities at any given stage are labeled numerically. The notation $l(C)$ will return the label of community C.

4.1.1 Singlet minimum label heuristic

In the parallel algorithm, if at any given iteration vertex i which is in a community by itself ($C(i) = i$, singlet community [24]), in hope for modularity gain might decide to move into another community, that holds only vertex j. This transition will only be applied if $l(C(j) < l(C(i)))$.

4.1.2 Generalized minimum label heuristic

In the parallel algorithm, if at any given iteration the vertex i has multiple neighboring communities providing modularity gains, the community with the minimum label will be selected. Swap situations might occur, when two vertices are transitioning into the other's community in the same iteration. This can delay the convergence, but can never lead to nontermination as the minimum required modularity gain threshold will guarantee a successful termination.

5 Hierarchical jet clustering

The hierarchical clustering is based on the Louvain clustering (Section 4). To work with it, the algorithm was tweaked to incorporate the specific needs of the jet clustering.

The Louvain method works on a weighted, undirected graph, while on the other hand jet clustering (Subsection 2.2) uses a list of input particles, hence this list needs to be transformed into a graph. Obviously the particles themselves will be appropriate for the nodes. As the k_t clustering uses the distance between the elements, a function assigning an edge to the nodes with the distance between the two adjacent items is a logical solution. The problem in this case is the sheer volume of edges, as this will create a fully connected graph with $n * (n-1)/2$ total links. Instead making the connection between nearest neighbours and second to nearest proves to be sufficient. Also, the edges will now contain directionality. In those cases, when the particle's nearest "neighbour" is the beam, the node in graph will be isolated, and will represent a singular jet. The original hierarchical process is greedy in the sense, that it relies on modularity gain to drive the computation. This is important, because there is no information about the clusters before starting the computation. While using the k_t algorithm it is known, that the processing will end, when all particles are assigned into a jet, thus eliminating the need to compute the modularity for the graph.

The result of a hierarchical clustering is the dendogram. This tree will contain the connection between the different phases of the cluster generation, leading to the final assignments. Originally this is implemented by generating a level of this tree, a new graph is induced by renumbering the nodes based on their actual cluster assignment and recalculate the edges for them. Jet clustering doesn't need the inner weight of the clusters, only the calculated distance from that subjet and that will be incorporated into the graph itself through the edges. The solution proposed here doesn't require the regeneration of each levels graph, but the graph will be dynamically morphed, by applying the changes through the different phases. Also the terminology used for the generation in the original Louvain algorithm is to move the nodes into clusters, where each individual node will check which cluster will be the best in the actual phase. Here the nodes connected with a directed edge will decide among each other in such a way, that the node pointing to another one will draw that to itself.

5.1 Sequential processing

Initially the result of the Louvain clustering depends on the original order of the input value. The same can be said about the $\mathrm{k_t}$ clustering (Subsection 2.3.2) as well: always the two closest elements are combined into a new jet, thus the input should be ascending ordered. This way always the first element of this list will be tested. After generating the new item, its distance should be calculated against the remaining elements and finally it needs to be inserted into the input array using a sorted insert based on the calculated distance and a link will be generated between this item and its nearest neighbour, while the links to the original subjets will be removed. The algorithm checks after this recombination if the other elements nearest neighbour is the new recombined jet or a completely different item. If the distance between an element and the recombined jet becomes bigger, than how much for its individual part was, then have to check if there are additional nearer neighbours and the closest of them will be linked too and the two edges will be set this way. The processing continues until any of the original input particles are present.

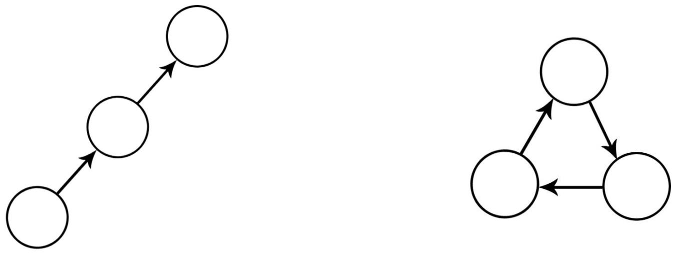
5.2 Parallel processing

The evaluation of the original hierarchical clustering in parallel required some additional heuristics (Subsection 4.1) to keep the precision and consistency of the base algorithm. By introducing the method to a more complex clustering process, further requirements have to be satisfied to compute the final inclusive jets. These additional heuristics are closely connected to the structure of the generated graph.

