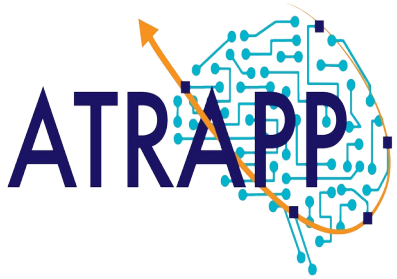


Tracking with Hashing in ACTS

Jeremy Couthures

M2 internship at LAPP, Annecy
Supervised by Jessica Levêque and Sabine Elles

The Hashing step

Track finding suffers from combinatorial issue:
Try to expand each seed with the remaining Space Points

⟶ Two speed up approaches:
- Reduce the number of seeds (linear)
- Reduce the number of remaining Space Points (combinatorial)

Introduction of the hashing step:
1. Groups space points into buckets
2. Seeding is done separately on each bucket to reduce the number of seeds fed to the TrackFinding Algorithm

- Method previously investigated by Sabrina Amrouche
- **My internship goal:** import the hashing step with the Annoy algorithm into ACTS and investigate performance with the full tracking chain

Evaluation of performance

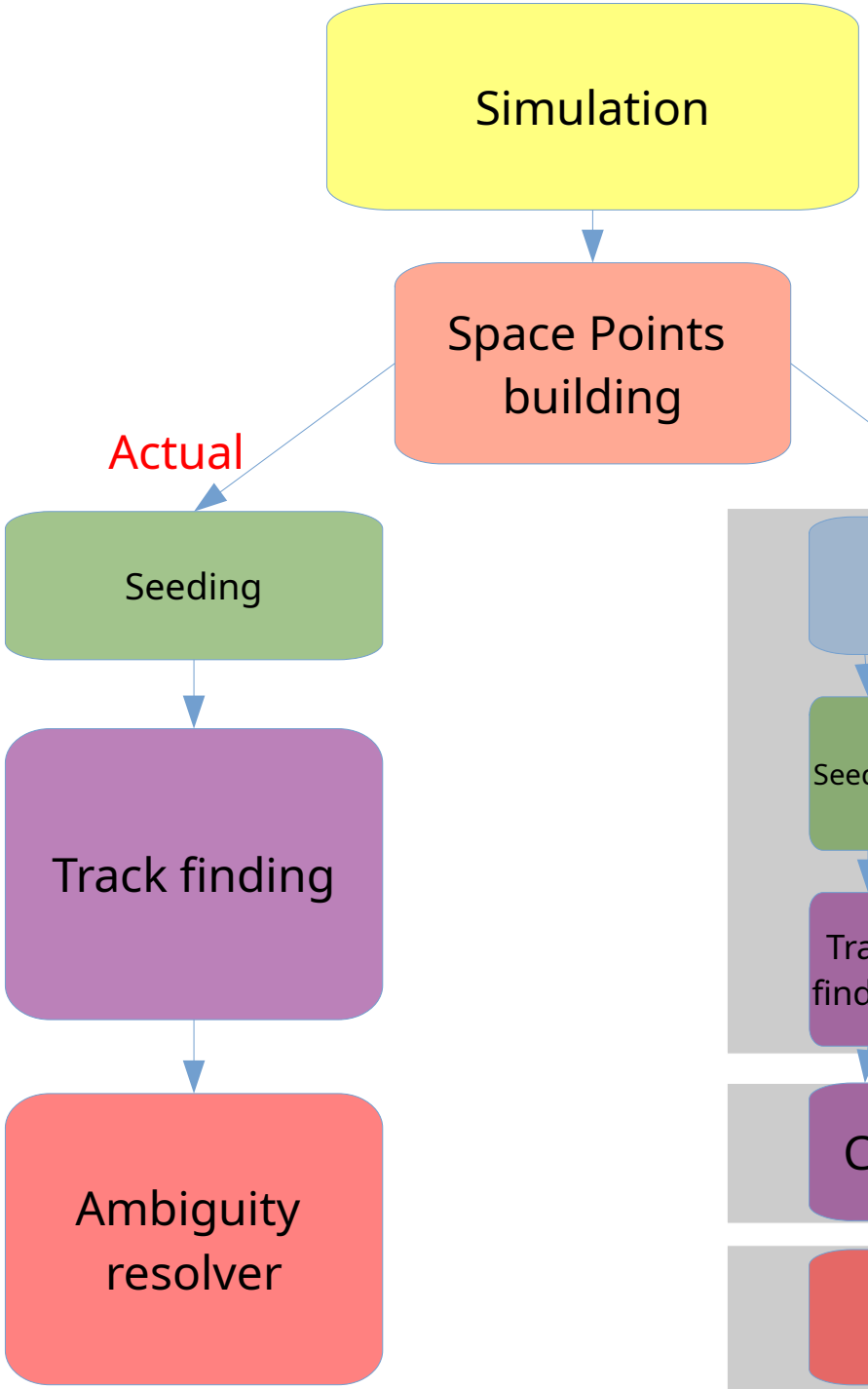
Physics: (CKF performances)

