PFAX Tuning | Purity Calculation

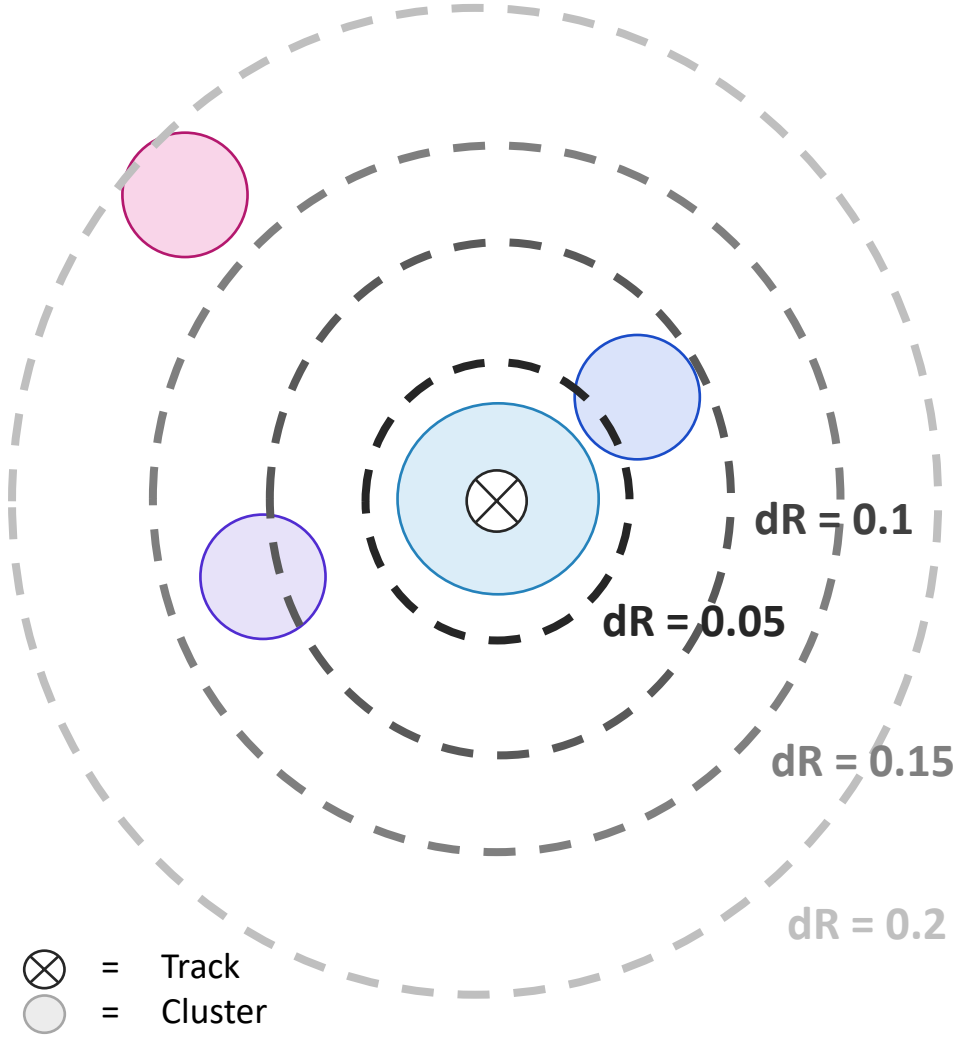
- **Purity:** fraction of energy deposited by a particle contained in a cluster
  - Directly correlated with E/p_{trk} and E/E_{par}, but properly normalized since only a fraction of E_{par} will be deposited in a calorimeter
  - Will be used to set thresholds in PFA0 and PFA2

- **Algorithm** (for single particle events)**:**
  1. **Let** eDepPar **=** 0
  2. **Let** mapClust be a map of cluster index onto purities
  3. **For each** cluster with index iClust **do:**
     - A. **Let** eDepClust **=** 0
     - B. **For each** recHit **in** cluster.getHits() **do:**
       - i. **Let** rawHit **=** recHit.getRawHit()
       - ii. **For each** simHit **in** links(rawHit) **do:**
         - a. eDepClust **+=** simHit.getEnergy()
         - b. eDepPar **+=** simHit.getEnergy()
     - C. mapClust[iClust] **=** eDepClust
  4. **For each** entry with index i **in** mapClust **do:**
     1. mapClust[i] **=** mapClust[i] / eDepPar

