Particle Flow Status

Derek Anderson (ISU)

ePIC Collaboration Meeting

January 11th, 2024

Particle Flow Status | To-Do ca. July 2023 vs. Now

Particle Flow Discussion | To-Do

Missing Infrastructure (Major):
- PF Framework
  - Factories
  - Algorithm + configuration files
- Improved track-cluster associator
  - Extend to include Hcals
  - However, truth-based implementation may work for interim

Missing Infrastructure (Minor):
- PFObject Visualizer:
  - Plugin (or service?) to visualize clusters, tracks, etc.
  - Crucial for debugging
- Downstream analysis:
  - Code to look at impact of changes
  - Existing jet benchmarks are good starting place

Open Questions:
- Does implemented cluster splitting work in non-enabled* detectors?
- How well do existing MC-cluster associations work?
  - Currently handled by MatchClusters algorithm
  - Would a separate MC-cluster associator be better?

Major = necessary for implementation
Minor = can be pursued in parallel with implementation
Yellow = connection with other groups
* = existing implementation enabled for central ECals and ECalLumiSpec (not enabled for Imaging/SciFi)

July 13th, 2023 | Derek Anderson, ePIC Jet/HF WG | 10/10

- Slide from special Jet/HF meeting on PF
  - Was on July 13th, 2023
- Colors indicate current status:
  - ☑ **Green** = done
  - 💻 **Yellow** = in progress (& nearing completion)
  - ☒ **Red** = not started (but target for this workfest)

☞ Huge thanks to Tyler for implementing track-cluster matcher!

Particle Flow Status | PR and Github Links

- **PR #1186 now in draft**
  - Initial stab at implementing PFAlpha (see next slide)
  - Feel free to clone and test out!

- **Accomplishes:**
  - Creates infrastructure for PF (algorithm + corresponding factory)
  - "Improves" on existing `MatchClusters` algorithm by
    - Folding in HCal's in addition to ECal's
    - Using realistic track-cluster matching in contrast to using truth information

☑ = done 💻 = in progress ☒ = not started

- Work ongoing responding to feedback on PR1186
  - Link to development branch
  - Currently won't run (accidentally pushed bug to branch)

- **Improvements being made:**
  - ☑ Simplification of factory and conversion to JOmniFactory
  - ☑ Inclusion of energy thresholds
  - ☑ Simplification of user interface
  - ☑ Consolidation of tools into a namespace
  - 💻 Making code more expressive
  - ☒ Modularization to make code more maintainable
  - ☒ Guarding against wrong input types

PFAlpha | Overview

- **Particle Flow Alpha (PFAlpha):** a simplistic "bare bones" PFA
  - Will function as baseline to compare against more sophisticated PFAs
  - **And more importantly:** kickstarts general PF development
    - ☞ Both infrastructurally and conceptually

- **The gist:**
  1) Project tracks through calos
  2) Associate all calo clusters in cone of size R around track
  3) Sum all calo energy in cone and subtract expected track energy from sum
  4) Merge leftover clusters in cones of size R
  5) **Return PFObjects**
     - Tracks
     - Subtracted, merged clusters

Parameters
- R_{sum}^{ECal}: radius in (η, φ) in which to combine ECal clusters
- R_{sum}^{HCal}: same but for HCal
- f_{sub}^{ECal}: fraction of track energy to subtract from ECal clusters
- f_{sub}^{ECal}: same but for HCal

Oversight: minimum momentum of tracks or energy of clusters thresholds
- **Not in PR yet!**
  - ☞ Will be added in while responding to feedback

PFAlpha | Detailed Breakdown of Algorithm

1) **Subtract projected E_{trk} from ECal, HCal clusters**
   a) Identify seed (highest p_{trk}) track projection at inner face of ECal
   b) Sum E_{trk} of all projections in R_{sum}^{ECal}, R_{sum}^{HCal} of seed
   c) Sum E_{clust} of all ECal, HCal clusters in R_{sum}^{ECal}, R_{sum}^{HCal} respectively
   d) If $\Sigma E_{trk}^{ECal,HCal} < \Sigma E_{clust}^{ECal,HCal}$
      i. Subtract $f_{trk}^{ECal,HCal} \times E_{trk}^{ECal,HCal}$ of nearest projection from each cluster
      ii. Pass subtracted clusters onto step 2
   e) Repeat 1(a) – 1(d)(ii) until all projections have been used