- efficiency = $$\frac{\#\ tracks\ matched\ to\ a\ truth\ particle}{\#\ reconstructible\ particles\ (>1GeV, >9\ hits)}$$
- fake rate = $$\frac{\#\ tracks\ not\ matched}{\#\ reconstructed\ tracks}$$
- duplication rate = $$\frac{\#\ reconstructible\ particles\ with > 1\ track\ match}{\#reconstructible\ particles\ (>1GeV, >9\ hits)}$$

Computing:

- Monitoring of CPU time (sequencer)

Approaches



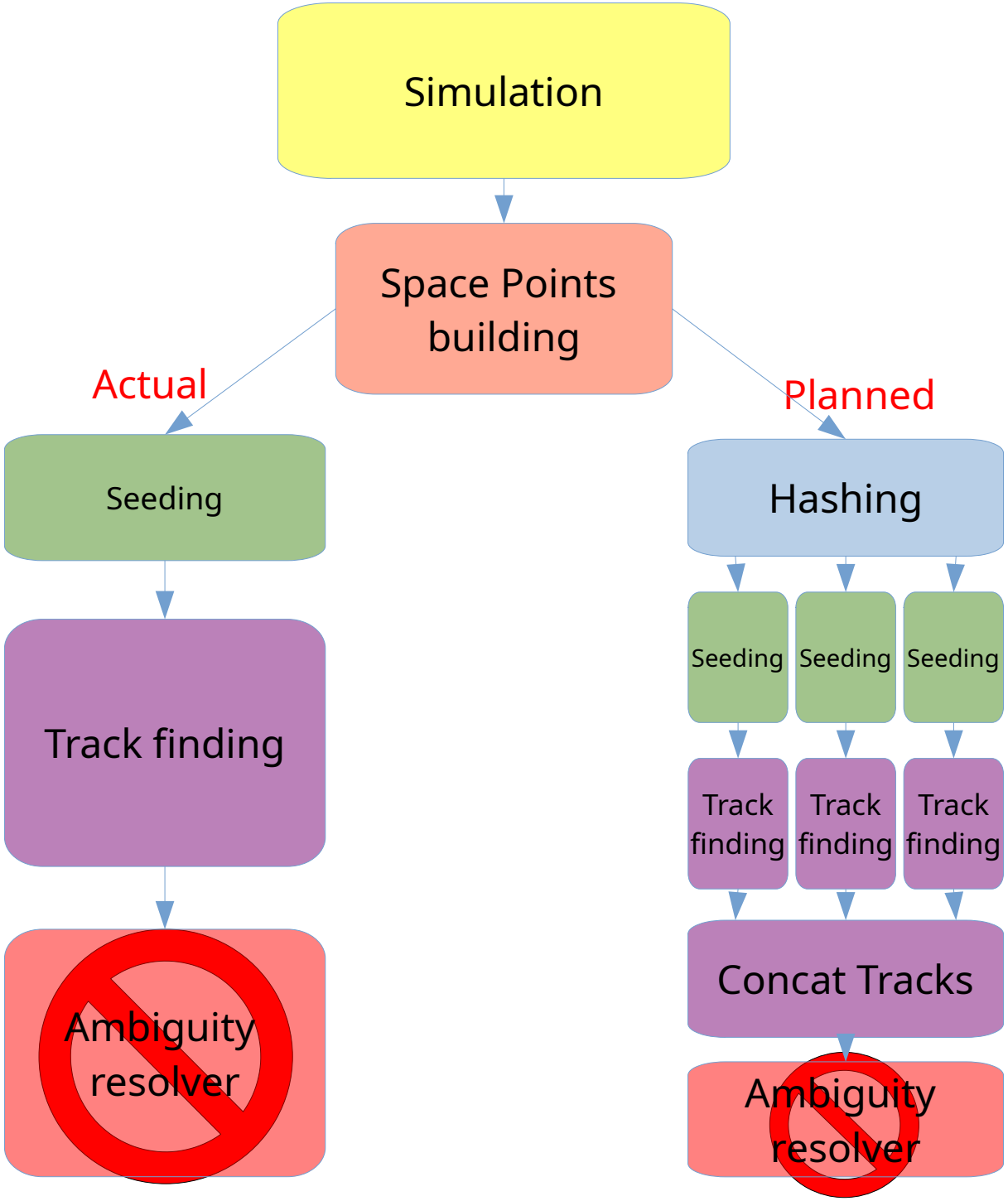
Full parallelization:

- Hashing creates small groups of Space Points (=buckets)
- Seeding and tracking are applied on each bucket

→ small number of seeds and possible expand combinations per bucket

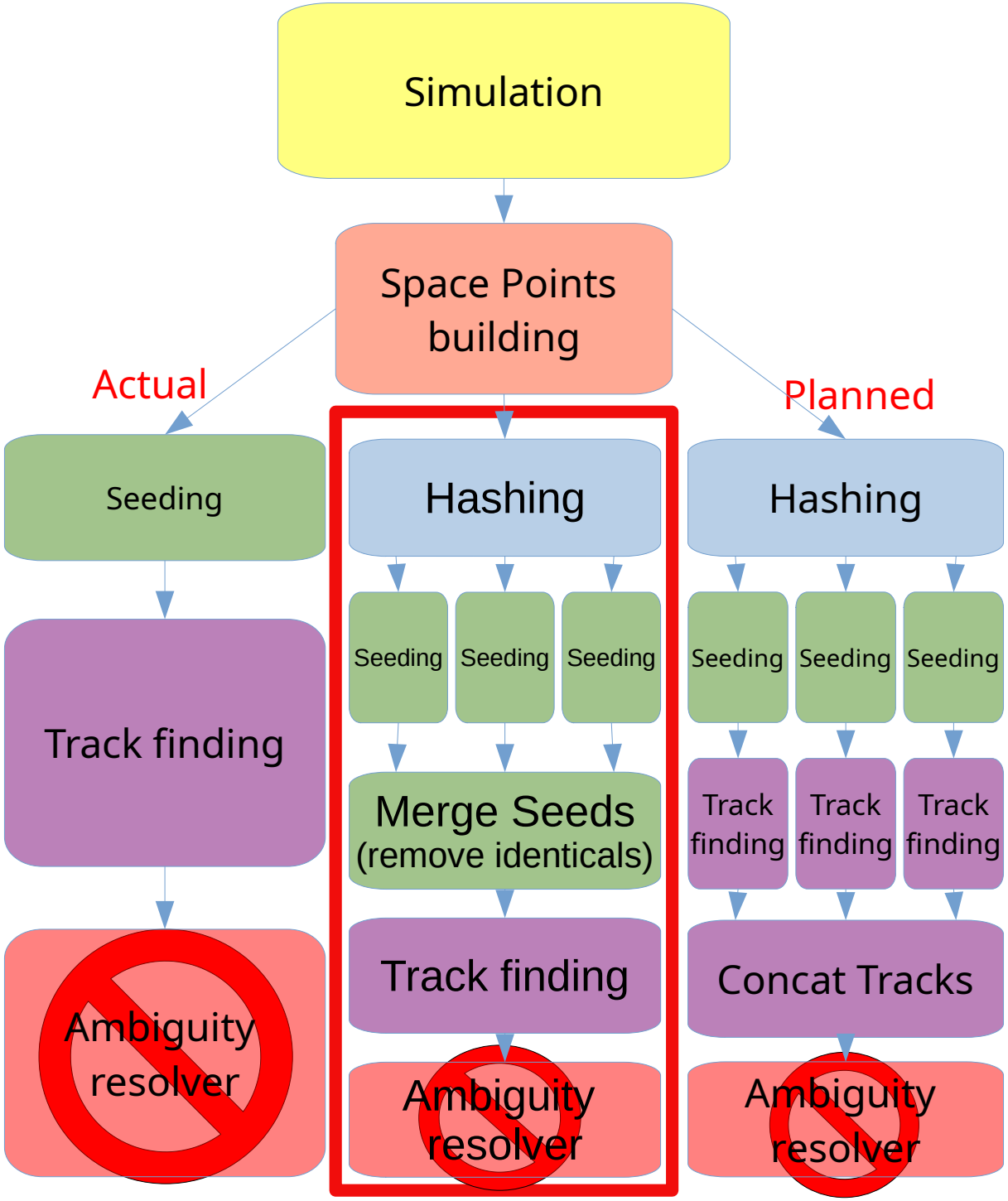
- Putting the tracks back together,
- And remove duplicates

