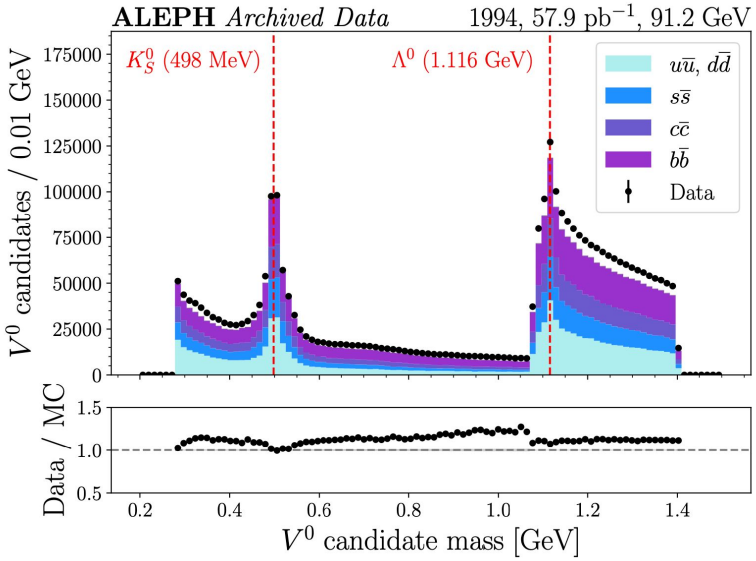
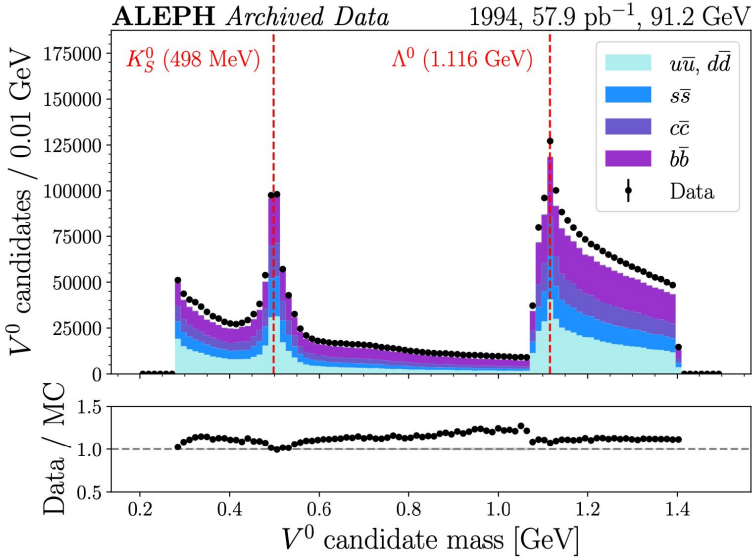
Context & overview

- Archived ALEPH data, converted to EDM4hep: [Documentation] [LEP data meetings]
- Used for jet flavour tagging: "*Modern jet flavour tagging in hadronic Z decays with archived ALEPH data*" [arXiv:2603.06524]
  - [ntuple production code] with FCCAnalyses
- Significant effort in optimizing the SV/V0 reconstruction, along the way we ran into bugs and missing functionality in FCCAnalyses, mainly in the vertexing code
  - Some fixes already in dedicated fork, move to upstream a.s.a.p



Context & overview

- Archived ALEPH data, converted to EDM4hep: [Documentation] [LEP data meetings]
- Used for jet flavour tagging: "*Modern jet flavour tagging in hadronic Z decays with archived ALEPH data*" [arXiv:2603.06524]
  - [ntuple production code] with FCCAnalyses
- Significant effort in optimizing the SV/V0 reconstruction, along the way we ran into bugs and missing functionality in FCCAnalyses, mainly in the vertexing code
  - Some fixes already in dedicated fork, move to upstream a.s.a.p



Today: Overview of the FCCAnalyses issues, three groups:

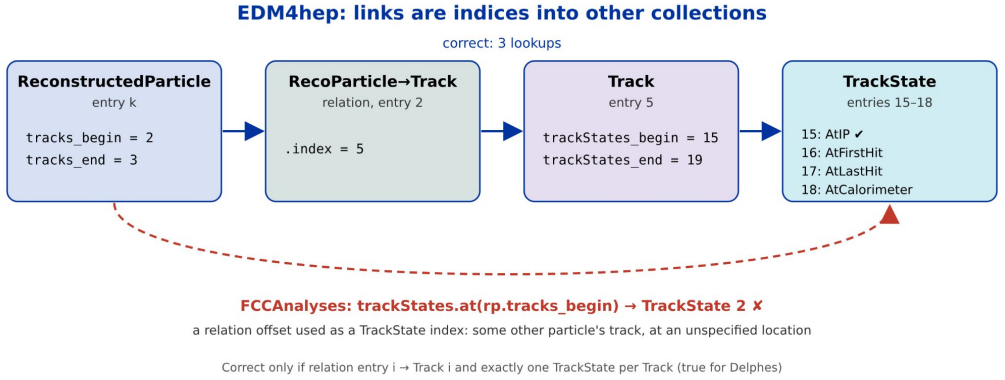
- Real bugs
- Hardcoded detector assumptions
- Missing configurability

Bug: Track association of reconstructed particles

- Bug in `ReconstructedParticle2Track`: the functions that get track parameters (d0, z0, φ, covariances, ...) for a reconstructed particle don't follow the EDM4hep link logic (`XPtoPar_dxy`, ~33 functions, also used by the `JetConstituentsUtils` pfcand helpers)
- In EDM4hep, links between collections are indices:
  - `tracks_begin/end` gives a range in the `RecoParticle→Track` relation
  - The relation's `.index` gives the Track
  - The Track's `trackStates_begin/end` gives its TrackStates (AtIP, AtFirstHit, ...)

analyzers/../ReconstructedParticle2Track.cc

```
64  ROOT::VecOps::RVec<float> XPtoPar_dxy(const ROOT::VecOps::RVec<edm4hep::ReconstructedParticleData>& in,
65                      const ROOT::VecOps::RVec<edm4hep::TrackState>& tracks,
66                      const TLorentzVector& V, // primary vertex
67                      const float& Bz) {
68
69    const double cSpeed = 2.99792458e8 * 1.0e-9;
70
71    ROOT::VecOps::RVec<float> out;
72
73    for (const auto & rp: in) {
74
75      if( rp.tracks_begin < tracks.size()) {
76
77        float D0_wrt0 = tracks.at(rp.tracks_begin).D0;
78        float Z0_wrt0 = tracks.at(rp.tracks_begin).Z0;
79        float phi0_wrt0 = tracks.at(rp.tracks_begin).phi;
```

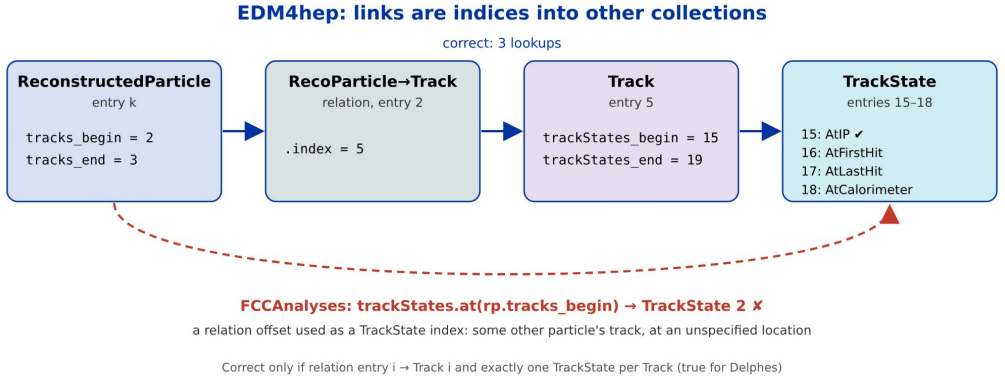


Bug: Track association of reconstructed particles

- Bug in `ReconstructedParticle2Track`: the functions that get track parameters (d0, z0, φ, covariances, ...) for a reconstructed particle don't follow the EDM4hep link logic (`XPtoPar_dxy`, ~33 functions, also used by the `JetConstituentsUtils` pfcand helpers)
- FCCAnalyses instead uses `trackStates.at(rp.tracks_begin)`. This is only correct if relation i points to track i and each track has exactly one state, as in Delphes
- Neutrals: the guard is `tracks_begin < size()` , not `begin != end`, so particles without tracks still read one