5.2.1 Transitivity effect

Let's call the transitivity effect the case, when multiple particles or subjets are having at least 1 input and 1 output edge. The different problems, that can rise from this are explored in this subsection.

Processing in parallel in one phase multiple nodes will try to merge themselves with the connected nearest neighbour, while also that neighbour might do the same at the same time, building up a chain. Just following through that chain and merging the nodes together, might omit a potential change in the nearest neighbours, that might appear by computing the separate recombined subjets. In a simple case (Figure 2a) this can be solved, by only applying the draw from the node, that is not tried to be recombined by another node, meaning the node doesn't have incoming edges.



(a) Chained nodes in the graph

(b) Cycle in the graph

Figure 2

A more complex case might involve cycles (Figure 2b) on this chain, where all the nodes have inbound links and the previous solution can't be applied. While processing the graph, if no cycles are present, then new subjets will be generated. Even if there are cycles, but still have simple chains, the computation can continue. At one point the clustering will not be able to push any node into a new cluster as only cycles are available, effectively halting the processing. If the clusterization is incomplete, there are nodes, that are not assigned to jets and no cluster assignment is taking place in a given phase, it is known to have cycles only, so there is no need for additional cycle checking.

The next step is to eliminate the cycle. For this the two nodes with the shortest distance needs to be found, thus a minimum search is required. This way the two elements are getting removed from the chain, with all of its connections. In case of the new subjet invocating a new cycle, this search will be repeated. If the graph has multiple unconnected components (Figure 3), the search can be done in parallel among all the components, but not for connected components, as they might connect to the same cycle.

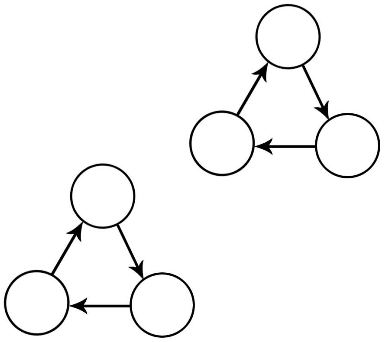


Figure 3: Multiple components in the graph

5.3 Results

The complexity of the $\mathsf{k_t}$ jet algorithm is $\Theta(\mathsf{N}^2)$, which requires a high amount of computation to be done. It was shown in [18], that by applying parallelization, the runtime of this process can be reduced considerably. Tests running on the system detailed in Table 1, using the raw data from an event (containing 140535 points) simulated with the AliRoot framework's PbPbbench [29] test application.

The system used for development and testing is described in Table 1.

CPU	GPU	OS	Compiler
Intel Core i7 4710HQ	GeForce GTX 980M	Windows 10 Pro	Visual C++ 2013

Table 1: The test system

The previously proposed parallel implementation takes $207,9$ seconds compared to the FastJet [13] (Subsection 2.3.3) framework's sequential $\mathsf{k_t}$ implementation, that needs $347,18$ seconds to finish the clustering, giving a 1.67 faster runtime. On the other hand the new hierarchical jet clustering (Section

5) method needs only 81 seconds to conclude the generation, while producing the same output and the parallel implementation takes only 26 seconds to finish the clustering. This leads to a 13 times faster evaluation compared to the sequential k_t algorithm and 8 times faster compared to the parallel implementation of the same jet clustering (Table 2).

Algorithm	Complexity	Runtime
k_t	$\Theta(N^2)$	$347,18$ s
parallel k_t	$\Theta(N^2)$	$207,9$ s
hierarchical jet clustering	$O(N)$	81 s
parallel hierarchical jet clustering	$O(N)$	26 s

Table 2: Runtimes of the k_t and parallel k_t algorithm and the hierarchical jet clustering

5.3.1 Complexity

In every step, where a new subjet is generated, it's distance will be computed to the other remaining $n–2$ nodes, where n is the number of nodes in the actual phase. If the computation is running sequentially (Subsection 5.1), the number of nodes decreases by 1 between each phase, while in parallel (Subsection 5.2) it depends on how many subjets will be computed at once. Overall this part will take $(n-2)+(n-4)+\cdots+1$, because the process will continue until all original particles are assigned to a jet. In the worst case it might be, that always subjets will be merged and at the end there will be 1 subjet and 1 particle.