Approaches



- **No Ambiguity resolver yet**
  → cannot remove duplicated tracks
  → cannot estimate the impact of duplication rate on total running time

Approaches



- **Seeding parallelization:**
  - Less seeds per bucket
  - Expected to loose time on merging

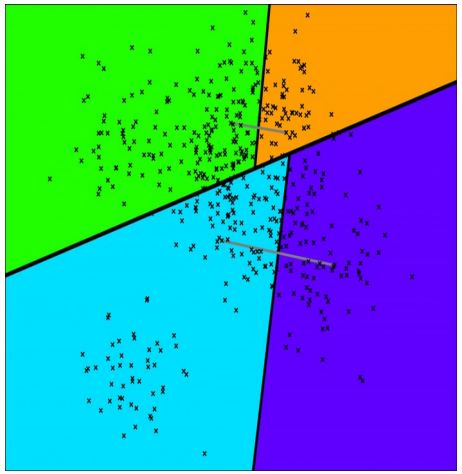
Hashing algorithm

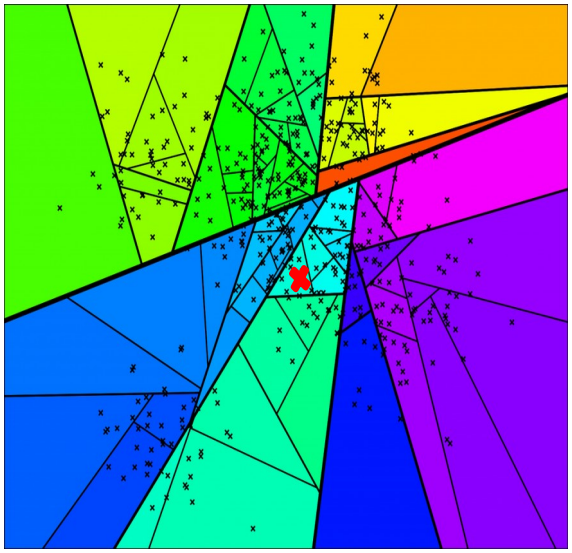
Approximate Nearest Neighbors Oh Yeah (Annoy)
k Nearest Neighbors Machine Learning Algorithm
→ need to define a distance, hence a metric relevant for our problem

- **Angular metric**: (see next slides)
- **Features**: x, y
- **Parameter:** number of neighbors in a bucket (set to 20 here)

