Comparison of Jet reconstruction Algorithms

Miner Jacob

***Abstract*— Jet algorithms**

***Index Terms*— High Energy Physics, Jets, k_T, Cone, Jet Algorithm**

I. Introduction

IN any process producing colored objects, namely quarks and gluons, the short and long distance physics are distinctly different. On the scale of a few fermi, 10^{-15} m, the colored objects are free to move about, however; on the scale of a few centimeters the colored objects must be confined into color singlets. This process of quarks and gluons showering, hadronization, forms many mesons and baryons which can then decay and form the final state objects measured in our detectors. The spray of hadrons, known as jets, provide our link between the short scale physics and the final state observations.

The final state objects we measure are never nicely formed into well separated jets. Effects like detector smearing and inefficiency add to our uncertainty in grouping particles into jets. The fact that there is no one-to-one correspondence between the short distance physics and the final hadronized state complicates the definition of jets. In effect there is no single definition of a jet that is correct; our most strict requirement is that we must use a definition that is consistent with both theory and experiment.

II. Algorithm Types

Experimental data begins with a large number of particles found by the detector and a list of their four-momenta. The particles' energy, p_T and position are obtained from this information. We use this to define the coordinate system $\phi =$ azimuthal angle and rapidity $y = frac12\log(\frac{E+p_L}{E-p_L})$ or more commonly pseudorapidity $\eta = -\ln(\tan(\frac{\theta}{2}))$. Out of this state we must use algorithms, based on a specific jet definition, to reconstruct the jets. The two main jet definitions in use are the Cone Jet and k_T Jet based algorithms. The cone jet algorithms assume that jets will show up in roughly circular regions in the angular plane of the calorimeters and seeks to find stable regions of energy. The k_T jet algorithms attempt to locate clusters of particles which are close in momentum space and thus are akin to a final state shower which will be roughly collinear. Though both definitions work well with theory, their implementations and quirks in practice leave room for optimization.

A. Cone Algorithm

The cone-based algorithms begin by defining the calorimeter towers above a pre-defined value (typically 1 GeV) as

This work was funded by NSF and sponsored by UW

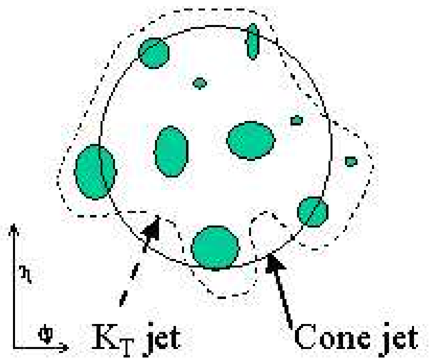

Fig. 1. Difference in jet Structure for Cone and k_T algorithms. The circle represents a cone algoithm's stable jet, the dashed line represents a stable k_T jet and the small ovals represent energy deposited in the calorimeter. Notice that the Cone algorithm uses a rigid boundary whle the k_T algorthm produces more amorphous jets.

seeds, or places to initially put cones. For consistency with perterbation theory we need to look everywhere, though that can be computationally inefficient and it is rarely done in practice. The algorithm then analyzes each cone by calculating the E-scheme centroid,

$$k \subset C \text{ iff } \sqrt{(y_k - y_C)^2 + (\phi_k - \phi_C)^2} \leq R_{cone},$$
$$p_C = (E_C, \overrightarrow{p}_C) = \sum_{k \subset C} (E_k, \overrightarrow{p_k})$$
$$\overline{y}_C \equiv \frac{1}{2}\ln\frac{E_C + p_{z,C}}{E_C - p_{z,C}},\ \overline{\phi}_C \equiv \tan^{-1}\frac{p_{y,C}}{p_{x,C}}.$$

of all the energy within specified cone radius, where typically R = 0.7. If $\eta_{cone} = \eta_{centroid}$ and $\phi_{cone} = \phi_{centroid}$ then it is defined to be a stable jet. If the cone's centroid is not its center, then the cone is moved to the centroid and iterated. This process is repeated until we have found all final state jets.

B. Cone Algorithm Problems

Though the initial definition is consistent with theory, we find that the introduction of seeds leads to an infrared sensitivity in perturbation theory. Also, the Cone jet Algorithm has a problematic splash-out effect, as can be seen in 1. We see that nearby regions of energy that "should" be in the jet can be excluded from the stable jet found. Finally, there is the issue that of where to put the energy if two cones overlap. Though all jet algorithms have their own quirks, resolutions to these issues were needed for more accurate results.