- **Notes:**
  - eDepPar is total amount of energy deposited by particle
  - eDepClust is total amount of deposited energy contained in a cluster
  - Notice the hit names, there are 3 kinds:
    - **Sim Hits**, which are sums of energy deposits in a calorimeter cell
      - e.g. EcalEndcapPHits
    - **Raw Hits**, which are these sums *after* digitization
      - e.g. EcalEndcapPRawHits
    - **Reco Hits**, which are the calorimeter cells *after* reconstructing their energies
      - e.g. EcalEndcapPRecHits
  - The line 3(A)(ii) is shorthand for using the link navigator
    - The associations between raw and sim hits are stored in **"*RawHitLinks"** collections
      - e.g. EcalEndcapPRawHitLinks
    - We're in the middle of switching over from associations (like I was using in this macro) to links
    - And the navigator will make working with these *much* easier!
  - For DIS events, will need to use CaloHitContributions. **NOT** saved in campaign output!
    - So will need to run sim + reco on your own…

PFAO Tuning | Merging Window Algorithm

- $\langle E/p \rangle$, $n\sigma(E/p)$ determined directly from E/p distribution
  - ☞ But determining the merging window (dR) needs some additional work…
  - ☞ We also want to see how certain quantities (X) change as increase dR!
- **Algorithm:**
  1. **For each** track-cluster match **do**
     - A. **Compute** nSigma with respect to avgEoP
     - B. **Skip if** nSigma below threshold
     - C. **For each** n **in** range(0, N) **do**
       - i. **Let** dR **=** n*dRstep
       - ii. **Let** eSum **=** track-cluster.getEnergy()
       - iii. **For each** cluster **do**
         - a. **Skip if** cluster **in** track-cluster match
         - b. **Compute** dist(cluster, track projection)
         - c. **If** dist **<** dR ,
           - › eSum **+=** cluster.getEnergy()
       - iv. **Compute** X with eSum
       - v. **Plot** eSum, X vs. n, dR
- **Parameters:**
  - **avgEoP** = mean of main peak of E/p distribution
  - **threshold** = min. nSigma for a “good” track-cluster match
  - **N** = no. of iterations
  - **dRstep** = step size between merging windows



PFA0 Tuning | Quantities to Check

- **What particles to use for single particle events?**
  - EMCal = $e^{\pm}$
  - HCal = $\pi^{\pm}$

- **What quantities to do we want to check vs. dR?**
  - E/p_{trk}, where p_{trk} is the track momentum
  - $n\sigma(E/p_{trk})$ with respect to main peak of E/p distribution
  - $f = E/E_{par}$, where E_{par} is the particle energy
  - $\varepsilon = E_{dep}/E_{par}^{dep}$, i.e. purity (slide 1)

☞ Will need to also determine where to set $n\sigma(E/p)$ threshold:
  - Can determine that by looking at by looking at
    - **E/p_{trk} vs. purity**
    - **$n\sigma(E/p)$ vs. purity**
  - Should see distinct populations (e.g. fig.s 11, 14 from the ATLAS PF Paper)
  - Other quantities that could be useful to plot vs. purity:
    - E_{clust}
    - η_{clust}
    - $f = E/E_{par}$
    - $\Delta r_{track-clust}$