Annoy training

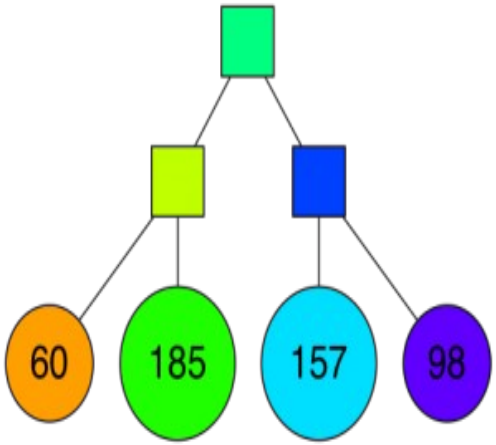
Space separation

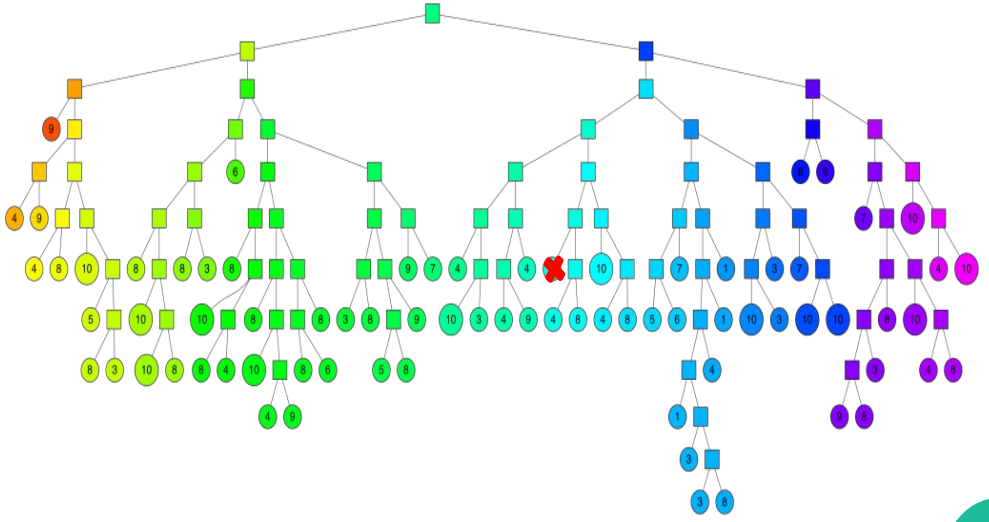




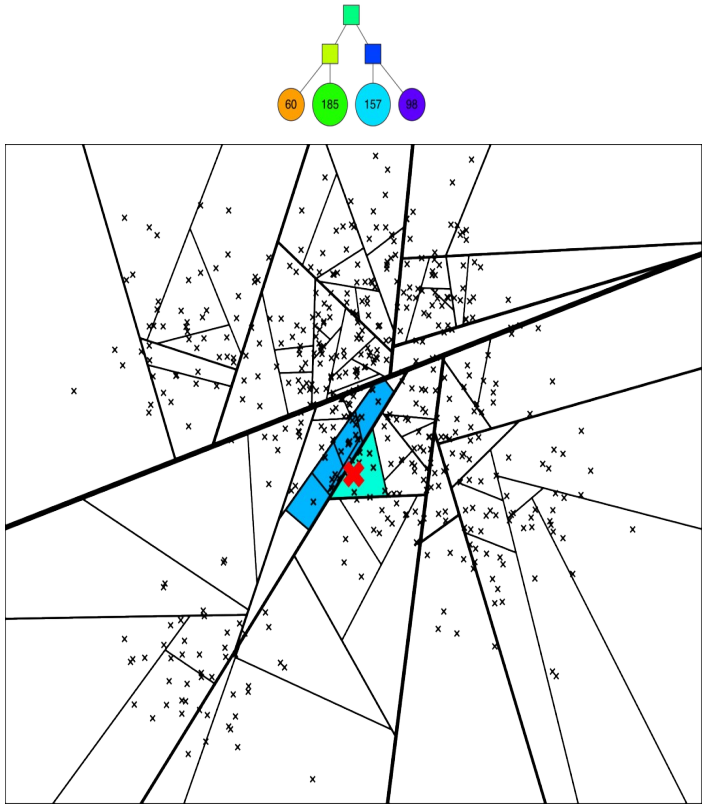
Takes two random points iteratively

Corresponding binary tree



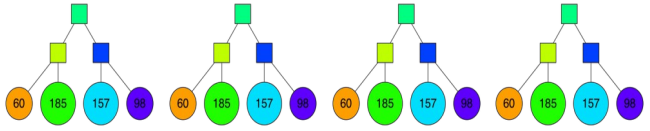


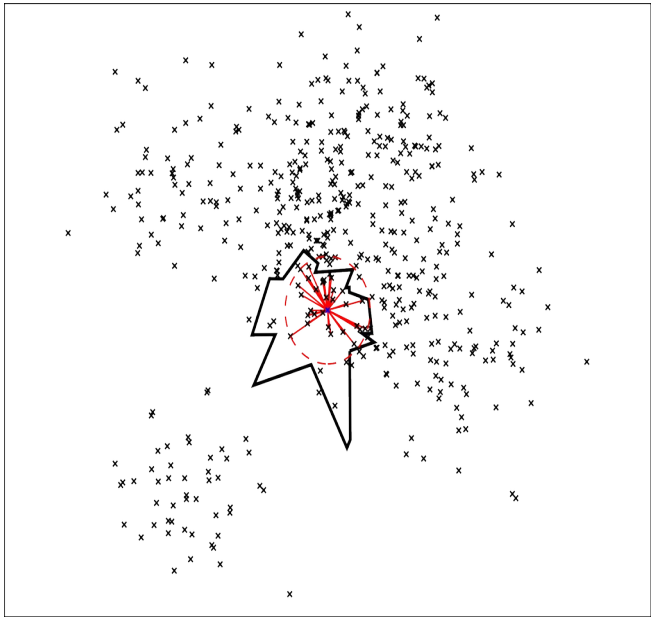
Annoy query



Merge neighbor subspaces

Approximation





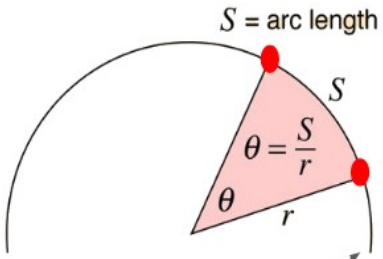
Union of trees' subspace

- Annoy tuning parameters: number of neighbors, number of trees, metric used, features used, number of subspace to look at