C. Cone Algorithm Fixes

The resolutions to these issues are relatively subtle changes in the cone jet algorithms. Overlap issue can be resolved

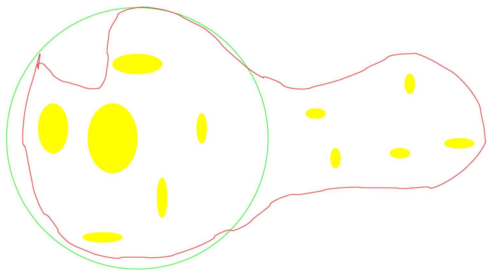

Fig. 2. Splash-in or vacuum effect of the k_T jet algorithms.

by an additional parameter, f_{merge}, which determines what percentage of the overlap energy must be in one jet before we put all of the energy into it. Otherwise we will put the individual pieces of energy into their respective closest jet. To "solve" the infrared sensitivity we can add a step after finding stable cones, we search for a stable cone at the midpoint of two cones that have separation $R<d<2R$. Finally, a more robust solution to the problems cause by seeds is to place seeds everywhere, or at least at every calorimeter cell. Though these solve many of the largest problems there remain quirks with the cone algorithm.

D. *k_T Algorithm*

The k_t jet algorithm uses the knowledge that final state particles in a shower are largely collinear ie. have small transverse momentum between their constituent particles. The algorithm begins by creating a list of the momentum-space distance and the distance from the beam as,

$$d_{ij} = min(k_{ti}^2, k_{tj}^2)R_{ij}^2/D^2$$
$$R_{ij}^2 = (y_i - y_j)^2 + (\phi_i - \phi_j)^2 d_{iB} = k_{ti}^2$$

using the resolution parameter, D, to define roughly how large jets can become. It then finds the minimum of this list and if the distance is between two objects it merges them, if the distance is a beam distance then it calls this object a jet. This definition is theoretically motivated and consistent though it has seen far less use at hadron colliders and its systematic uncertainties are less well understood than those of the cone jet algorithms.

E. *k_T Algorithm Problems*

The errors inherent in the k_T jet algorithms are primarily due to its splash-in effect, or the fact that it tends to include lots of soft particles in the jet. In 2 the soft energy to the left is included in the k_T algorithm's definition of the jet. When compared to cone algorithms, k_T algorithms have the tendency to combine more energy into jets and the splash-in effect is hard to characterize between similar events. Thus, making corrections to the data analyzed with a k_T algorithm is harder to do in a consistent manner. k_T jet algorithms do not have a strict guideline for the final shape of their jets and this can be both a positive and a negative fact due the the potentially amorphous jets that are produced.

III. Algorithm Timing

Given any jet algorithm, it is important to understand both its quirks in reproducing jets as well as the time it takes to construct the jets from data. With the number of particles per even, N, of the order 10^2 in Tevatron it was less important for extremely optimized algorithms. In the LHC we will have $N \sim 10^3$ and we will also have ~20 events per collision, thus the speed of jet algorithms needs strong consideration so that the computations can keep up with data.

A. *JetClu Subprocesses*

The run 1 cone algorithm JetClu is a simple cone jet algorithm implementing seeds and without any of the aforementioned corrections to cone algorithms. The four main phases of the algorithm are

1) finding seed towers
2) Preclustering
3) Finding Stable Cones
4) Split/Merge

In the figure 4 with $N \sim 10^2$ the seed tower phase takes the longest and scales roughly as N. We find that preclustering takes a negligible amount of time and that finding stable cones scales roughly as $N^{1.5}$. The dominant process for N >10^3 is the split/merge phase, which scales roughly as $N^{2.3}$. Preclustering, a time saving technique that is actually physically motivated, was removed from JetClu when the Midpoint algorithms were written for CDF and DØ run 2. Preclustering assumes that energy is inherently spread out amongst neighboring calorimeter cells and groups cells together. It is important to notice that the preclustering has a positive effect on the final two stages, the reduction in cones passed on reduces the overall time to construct jets.

B. *Midpoint Subprocesses*

The midpoint algorithms follow a similar pattern and have 3 main stages

1) Find stable cones from seeds
2) Find midpoint cones
3) split/merge

whose timings are shown in 5. For $N \sim 10^2$ finding stable cones from seeds is dominant and scales as $N^{2.5}$. Finding midpoints scales roughly as N^3 and the dominant process for $N \sim 10^3$ is again split/merge that scales as $N^{3.5}$. In both the CDF and DØ implementations of this algorithm we find that it is slower than the JetClu implementation.

C. *Fastjet*

Fastjet **??** is an optimized version of the k_T jet algorithm. For $N > 10^3$ fastjet scales as $N \ln(N)$ and a more robust version scales as N^2. This algorithm's implementation of the CGAL libraries for efficient list management reduces the time necessary to combine particles and update the list. The effective time difference between Fastjet and the cone algorithms is only noticeable with N >10^3 though above this region the savings can be an order of magnitude or more. This begs the question, can we implement a similar technique on the cone algorithms to reduce their scaling to roughly $N \ln(N)$?