- **What calorimeters to check?**
  - **Backwards HCal** (NHCal, id = 113)
    - Clusters = HcalEndcapNClusters
    - Matches = HcalEndcapNTrackClusterMatches
  - **Backwards EMCal** (EEEMCal, id = 103)
    - Clusters = EcalEndcapNClusters
    - Matches = EcalEndcapNTrackClusterMatches
  - **Barrel HCal** (BHCal, id = 111)
    - Clusters = HcalBarrelClusters
    - Matches = HcalBarrelTrackClusterMatches
  - **Barrel EMCal** (BIC, id = 101)
    - Clusters = EcalBarrelClusters
    - Matches = EcalBarrelTrackClusterMatches
  - **Forward HCal** (LFHCal, id = 112)
    - Clusters = LFHCALClusters
    - Matches = LFHCALTrackClusterMatches
  - **Forward EMCal** (FEMC, id = 102)
    - Clusters = EcalEndcapPClusters
    - Matches = EcalEndcapPTrackClusterMatches
  - **Forward HCal Insert** (Insert, id = 115)
    - Clusters = HcalEndcapPInsertClusters
    - Matches = HcalEndcapPInsertTrackClusterMatches

PFA1 Tuning | Sub Fraction and Surface

- PFA1 not as algorithmically challenging to tune (no merging windows to deal with)
  - For initial working point, two primary quantities to tune:
    - s_{use} = which projection surface (1 or 2) to use, and
    - f_{sub} = fraction of track energy to subtract
    - ☞ Can explore using resolutions for subtraction in follow-up
  - **Goal is to get $E_{clust} - f_{sub}E_{trk}$ as close to 0 as possible for charged particles!**

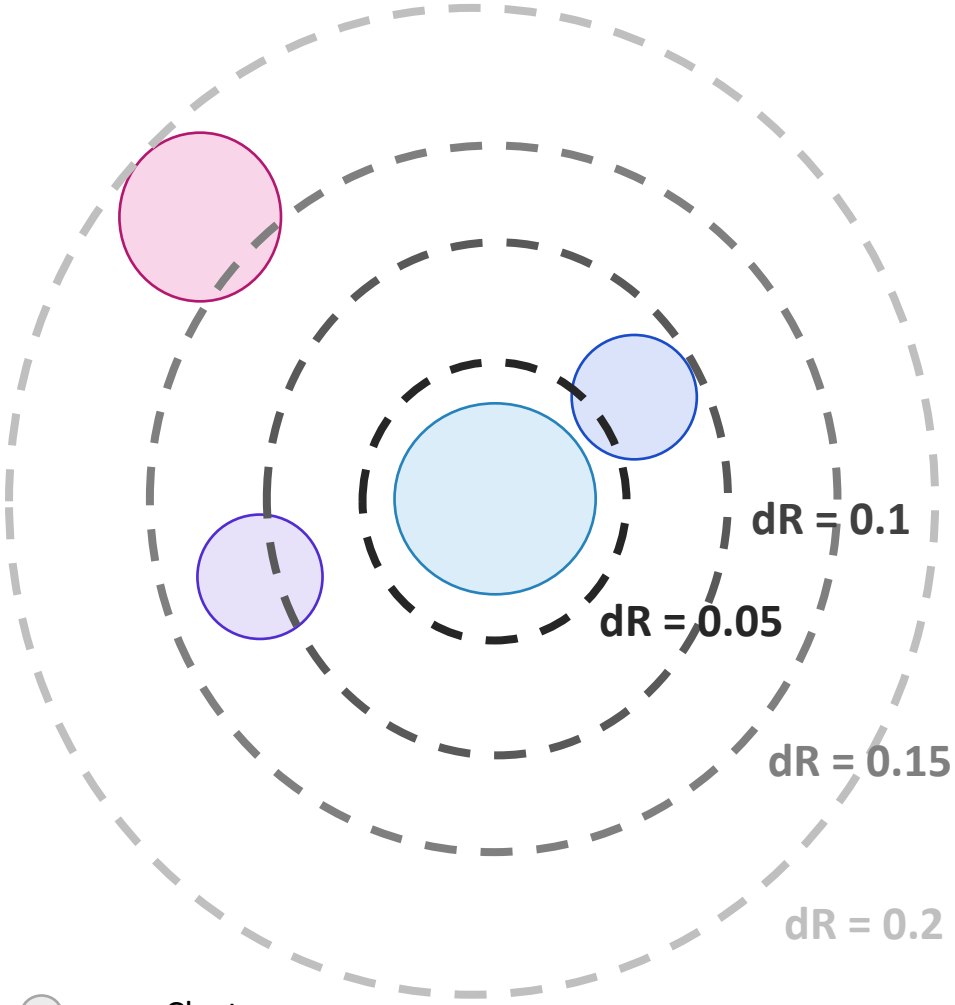
- **How to tune?** For each calorimeter do:
  1) Plot E_{clust}/E_{trk}
  2) Set f_{sub} to $\langle E_{clust}/E_{trk} \rangle$
  3) Plot $E_{clust} - f_{sub}E_{trk}$ for $s_{use} = 1, 2$
- **What particles to use for single particle events?**
  - EMCal = $e^{\pm}$
  - HCal = $\pi^{\pm}$