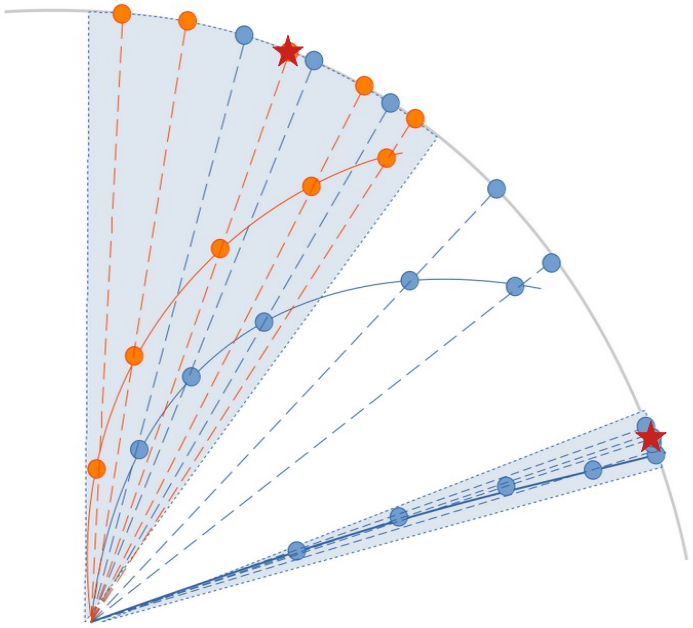
Metric definition

Metric used : <u>angular distance</u>

$\theta = S / R$



where S = distance travelled and R = radius of the circle

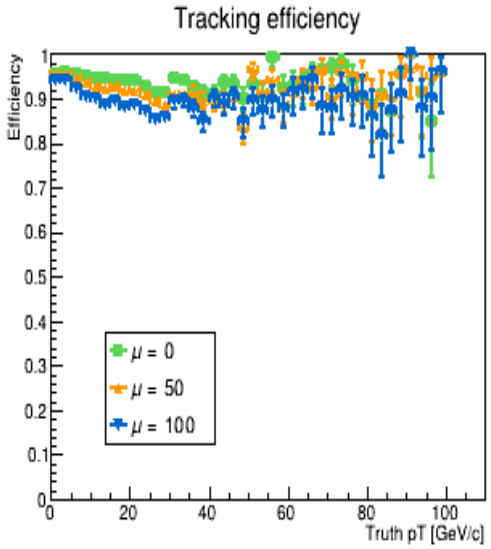


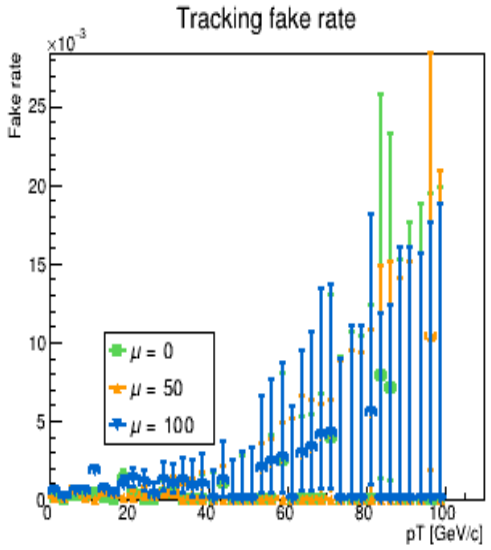
- Projection of a track on a circle
- Bucket: the nearest points on that circle

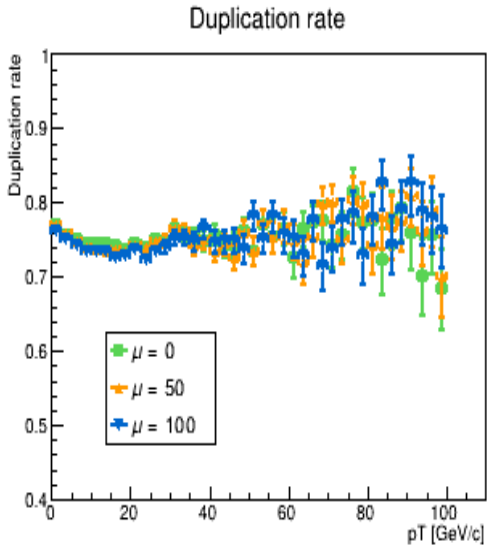
→ High pT track ~ linear track: all the hits are expected to fall in the same bucket