2) **Combine remaining ECal, HCal clusters into topo-clusters**
   a) **Combine nearby ECal, HCal clusters**
      i. Identify seed (highest E_{clust}) ECal cluster
      ii. Merge all ECal, HCal clusters in R_{sum}^{ECal}, R_{sum}^{HCal} of seed
      iii. Repeat 2(a)(i) – 2(a)(iii) until no ECal clusters are left
   b) **Combine remaining HCal clusters**
      i. Identify seed HCal cluster
      ii. Add all HCal clusters in R_{sum}^{HCal} of seed
      iii. Repeat 2(b)(i) – 2(b)(iii) until no HCal clusters left
3) **Return PFObjects**

PFAlpha | EICrecon Implementation

```
// ----------------------------------------------------------------------------
//! Primary Algorithm Call
// ----------------------------------------------------------------------------
/*! The particular algorithm to be run for a pair of calorimeters is specified
 *  by the `flowAlgo` option. Returns collection of particle flow objects, i.e.
 *  tracks or  combinations of calorimeter clusters.
 */
std::unique_ptr<edm4eic::ReconstructedParticleCollection> ParticleFlow::process(
      TrkInput     inTrks,
      VecCaloInput vecInCalos,
      VecCaloIDs   vecCaloIDs
) {

  // set inputs
  m_inTrks     = inTrks;
  m_vecInCalos = vecInCalos;
  m_vecCaloIDs = vecCaloIDs;
  m_log -> trace("Running particle flow algorithm");

  // instantiate collection to hold produced reco particles
  m_outPFO = std::make_unique<edm4eic::ReconstructedParticleCollection>();

  // initialize track map
  initialize_track_map(m_inTrks.first, m_trkMap);

  // loop over pairs of input calos
  for (size_t iCaloPair = 0; iCaloPair < m_const.nCaloPairs; iCaloPair++) {

    // run selected algorithm
    //   - if unknown option is selected, throw exception
    switch (m_cfg.flowAlgo[iCaloPair]) {

      case FlowAlgo::Alpha:
        m_log -> trace("Running PF Alpha algorithm for calorimeter pair #{}", iCaloPair);
        do_pf_alpha(iCaloPair, m_vecInCalos[iCaloPair], m_vecCaloIDs[iCaloPair]);
        break;

      default:
        m_log -> error("Unknown PF algorithm option ({}) selected!", m_cfg.flowAlgo[iCaloPair]);
        throw JException("invalid argument");
        break;
    }
  }  // end calo pair loop

  // save unused tracks and return output collection
  save_unused_tracks_to_output(m_trkMap);
  m_log -> trace("Finished running particle flow algorithm");

  // return output collection
  return std::move(m_outPFO);

}  // end 'process(ParticleFlow::TrkInput, ParticleFlow::VecCaloInput, ParticleFlow::VecCaloIDs)'
```

- **Current EICrecon implementation:**
  - Input:
    - Reco. tracks
    - Track projections
    - Calo. clusters from all 6 calorimeters on central detector
  - Organizes cluster collections into 3 ECal-HCal pairs based on pseudorapidity
- Then choice of PFA to process each pair of calorimeters is a user-configurable parameter
  - **Left:** primary algorithm call for PF factory
  - **Rationale:**
    - This allows each region of detector to be processed differently (reflecting different physics needs)
    - And particular algorithms can be switched out on fly

PFAlpha | EICrecon Implementation

```
namespace eicrecon {

  struct ParticleFlowConfig {

    // Note: the vector elements correspond
    // to different eta regions
    //   [0] = negative
    //   [1] = central
    //   [2] = positive
    std::vector<int>   flowAlgo      = {0, 0, 0};        // choice of particle flow algorithm
    std::vector<float> ecalSumRadius = {1.0, 1.0, 1.0};  // radius of cone to sum energy from ecal clusters
    std::vector<float> hcalSumRadius = {1.0, 1.0, 1.0};  // radius of cone to sum energy from hcal clusters
    std::vector<float> ecalFracSub   = {1.0, 1.0, 1.0};  // fraction of track energy to subtract from ecal sum
    std::vector<float> hcalFracSub   = {1.0, 1.0, 1.0};  // fraction of track energy to subtract from hcal sum

  };

}  // end eicrecon namespace
```