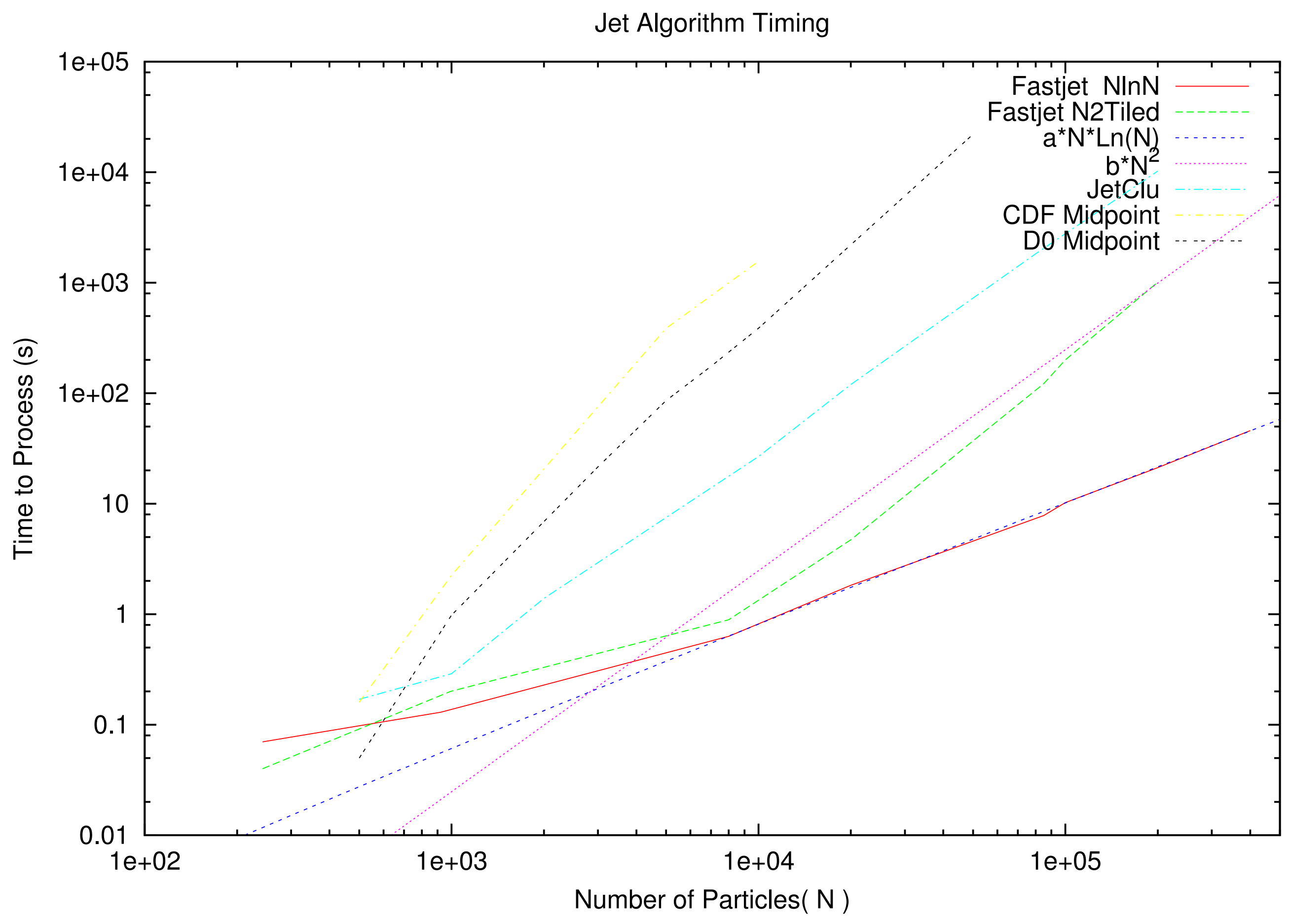

Fig. 3. Timing versus the number of particles per event, N, for various jet algorithms.