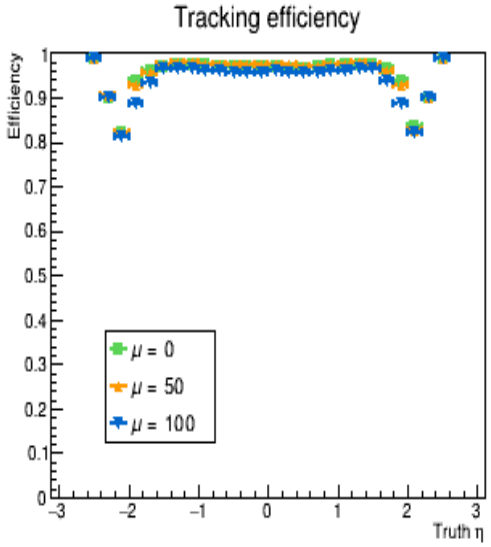
→ Low pT tracks at high μ:
Buckets may contain mixed hits from several tracks → Bad seeds?

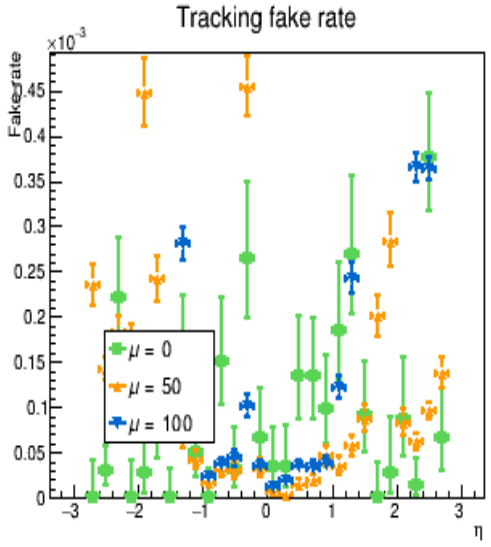
Physics performances (no hashing) Reference