- **Also need help in tuning track-cluster match distance for barrel, forward endcap!** To get rough working point:
  1) Plot E/p, set $thresh$ to be max(0.1, inflection point)
  2) Plot dist(track, cluster) for $E/p > thresh$
  3) Set distance to be at edge of main peak of dist(track, cluster)

- **What calorimeters to check?**
  - **Backwards HCal** (NHCal, id = 113)
    - Clusters = HcalEndcapNClusters
    - Matches = HcalEndcapNTrackClusterMatches
  - **Backwards EMCal** (EEEMCal, id = 103)
    - Clusters = EcalEndcapNClusters
    - Matches = EcalEndcapNTrackClusterMatches
  - **Barrel HCal** (BHCal, id = 111)
    - Clusters = HcalBarrelClusters
    - Matches = HcalBarrelTrackClusterMatches
  - **Barrel EMCal** (BIC, id = 101)
    - Clusters = EcalBarrelClusters
    - Matches = EcalBarrelTrackClusterMatches
  - **Forward HCal** (LFHCal, id = 112)
    - Clusters = LFHCALClusters
    - Matches = LFHCALTrackClusterMatches
  - **Forward EMCal** (FEMC, id = 102)
    - Clusters = EcalEndcapPClusters
    - Matches = EcalEndcapPTrackClusterMatches
  - **Forward HCal Insert** (Insert, id = 115)
    - Clusters = HcalEndcapPInsertClusters
    - Matches = HcalEndcapPInsertTrackClusterMatches

PFA2 Tuning | Merging Window Algorithm

- Similar to PFA0 algorithm, but have 2 merging windows to determine!
- **Algorithm:**
  1. **Sort** EMCal clusters, HCal clusters by decreasing energy
  2. **For each** unused cluster* **in** EMCal **do**
     - A. **For each** n **in** range(0, N) **do**
       - I. **For each** m **in** range(0, M) **do**
         - a. **Let** dREM = n*dREMstep
         - b. **Let** dRH = n*dRHstep
         - c. **Let** eSum = cluster.getEnergy()
         - d. **For each** unused cluster **in** EMCal **do**
           - i. **Skip if** cluster = cluster*
           - ii. **Compute** dist(cluster*, cluster)
           - iii. **If** dist < dREMstep
             - › eSum += cluster.getEnergy()
         - e. **For each** unused cluster **in** HCal **do**
           - i. **Compute** dist(cluster*, cluster)
           - ii. **If** dist < dRHstep
             - › eSum += cluster.getEnergy()
         - f. **Compute** X with eSum
         - g. **Plot** eSum, X vs. n, m, dREM, drH
  3. **For each** unused cluster* **in** HCal **do**
     1. **Repeat 2A** excluding EMCal lines as needed
- **Parameters:**
  - **N, M** = no. of EMCal, HCal iterations
  - **dRXstep** = step size between merging windows for X calo



= Cluster
= Size ~ cluster energy

PFA2 Tuning | Quantities to Check

- **What particles to use for single particle events?**
  - EMCal only = γ
  - EMCal + HCal = K_L^0, n

- **What quantities to do we want to check vs. dR?**
  - $f = E/E_{par}$, where E_{par} is the particle energy
  - $\varepsilon = E_{dep}/E_{par}^{dep}$, i.e. purity (slide 1)

