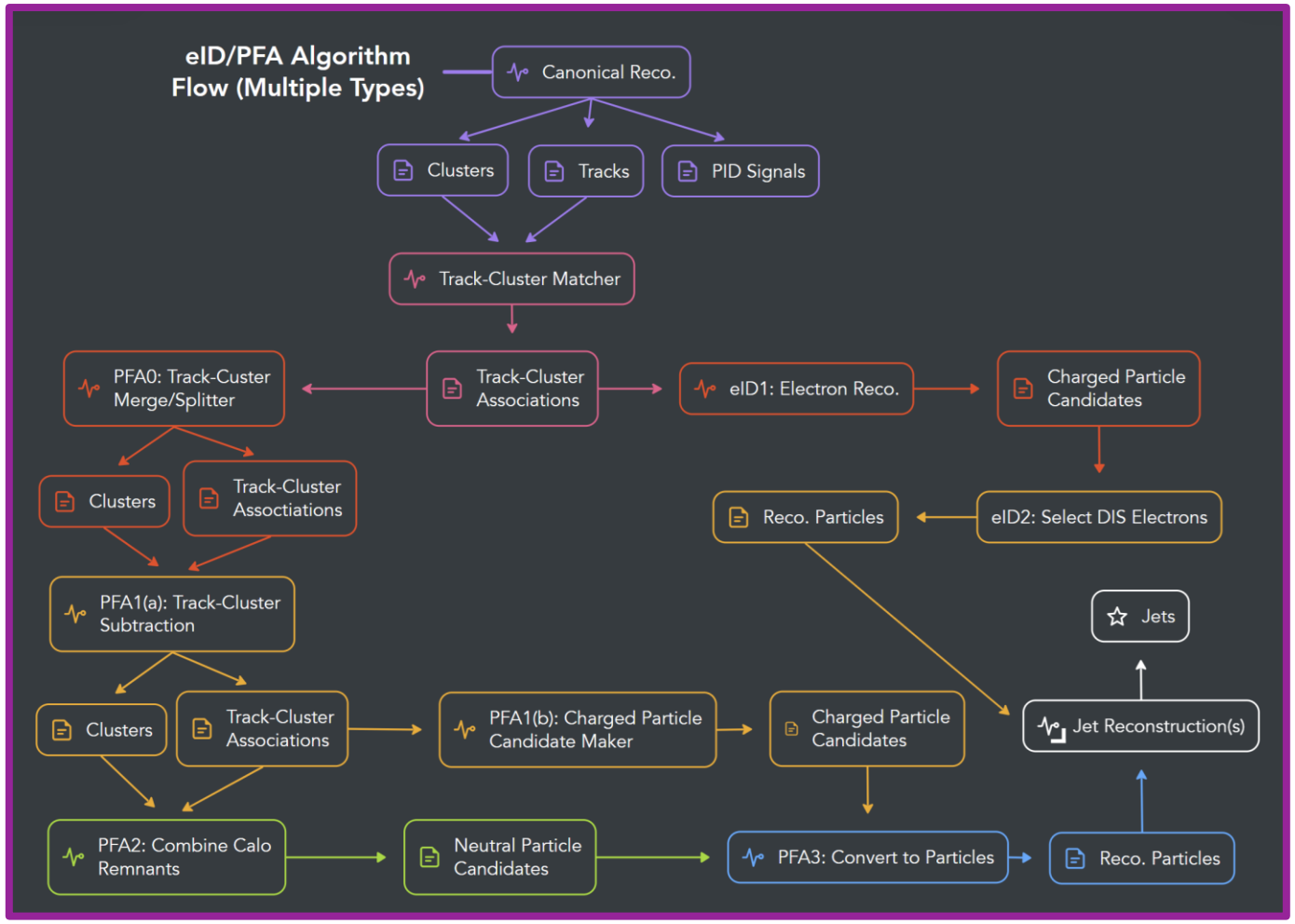
PF Tech | PFAlpha Overview

PFAlpha: the proposed baseline, looks like:

1) **[PFA-1]** Match tracks to EMCal, HCal clusters
2) **[PFA0]** Merge clusters based on track E/p in a cone of size R_0
3) **[PFA1a]** Subtract expected track energy from merged clusters
   - Split into tracks + expected energy, and remnant clusters (leftover energy)
4) **[PFA1b]** Convert tracks + expected energy to particle candidates
5) **[PFA2]** Combine remnant EMCal, HCal clusters in a cone of size R_1, convert to particle candidate
6) **[PFA3]** Covert candidates to reconstructed particles



PF Tech | Current Status

- **Last update:** EIC UGM Jet/HF Workfest
  - ☞ See here!
- **Updates since:**
  - JANA2 2.4.3 release resolves issue with link collections
    - ☞ Cf. JANA2#462: link collections' templating exposed unused, erroneous JFactorPodioT::Insert method
  - Path to completing PFA0 is now clear!