analyzers/../ReconstructedParticle2Track.cc

```
64  ROOT::VecOps::RVec<float> XPtoPar_dxy(const ROOT::VecOps::RVec<edm4hep::ReconstructedParticleData>& in,
65                    const ROOT::VecOps::RVec<edm4hep::TrackState>& tracks,
66                    const TLorentzVector& V, // primary vertex
67                    const float& Bz) {
68
69    const double cSpeed = 2.99792458e8 * 1.0e-9;
70
71    ROOT::VecOps::RVec<float> out;
72
73    for (const auto & rp: in) {
74
75      if( rp.tracks_begin < tracks.size()) {
76
77        float D0_wrt0 = tracks.at(rp.tracks_begin).D0;
78        float Z0_wrt0 = tracks.at(rp.tracks_begin).Z0;
79        float phi0_wrt0 = tracks.at(rp.tracks_begin).phi;
```



Bug: Track association of reconstructed particles

- Effect on ALEPH data and MC:
  - Charged candidates with their own track: 13%
  - Neutral candidates with a track: 99%
  - Affected: all pfcand track parameters and covariances, Sip2d/3d, JetDist.
- Workaround (ntuple-production PR #10, M. Defranchis): reorder `TrackStates` through `_RecoParticles_tracks.index` before calling

	before	after
own-track association	0.13	1.00
VDET d0 pull core (data)	3.03	1.42
neutrals with a track	0.99	0.00

- Proposal: fix this in FCCAnalyses, at the source
  - Resolve particle → relation → Track → TrackState, with an explicit state choice.
  - Check begin != end for neutrals.
- Any input where the relation isn't the identity, or with more than one state per track, is affected

Bug: Smaller bugs

- Beamspot units in `get_PrimaryTracks`
  - Beamspot position and width converted with 1e-6, but `VertexFitter_Tk` uses 1e-3
  - Beamspot constraint 1000× too tight in the primary-vertex track selection whenever `BeamSpotConstraint = true`
- Fragile validity check in `VertexSeed_best`
  - "Was a seed pair found?" decided by `chi2_min != 99`, an exact float comparison
  - Fix: initialise the pair indices to −1 and check `isel, jsel >= 0`, `addTrack_best` in the same file already does this
- No protection against failed fits in `get_PrimaryTracks`
  - A NaN vertex position or track χ^2 passes silently through the pruning loop
  - Fix: check for finite values, warn, and fall back to the unpruned tracks. Also warn if pruning removes all tracks.

Hardcoded: 2 T magnetic field

- Track momenta at the vertex come from `VertexMore` (Delphes `TrkUtil`), which assumes Bz = 2 T
  - Feed every SV/V0 invariant mass! At ALEPH (1.5 T), momenta are overestimated by 2/1.5 ≈ 1.33
- `VertexFinderLCFIPlus` has no B-field argument
- Fixes on the fork:
  - `VertexFitter_Tk`: new `solenoidBz` argument (default 2.0), with momenta scaled by `solenoidBz`/2
  - `solenoidBz` propagated through whole chain
  - Defaults are unchanged!
- Still hardcoded: `VertexingUtils::ParToP, XPtoPar, get_trackMom, ParToACTS, CovToACTS` (double fB = 2;)
- Proposal: pass Bz consistently upstream, plus a small Delphes PR for `VertexMore`

Delphes/external/TrackCovariance/VertexMore.cc

```
27  void VertexMore::Init(VertexFit* V)
28  {
29      //
30      // Set magnetic field and units
31      Double_t Bz = 2.;
32      SetB(Bz);
33      fa = -Bz * cSpeed();
34      if(fUnits)fa = -Bz * cSpeed()*1.0e-3;
```

analyzers/../VertexingUtils.cc

```
290  float get_trackMom(edm4hep::TrackState &atrack) {
291    double fB = 2; // 2 Tesla
292
293    float C = -0.5 * 1e3 * atrack.omega;
294    float phi0 = atrack.phi;
295    float ct = atrack.tanLambda;
296    //
297    float pt = fB * 0.2998 / TMath::Abs(2 * C);
298    TVector3 p(pt * TMath::Cos(phi0), pt * TMath::Sin(phi0), pt * ct);
299    float result = p.Mag();
300    return result;
301  }
```