If chains will be present, to break them up can be done in constant time as the node that isn't pulled by someone will merge it's nearest neighbour. In the case of cycles, the complexity comes from finding the unconnected components and doing the minimum search in each of them. To find the minimum edge, the complexity will be $O(N)$. For the component search, if the graph is stored with the list of edges, to find all components it will take $O(N+M)$ steps, where *N* is the number of nodes and *M* is the number of edges.

Overall the complexity of the proposed algorithm is $O(N)+O(N)+O(N+M)=O(N)$.

6 Summary

In this paper a new kind of jet clustering algorithm is detailed, that builds on some fundamental characteristics of hierarchical clustering used on real network datasets. Thanks to this approach the $\Theta(N^2)$ complexity of the original k_t jet algorithm was reduced to be linear (Subsection 5.3.1). This and the higher granularity coming from this allows for further parallelization, that greatly helps reducing the time of processing, providing a 13 times faster computation. Also thanks to this GPU based computing for jet clustering becomes implementable and as it was introduced in [17] for the Louvain clustering method, the use of many-core architectures further decreases the runtime even with a factor of 12.

7 Future work

The proposed approach needs more thorough testing, with different data sets to see if the precision always stays the same and produces the same result as the original k_t algorithm. On the other hand the modifications to the hierarchical clustering should be also applied on the full GPU implementation to see the performance benefits of the new solution with extremely parallelizable architectures.

A bigger step will be to see how this solution can be further improved upon, potentially using machine learning in the process and providing a fundamentally different approach to clustering.

References

[1] A. Ali, G. Kramer, Jets and QCD: A historical review of the discovery of the quark and gluon jets and its impact on QCD *Eur. Phys. J. H* **36** (2011) 245–326. [arXiv:1012.2288 [hep-ph]]. ⇒196

[2] G. Arnison et al. [UA1 Collaboration], Hadronic jet production at the CERN proton-antiproton collider, *Phys. Lett. B* **132** (1983) 214. ⇒199

[3] R. Atkin, Review of j-et reconstruction algorithms, *Journ. of Phys.: Conf. Ser.* **645** (2015) 012008. ⇒196, 200

[4] D. Bader, J. McCloskey, Modularity and graph algorithms, *SIAM AN10 Minisymposium on Analyzing Massive Real-World Graphs* (2009) 12–16. ⇒204

[5] F. Beaudette [CMS Collaboration], Performance of the particle flow algorithm in CMS, *PoS ICHEP 2010* (2010) 002. ⇒197

[6] J.W. Berry, B. Hendrickson, R.A. LaViolette, C. A. Phillips, Tolerating the community detection resolution limit with edge weighting, *Phys. Rev. E* **83,** 5 (2011) 056119. ⇒204

[7] G. C. Blazey, J. R. Dittmann, S. D. Ellis, V. D. Elvira, K. Frame, S. Grinstein, R. Hirosky and R. Piegaia et al., *Run II Jet Physics: Proc. of the Run II QCD and Weak Boson Physics Workshop*, [arXiv:hep-ex/0005012] ⇒199

[8] V. D. Blondel, J.-L. Guillaume, R. Lambiotte, E. Lefebvre, Fast unfolding of communities in large networks, *Journal of Statistical Mechanics: Theory and Experiment* **10** (2008) doi:P10008 ⇒204

[9] M.G. Bowler, *Femptophysics*, Pergamon Press 1990. ⇒196

[10] U. Brandes, D. Delling, M. Gaertler, R. Gorke, M. Hoefer, Z. Nikoloski, D. Wagner, On modularity clustering, *IEEE Trans. Knowl. Data Eng.* **20,** 2 (2008) 172–188. ⇒204

[11] M. Cacciari, G. P. Salam, *Phys. Rev. Lett* **B641** (2006) 57–61 [hep-ph/0512210] ⇒202

[12] M. Cacciari, G. P. Salam, G. Soyez. The anti-K_t jet clustering algorithm, *JHEP* **0804** (2008) 063 [arXiv:0802.1189 [hep-ph]]. ⇒199, 201

[13] M. Cacciari, G. P. Salam, G. Soyez, FastJet user manual, *Eur. Phys. J. C.* **72** (2012) 1896, arXiv:1111.6097v1 ⇒200, 202, 209