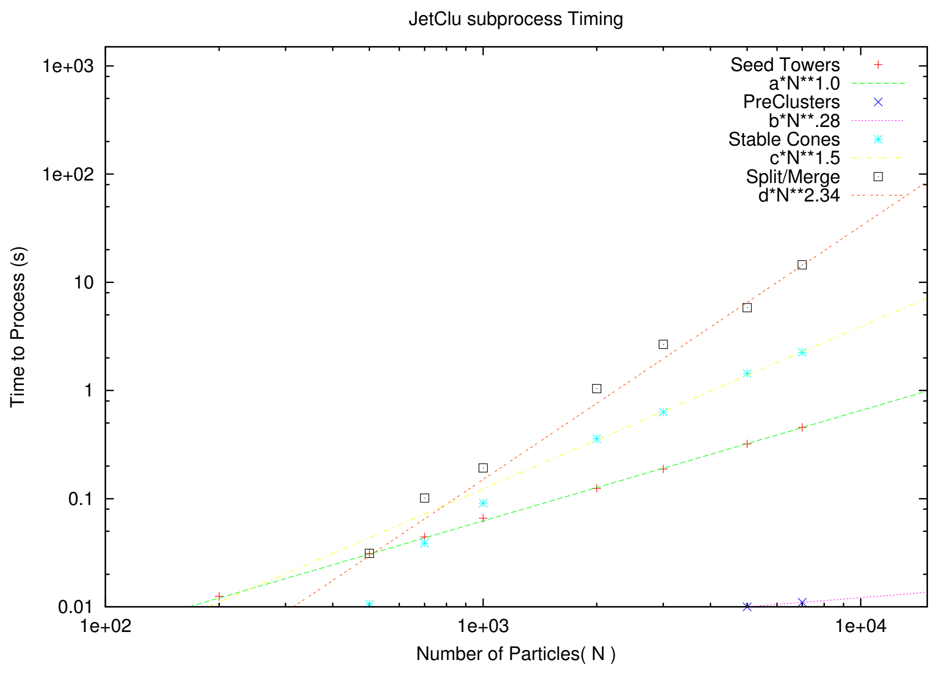

Fig. 4. Timing of the four subprocesses of the JetClu algorithm. The best fit lines give a rough idea of how time scales with N.

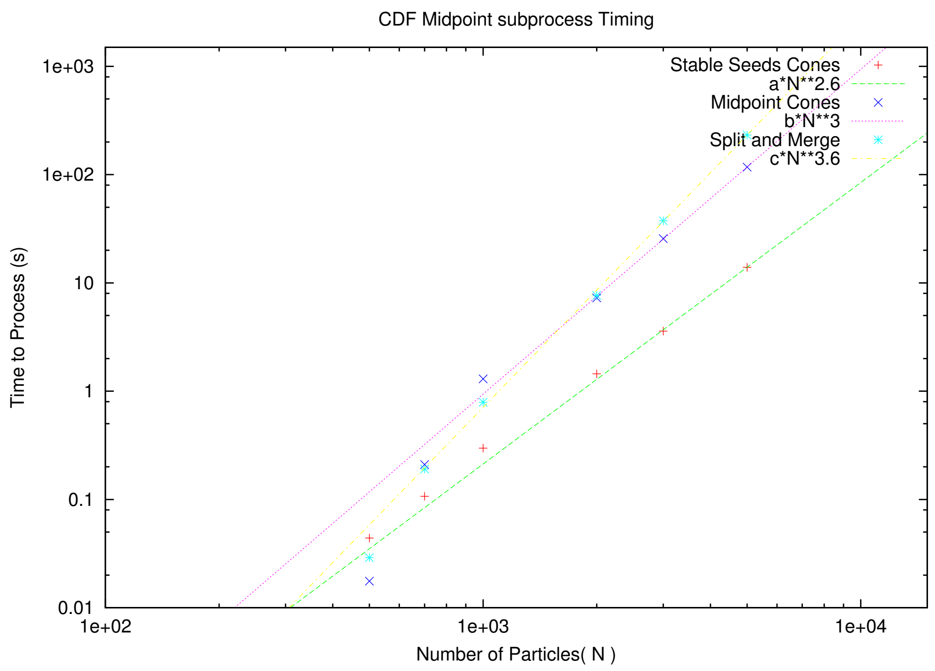

Fig. 5. CDF Midpoint subprocess timings, shows how the time of the various subprocesses scales with N.

IV. Conclusion

Throughout all of this discussion one must realize that jets will play an extremely important role in the LHC. The p^+p^+ collider will produce large number of events with ∼20 interactions per event and $N \sim 10^3$. Thus the large number of colored objects will also be produced and thus there will be an abundance of jets. Thus it is extremely important to have an efficient jet reconstruction whose uncertainties are small and easily corrected. Though there is no right way to define jets we must have a CONSISTENT jet definition that meshes well with experiment and theory.

Acknowledgment

Thank you to Steve Ellis, Matt Strassler, Kyle Armour, Jon Walsh and Rob Schabinger for letting me join their group for the summer.

References

[1] G. Salam and Matteo Cacciari, hep-ph/0512210
[2] S. D. Ellis and D. E. Soper, Phys. Rev. D 48 (1993) 3160 [hep-ph/9305266]. jets2 S. Catani, Y. L. Dokshitzer, M. H. Seymour and B. R. Webber, Nucl. Phys. B 406 (1993) 187
[3] Steve Ellis's web page, http://www.phys.washington.edu/users/ellis/