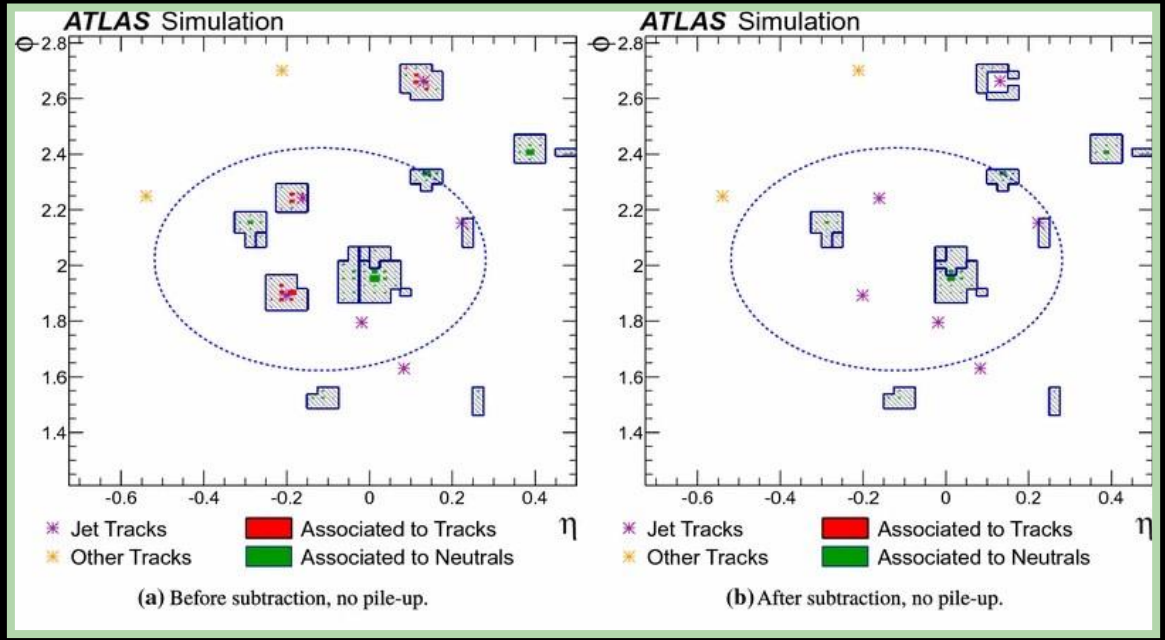
- **But this is already unwieldy for a simple PFA**
  - Switching to JOmniFactory made it easy to into 3 factories
    - 1 for each eta region
  - Vastly simplifies user interface!
    - ☞ Only 5 parameters instead of 15!
- **Furthermore:** subtraction and merging steps in same algorithm will quickly bloat implementation file
  - **Already big with only PFAlpha!**
  - It could be prudent to split steps into 2 separate algorithms…

To-Do | Potential Workfest Tasks

1) Developing debugging and visualization tools

- AFAIK no general-purpose code to concisely visualize particles, tracks, and clusters at once
  - eg. something like ATLAS (shown right)
- Tools to do this would be HUGE help in debugging/performance studies
- Prepared (tiny) DD4Hep/EICrecon output with PF collections for development

RCF Path: /sphenix/user/danderson/jets/ForPFWorkfest



ATLAS [arXiv:1703.10485]

2) Continuing work on PR comments

To-Do | Final Comments

IF YOU SEE SOMETHING, SAY SOMETHING

- Please let me know if you see anything dumb or that can be improved!
  - There haven't been many eyeballs on this code yet
  - There are **much** better ways of doing things than I thought of
  - And even though PFAlpha is a baseline, it can and **should** be improved!

WE NEED YOU

- Particle Flow is a **BIG** project
  - Baselines will be in soon, but this is a challenging subject
  - And development will continue for the foreseeable future
  - There is **LOTS** of room to contribute to this effort
- And if you think of something PF-related you'd like to see implemented, talk to me!
  - ☞ I'll work with you to see that this gets implemented!

Discussion

- Note A
- Note B