- **PFA0 next steps:** EICrecon#1699
  - Confirm link bugfix works
  - Wire in output of track-cluster matcher, remove duplicated code

- **Post-PFA0 next steps:**
  - Implement "promotion" algorithm to convert Track-Protocluster Links to Track-Cluster Matches
  - Revive and finish PFA1 (EICrecon#1627)
  - Complete particle candidate discussion & implement decided changes
    - ☞ Discuss today
  - Begin PFA2 (calo remnant combiner)
    - ☞ Discuss today

PF Tech | Related Threads

- **Threads in-progress:**
  - Wiring in track-cluster matches in all relevant algorithms (incl. eID)
  - Deprecation of TrackClusterMatch
    - This is a *major* truth-info leak, but centralized track-cluster matches lets us patch this
    - Can replace existing algorithm with mostly identical one using matches
      - 👉 Cf. EICrecon#1956

- **Threads to begin:**
  - Upgrade Track-Cluster Matches from *associations* to *links*
  - Cross-calo clustering and/or cross-calo combining
  - Integration of David's BHCal ML model

- **Note:** goal of PFAlpha is to bring the following improvements over existing TrackClusterMatch
  - Integration of HCals
  - More balanced handling of tracker-vs-calo energy

PF Tech | Discussion: Candidate Types

```yaml
## A charged particle candidate
edm4eic::ChargedRecoParticleCandidate:
  Description: "Candidate charged reconstructed particle"
  Author: Tyler Kutz, Derek Anderson, Shujie Li
  OneToOneRelations:
    - edm4eic::Track track  // reconstructed track...other relations are matched to this
  OneToManyRelations:
    - edm4hep::ParticleID particleIDs  // associated particle IDs
    - edm4eic::Cluster    ecalClusters // ECAL clusters matched to this track
    - edm4eic::Cluster    hcalClusters // HCAL clusters matched to this track
  VectorMembers:
    - float ecalWeights  // weights of matched ecal clusters
    - float hcalWeights  // weights of matched hcal clusters
```

```yaml
## A neutral particle candidate
edm4eic::NeutralRecoParticleCandidate:
  Description: "Candidate neutral reconstructed particle"
  Author: Tyler Kutz, Derek Anderson, Shujie Li
  OneToManyRelations:
    - edm4hep::ParticleID particleIDs  // associated particle IDs
    - edm4eic::Cluster    ecalClusters // associated ECAL clusters
    - edm4eic::Cluster    hcalClusters // associated HCAL clusters
  VectorMembers:
    - float ecalWeights  // weights of associated ecal clusters
    - float hcalWeights  // weights of associated hcal clusters
```

- **Critical Idea from 2024 UGM Workfest:** a *pseudoparticle/candidate particle* type
  - In spirit, similar to a protocluster *but* for reco. particles
  - Brings together needed track + clusters with weights ahead of final reconstruction step
  - **Above:** current state of proposal

- **Alternative proposal from last discussion:**
  - Merging ECal & HCal fields will make ReconstructedParticle a superset of both
  - Could instead utilize cluster's `type` field or add extra code to quickly flag a cluster as an ECal or HCal…

PF Tech | Discussion: PFA2 Development

- **Calo Remnant Combiner:** combines remnant clusters from subtractor into neutral particle candidates
  - Still to-do, so how to handle rollout?
  - Could begin developing in parallel to PFA0-1 leg…

- **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

Note: maybe make inputs vectors of collections?

Parameters:
- $\mathbf{\Delta r_{add}^{em}}$**:** window to add ECal clusters
- $\mathbf{\Delta r_{add}^{h}}$**:** window to add HCal clusters
- $\{w_{em}\}$**:** weights (or weight) of ECals to use
- $\{w_{h}\}$**:** weights (or weight) of HCals to use

PF Tech | Discussion: Benchmarking

- **We need to benchmark!** So far very much neglected…
  - Want to collect folks' thoughts on good benchmarks/quantities to plot

- **Benchmarks:**
  - Energy conservation between steps (ie. total calo energy before vs. after)
  - E/p for charged particles
  - Number of clusters vs. number of tracks at each steps
  - PES/PER (only after PFA3)
  - Jet observables:
    - JES/JER (already implemented)
    - Substructure observables
      - Angularity?
      - EECs?