☞ I have less intuition about how these quantities behave for neutrals only, so will need some exploration…
  - Quantities that could be useful to plot vs. purity:
    - E_{clust}, E_{clust^*}
    - η_{clust}, η_{clust^*}
    - $f = E/E_{par}$
    - $\Delta r_{clust^*-clust}$, Δr_{clust^*-par}
  - Will need to balance recovering all visible energy (getting to $\varepsilon \sim 1$) vs. not merging showers
    - Might be useful to plot a few quantities for each calo in DIS to get a feel for this
      - (eta, phi) of CaloHitContributions
      - (eta, phi) of neutral primary particles (GeneratedParticles)

- **What calorimeters to check?**
  - **Backwards HCal** (NHCal, id = 113)
    - Clusters = HcalEndcapNClusters
    - Matches = HcalEndcapNTrackClusterMatches
  - **Backwards EMCal** (EEEMCal, id = 103)
    - Clusters = EcalEndcapNClusters
    - Matches = EcalEndcapNTrackClusterMatches
  - **Barrel HCal** (BHCal, id = 111)
    - Clusters = HcalBarrelClusters
    - Matches = HcalBarrelTrackClusterMatches
  - **Barrel EMCal** (BIC, id = 101)
    - Clusters = EcalBarrelClusters
    - Matches = EcalBarrelTrackClusterMatches
  - **Forward HCal** (LFHCal, id = 112)
    - Clusters = LFHCALClusters
    - Matches = LFHCALTrackClusterMatches
  - **Forward EMCal** (FEMC, id = 102)
    - Clusters = EcalEndcapPClusters
    - Matches = EcalEndcapPTrackClusterMatches
  - **Forward HCal Insert** (Insert, id = 115)
    - Clusters = HcalEndcapPInsertClusters
    - Matches = HcalEndcapPInsertTrackClusterMatches

PFA3 Tuning | Sub NCalo and

- PFA3 also not as algorithmically challenging to tune.
  - For initial working point, two primary quantities to tune:
    - N_{calo} = normalization of total calo energy, and
    - s_{had} = scale between EM/Hadronic energy
    - ☞ Can explore using resolutions for subtraction in follow-up
  - **Goal is to get as close to E_{par} as possible for neutral particles!**
    - ☞ Will use E_{trk} for charged particles for initial working point

- **How to tune?** For each pair of EM-H calos
  1) Find s_{had} st. $(E_{em} + s_{had}E_{had})/E_{par}$ has the smallest σ/μ
  2) Set N_{calo} to be $1/\mu$ of $(E_{em} + s_{had}E_{had})/E_{par}$ distribution w/ optimal s_{had}
- **What particles to use for single particle events?**
  - EMCal only = γ
  - EMCal + HCal = K_L^0, n

- **Note:** rigorous tuning of s_{had}, N_{calo} is a **VERY** involved process! Goal here is to just get a "good enough" starting point
  - ☞ Jan's done a lot of work on this for the barrel! (See here)

- **What calorimeters to check?**
  - **Backwards HCal** (NHCal, id = 113)
    - Clusters = HcalEndcapNClusters
    - Matches = HcalEndcapNTrackClusterMatches
  - **Backwards EMCal** (EEEMCal, id = 103)
    - Clusters = EcalEndcapNClusters
    - Matches = EcalEndcapNTrackClusterMatches
  - **Barrel HCal** (BHCal, id = 111)
    - Clusters = HcalBarrelClusters
    - Matches = HcalBarrelTrackClusterMatches
  - **Barrel EMCal** (BIC, id = 101)
    - Clusters = EcalBarrelClusters
    - Matches = EcalBarrelTrackClusterMatches
  - **Forward HCal** (LFHCal, id = 112)
    - Clusters = LFHCALClusters
    - Matches = LFHCALTrackClusterMatches
  - **Forward EMCal** (FEMC, id = 102)
    - Clusters = EcalEndcapPClusters
    - Matches = EcalEndcapPTrackClusterMatches
  - **Forward HCal Insert** (Insert, id = 115)
    - Clusters = HcalEndcapPInsertClusters
    - Matches = HcalEndcapPInsertTrackClusterMatches