[14] S. Catani, Y.L. Dokshitzer, M. H. Seymour, B. R. Webber, Longitudinally invariant K_t clustering algorithms for hadron hadron collisions. *Nucl. Phys. B* **406** (1993) 187224. ⇒199, 201

[15] S. D. Ellis, D. E. Soper, Successive combination jet algorithm for hadron collisions *Phys. Rev. D* **48,** 7 (1993) 3160. ⇒201

[16] S. D. Ellis, J. Huston, K. Hatakeyama, P. Loch, M. Tonnesmann, Jets in hadron-hadron collisions *Prog. Part. Nucl. Phys.* **60** (2008) 484 [arXiv:0712.2447 [hep-ph]]. ⇒196

[17] R. Forster, Louvain community detection with parallel heuristics on GPUs, *20th Jubilee IEEE Int. Conf. on Intelligent Engineering Systems* **20** (2016), ISBN:978-1-5090-1216-9, doi: 10.1109/INES.2016.7555126 ⇒211

[18] R. Forster, A. Fülöp, Parallel k_t jet clustering algorithm, *Acta Univ. Sapientiae Informatica* **9,** 1 (2017) 49–64. ⇒196, 209

[19] R. Forster, A. Fülöp, Jet browser model accelerated by GPUs, *Acta Univ. Sapientiae Informatica* **8,** 2 (2016) 171–185. ⇒196

[20] R. Forster, A. Fülöp, Yang-Mills lattice on CUDA, *Acta Univ. Sapientiae, Inf.*, **5,** 2 (2013) 184–211. ⇒196

[21] S. Fortunato, Community detection in graphs, *Phys. Rep.* **486,** 35 (2010) 75–174, http://dx.doi.org/10.1016/j.physrep.2009.11.002. ⇒204

[22] B. Hendrickson, T. G. Kolda, Graph partitioning models for parallel computing, *Parallel Comput.* **26,** 12 (2000) 1519–1534. ⇒204

[23] M. Hodgkinson, Missing ET performance in ATLAS, in *Proc. 34th International Conference in High Energy Physics (ICHEP08)*, Philadelphia, 2008, eConf C080730 [arXiv:hep-ex/0810.0181] ⇒201

[24] H. Lu, M. Halappanavar, A. Kalyanaraman, Parallel heuristics for scalable community detection, *Parallel Computing* 47 (2015) 1937 ⇒205, 206

[25] S. Moretti, L. Lonnblad, T. Sjostrand, New and old jet clustering algorithms for electron-positron events JHEP **9808** (1998) 001 [arXiv:hep-ph/9804296]. ⇒196

[26] T. Muta, *Foundation of Quantum Chrodinamics*, World Scientific Press 1986. ⇒196

[27] M.E.J. Newman, M. Girvan, Finding and evaluating community structure in networks, *Phys. Rev. E* **69,** 2 (2004) 026113. ⇒204

[28] M. E. Peskin, D. V. Schroeder, *Quantum Field Theory*, Westview Press, 1995. ⇒196, 197

[29] D. Rohr, S. Gorbunov, A. Szostak, M. Kretz, T. Kollegger, T. Breitner, T. Alt, ALICE HLT TPC Tracking of Pb-Pb events on GPUs, *Journal of Physics: Conference Series* 396 (2012) doi:10.1088/1742-6596/396/1/012044 ⇒209

[30] G. P. Salam, *Towards Jetography*, *Eur. Phys. J. C***67** (2010) 637–686. [arXiv:0906.1833 [hep-ph]]. ⇒196

[31] G. P. Salam, G. Soyez, A partical seedless infrared-safe cone jet algorithm, *JHEP* **0705** (2007) 086 [arXiv:0704.0292 [hep-ph]]. ⇒199

[32] G. Sterman, S. Weinberg, Jets from quantum chromodynamics, *Phys. Rev. Lett.* **39** (1977) 1436. ⇒196

[33] V. A. Traag, P. Van Dooren, Y. Nesterov, Narrow scope for resolution-limit-free community detection, *Phys. Rev. E* **84,** 1 (2011) 016114. ⇒204

[34] CMS collaboration, A Cambridge-Aachen (C-A) based jet algorithm for boosted top-jet tagging. *CMS PAS JME-09-001*, 2009 ⇒199

Received: November 6, 2017 • Revised: December 6, 2017