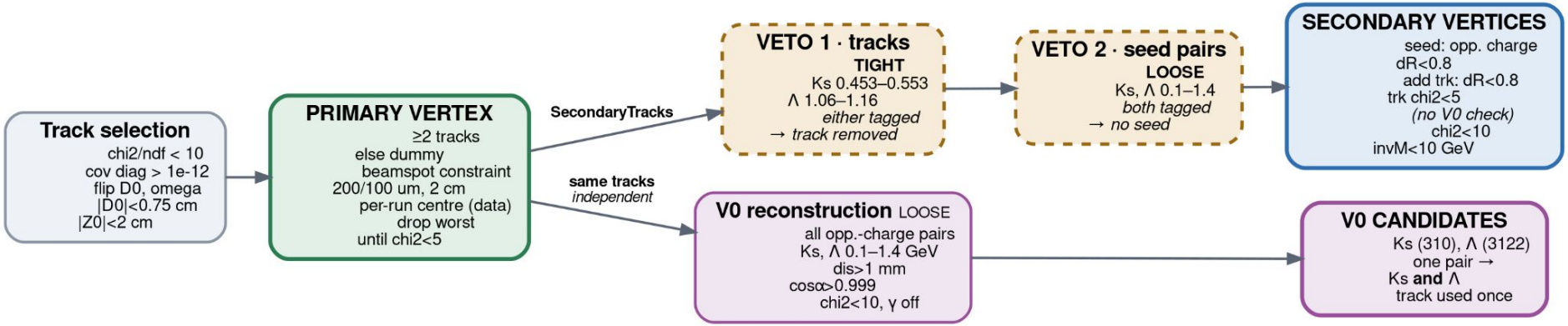
Missing configurability

- PV track pruning: `CHI2MAX = 25` in `get_PrimaryTracks` is hardcoded
  - Fix in fork: now an argument, propagated through chain, default 25
- SV finder: new options in `VertexFinderLCFIPlus`, all off by default:
  - `dR_prefilter_cut`: ΔR preselection of track pairs, for speed
  - `require_opp_charge`: opposite-charge seed pairs only
  - `tight_v0_seed`: tight instead of loose V0 veto at seeding, upstream it's loose at seeding but tight at track selection
  - Still hardcoded: the V0 veto mass windows and displacement cuts inside the SV finder (constraints_Ks/Lambda0/Gamma). These are tuned to FCC-ee resolution, we edited the source code for ALEPH, better to have a config?

Summary & proposals

- FCCAnalyses vertexing on ALEPH data exposed bugs that FCC Delphes MC never triggers
  - Major: hardcoded 2 T B-field; track association via tracks_begin, including neutrals.
  - Smaller: beamspot units, a fragile validity check, no handling of NaN fits.
- Fixes made on fork or overwriting core code, port back to upstream ! Proposed PR split:
  - Bugs (beamspot, validity check, NaN handling)
  - B field passed consistently
  - Track association through the relation
  - New options, with defaults unchanged
  - V0 logic, after discussion
- Collect everything in FCCAnalyses#515: one checklist item per PR above.
- For discussion: a vertexing configuration object instead of growing list of positional arguments, i.e. Bz, units, beamspot - could come from metadata? Algorithm-level: χ^2 cuts, V0 windows, prefilters, veto strictness Presets per detector/use-case ?

Discussion: V0 finder logic

- Current upstream behaviour (L576–626): Greedy, ie. each track pairs with the first passing partner, not the best χ^2, so the result depends on track order.
  - Fixed order: if (Ks) … else if (Λ) … else if (γ), so a K^0S candidate never tested as Λ or γ
  - Always exclusive: a used track can't enter another candidate.
- The same pattern is copied in 4 places (isV0, 2× get_V0s, get_V0s_jet)
- Fork (WIP): independent hypothesis checks, opt-in `exclusive_tracks` / `dR_pair_cut`.
  - Note: this changes the default output!
- Also new reconstruction chain from Matteo available in aleph-flavour-lab
- Questions:
  - Best χ^2 vs first match?
  - Overlapping hypotheses (one pair as both K^0S and Λ) allowed?
  - Exclusive vs inclusive candidates (e.g. a loose tier for ML)?
  - Patch the existing finder, or replace it with a new chain?

Summary of the tracks → PV → SV → V0 chain and its issues