Backup | PFA0 | Track-Cluster Merge/Splitter

- **Track-Cluster Merging/Splitting:** merges and then splits clusters based on track projections
  - Algorithm based on ATLAS's split recovery procedure
    - c.f. Eur. Phys. J. C (2017) 77:466
  - Implemented in EICrecon#1406, updating in EICrecon#1699

- **The algorithm:**
  1) Match track projection to cluster
  2) If matched, calculate significance b/n E_{clust} energy & expected E_{dep}:

$$S(E_{clust}) = \frac{E_{clust} - (p_{proj} \times \langle E/p \rangle)}{\sigma(E_{dep})}$$

  3) If $S < S_{cut}$, add clusters inside Δr_{add}
  4) If multiple tracks pointing to merged cluster:
     3) Split into one cluster for each track & reweight transverse shape by p_{trk}, track projection

Inputs:
- Clusters
- Track projections (**to-do:** track-cluster matches)
- **Optional:** reco-sim hit associations (or sim hits)

Outputs:
- Merged/Split clusters
- Track-merged/split cluster matches
- **Optional:** scale for transverse shape reweighting

Parameters:
- $\langle E/p \rangle$: average E/p
- $\sigma(E_{dep})$: spread of dep. energy
- S_{cut}: threshold to run split-recovery
- Δr_{add}: window to add clusters
- σ_{trk}: scale for transverse shape reweighting

Backup | PFA1(a) | Track-Cluster Subtractor (1/2)

- **Track-Cluster Subtractor:** subtracts momentum of matched track(s) from cluster
  - In progress at EICrecon#1627

- **The algorithm:**
  1) Build map of clusters onto *all* matched tracks
  2) For each cluster:
     a) Sum energy of matched tracks:

$$E_{trk} = \sum p_{trk}(S_{use}) \oplus m_{trk}$$

     b) Subtract sum: $E_{sub} = E_{clust} - f_{sub}E_{trk}$
     c) If NOT consistent w/ 0,
        - Create remnant cluster w/ E_{sub}
        - Set subtracted cluster energy to $E_{clust} - E_{sub}$
     d) Create an association for each track matched to subtracted cluster

Inputs:
- Track-cluster matches
- Track projections

Outputs:
- Subtracted clusters ($\sim E_{clust}$)
- Remnant clusters ($E_{clust} - E_{trk}$)
- Track-subtracted

Parameters:
- f_{sub}: fraction of track energy to subtract
- $m_{default}$: default mass to use for track energy
- S_{use}: surface to evaluate track momentum at
- $k_{do\ n\sigma?}$: turn on/off checking against resolutions
- $n\sigma_{cut}$: max no. of sigmas to be consistent w/ 0
- σ_{cal}: calo resolution to use in n-sigma cut

Backup | PFA1(a) | Track-Cluster Subtractor (2/2)

- **Track-Cluster Subtractor:** subtracts momentum of matched track(s) from cluster
  - In progress at EICrecon#1627

- **The algorithm:**
  1) Build map of clusters onto *all* matched tracks
  2) For each cluster:
     a) Sum energy of matched tracks:

$$E_{trk} = \sum p_{trk}(S_{use}) \oplus m_{trk}$$

     b) Subtract sum: $E_{sub} = E_{clust} - f_{sub}E_{trk}$
     c) If NOT consistent w/ 0,
        - Create remnant cluster w/ E_{sub}
        - Set subtracted cluster energy to $E_{clust} - E_{sub}$
     d) Create an association for each track matched to subtracted cluster

Sub-routine: is E_{sub} consistent w/ zero?

1) **If $E_{sub} < 0$, YES**
2) **Else if $k_{do\ n\sigma?}$**
   a) Calculate $n\sigma$ (σ_{trk} from cov. matrix)

$$n\sigma = \frac{E_{sub}}{\sigma_{trk} \oplus \sigma_{cal}}$$

   b) **If $n\sigma < n\sigma_{cut}$, YES**
3) **Else**
   a) **If $E_{sub} < \epsilon$, YES**

Note: epsilon here is std::numeric_limits<double>::epsilon()

Backup | PFA1(b) | Charged Candidate Maker

- **Charged Candidate Maker:** forms track-cluster matches into a charged particle candidate
  - In progress at EICrecon#2124

- **The algorithm:**
  1) Build map of tracks onto ***all matched clusters***
  2) For each **track:**
     a) For each matched cluster:
        i. Identify if in an ECal or an HCal by checking system ID
        ii. Select relevant weight
        iii. Add to relevant members
     b) Add to relevant member

Inputs:
- Track-cluster matches

Outputs:
- Charged particle candidates

Parameters:
- None!