- **Benchmarks (cont.):**
  - Jet observables (cont.):
    - D0/HF-tagged jet FFs
  - Event observables:
    - NECs

- **Important point:** keep existing algorithm(s) in place
  - Ie. both truth-based TrackClusterMatch *and* reco-based equivalent
  - Run benchmarks on output from all 3 – truth-based, reco-based, and PFAlpha – and compare directly

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

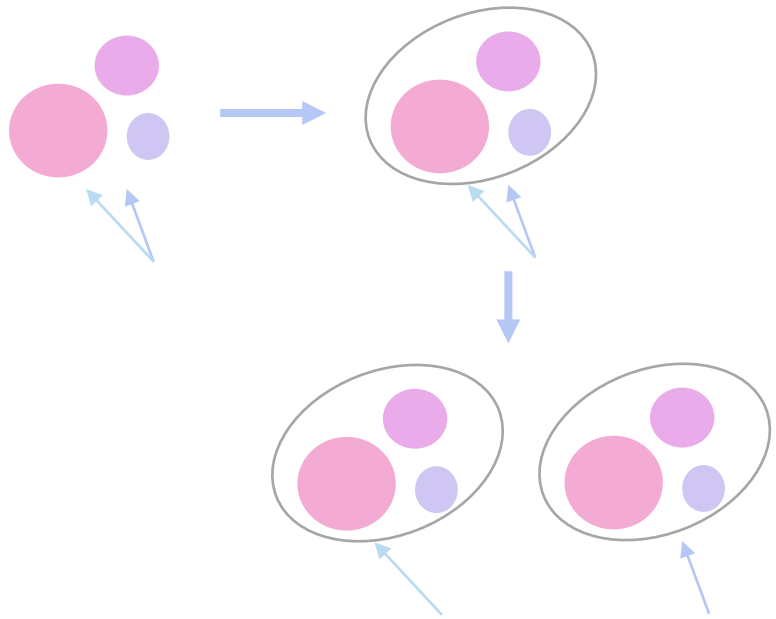
Backup | PFA0 w/ Diagram

- **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

- **The gist:**
  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



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

- **Track-Cluster Subtractor:** subtracts momentum of matched track(s) from cluster
  - In progress at EICrecon#1627 and in dev branch
  - Final changes before review waiting on EICrecon#1699
- **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_{trk}: tracking resolution to use in n-sigma cut
- σ_{cal}: calo resolution not use in n-sigma cut

Backup | PFA1(a): Track-Cluster Subtractor

- **Track-Cluster Subtractor:** subtracts momentum of matched track(s) from cluster
  - In progress at EICrecon#1627 and in dev branch
  - Final changes before review waiting on EICrecon#1699
- **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 $n\sigma$

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

   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
  - Still to-do!

- **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:
- $\{ID_{ecal}\}$: IDs of ECals to look for
- $\{ID_{hcal}\}$: IDs of HCals to look for
- $\{w_{em}\}$: weights of ECals to use (runs parallel to $\{Id_{ecal}\}$)
- $\{w_h\}$: weights of HCals to use (runs parallel to $\{Id_{hcal}\}$)

Backup | PFA2: Calo Remnant Combiner

- **Calo Remnant Combiner:** combines remnant clusters from subtractor into neutral particle candidates
  - Still to-do!

- **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

Note: maybe make inputs vectors of collections?

Parameters:
- Δr_{add}^{em}: window to add ECal clusters
- Δr_{add}^{h}: window to add HCal clusters
- $\{w_{em}\}$: weights (or weight) of ECals to use
- $\{w_{h}\}$: weights (or weight) of HCals to use

Backup | PFA3: Candidate-to-Particle Converter

- **Particle Converter:** takes candidate particles and turns them into reconstructed particles
  - Still to-do!

- **The algorithm:**
  1) Assign preliminary PID based on what info is available (e.g. no hcal clusters → electron, photon, or pi0)
  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
- Primary vertices (for neutral candidates)

Outputs:
- Reconstructed particles

Note: how to handle charged vs. neutral? Template algorithm? or split into two?

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
- σ_{cal}: calo resolution to use in energy calc

Backup | Associated tasks

Component	Parameter Tuning	Benchmarking
Track-Cluster Matcher	In progress	To-Do(?)
PFA0: Merge/Splitter	Some done*	To-Do*
PFA1(a): Subtractor	To-Do	To-Do
PFA1(b): Charged Regressor	To-Do	To-Do
PFA2: Remnant Combiner	To-Do	To-Do
PFA3: Neutral Regressor	To-Do	To-Do
CC Topocluster Maker	In progress	To-Do

- **Beyond development:** significant amount of work to be done tuning parameters/benchmarking implementations!
  - **Parameter tuning:** need to work out reasonable initial parameters for algorithms
    - ☞ Running (mostly) single particle sims
  - **Benchmarking:** need to assess performance of algorithms
    - ☞ Making ROOT macros which will get integrated into physics_benchmarks

* **Notes:**
- Some parameter tuning done for initial implementation
  - ☞ BIC still needs tuning, NHCal should be revisited
- Benchmarking can build off existing plugin used for testing