Backup | PFA2 | Calo Remnant Combiner

- **Calo Remnant Combiner:** combines remnant clusters from subtractor into neutral particle candidates
  - PR open at EICrecon#2195

- **The algorithm:**
  1) Combine nearby ECal, HCal clusters
     a) Identify seed ECal cluster
     b) Merge all ECal, HCal clusters in Δr_{add}^{em}, Δr_{add}^{h} of seed and create neutral candidate
     c) Repeat until no ECal clusters are left
  2) Combine remaining HCal clusters
     a) Identify seed HCal cluster
     b) Add all HCal clusters in Δr_{add}^{h} of seed and create neutral candidate
     c) Repeat until no HCal clusters are left

Inputs:
- Remnant ECal clusters
- Remnant HCal clusters

Outputs:
- Neutral particle candidates

Parameters:
- Δr_{add}^{em}: window to add ECal clusters
- Δr_{add}^{h}: window to add HCal clusters

Backup | PFA3 | Candidate Particle Convertor (1/2)

- **Particle Converter:** takes candidate particles and turns them into reconstructed particles
  - In progress at EICrecon#2399

- **The algorithm:**
  1) Assign preliminary PID based on what info is available (next slide)
  2) Calculate track energy
  $$E_{trk} = p_{trk} \oplus m_{pid}$$
  3) Calculate calorimeter energy
  $$E_{cal} = N_{cal}\left(\sum w_{em}E_{em} + \sum w_h E_h\right)$$
  4) If charged particle and $k_{use\ \sigma?}$, calculate resolution-weighted average of E_{cal} and E_{trk}
  5) Calculate remaining kinematics and create reconstructed particle

Inputs:
- Candidate charged/neutral particles
- Reconstructed charged particles (for PID)
- Primary vertices (for neutral candidates)

Outputs:
- Reconstructed particles

Parameters:
- $k_{use\ \sigma?}$: turn on/off using resolution in energy calculation for charged candidates
- N_{cal}: normalization of calo energy
- σ_{trk}: tracking resolution to use in energy calc (likely will use cov. matrix instead)
- σ_{cal}: calo resolution to use in energy calc

Backup | PFA3 | Candidate Particle Convertor (2/2)

- **Particle Converter:** takes candidate particles and turns them into reconstructed particles
  - In progress at EICrecon#2399

- **The algorithm:**
  1) Assign preliminary PID based on what info is available (next slide)
  2) Calculate track energy
  $$E_{trk} = p_{trk} \oplus m_{pid}$$
  3) Calculate calorimeter energy
  $$E_{cal} = N_{cal}\left(\sum w_{em}E_{em} + \sum w_h E_h\right)$$
  4) If charged particle and $k_{use\ \sigma?}$, calculate resolution-weighted average of E_{cal} and E_{trk}
  5) Calculate remaining kinematics and create reconstructed particle

Sub-routine: what PDG to assign?

1) Check what info is present:
   a) **If** at least 1 related track, **hasTrk = TRUE**
   b) **If** at least 1 EMCal cluster, **hasECal = TRUE**
   c) **If** at least 1 HCal cluster, **hasHCal = TRUE**
2) **If hasTrk**
   a) Use track to retrieve reco charged particle
   b) PDG = chrgPar.getPDG()
3) **Else**
   a) **If hasECal and !hasHCal,** PDG = <Photon>
   b) **If hasECal and hasHCal,** PDG = <Neutron>
   c) **If !hasECal and hasHCal,** PDG = <???>

Note: for initial pass electron/muon and pi0/gamma discrimination deferred to downstream
   ☞ Also can refine neutral PID by checking relative ECal vs. HCal contribution