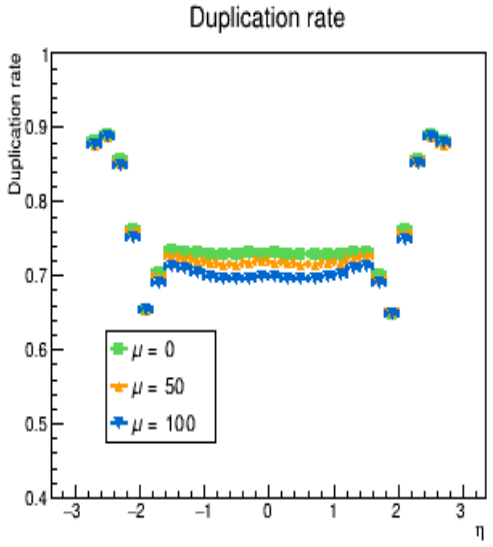








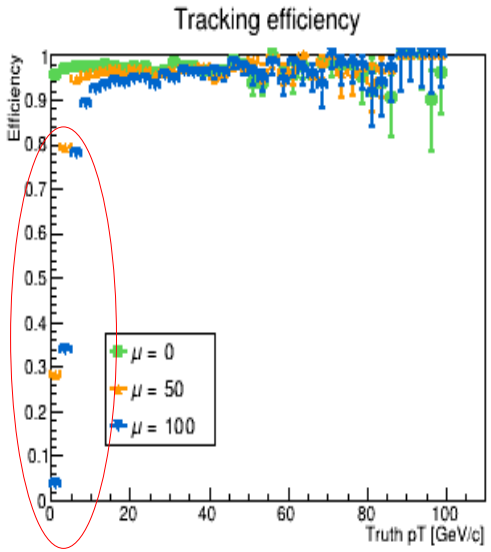


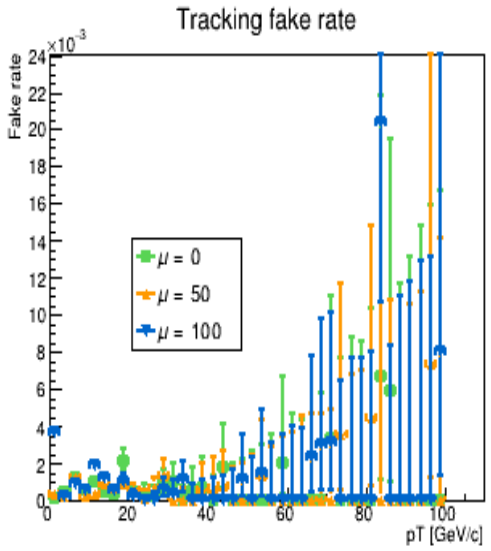


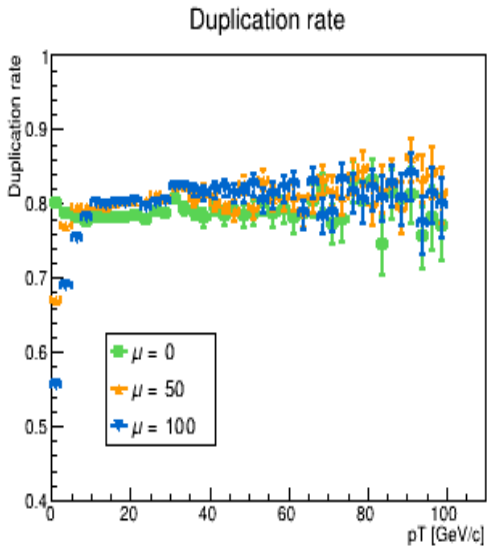
5000 $t\bar{t}$ events

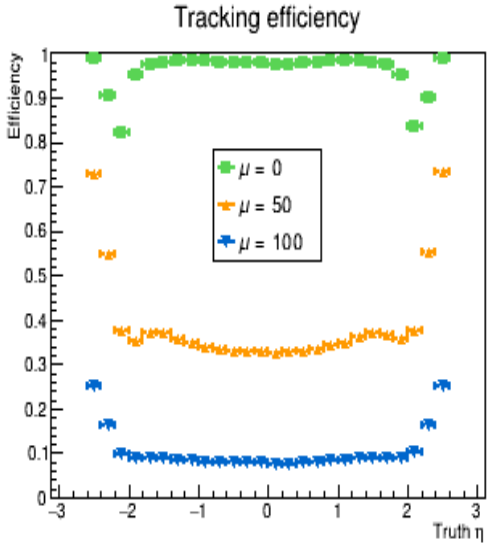
generic detector

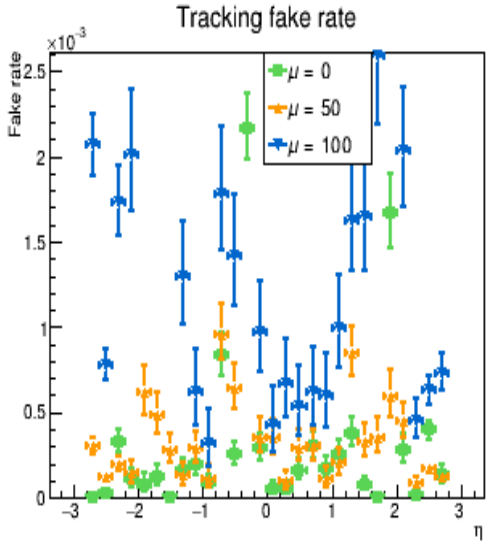
Physics performances with hashing

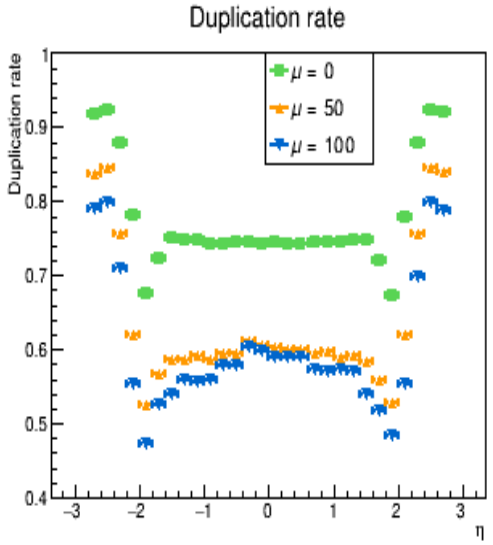






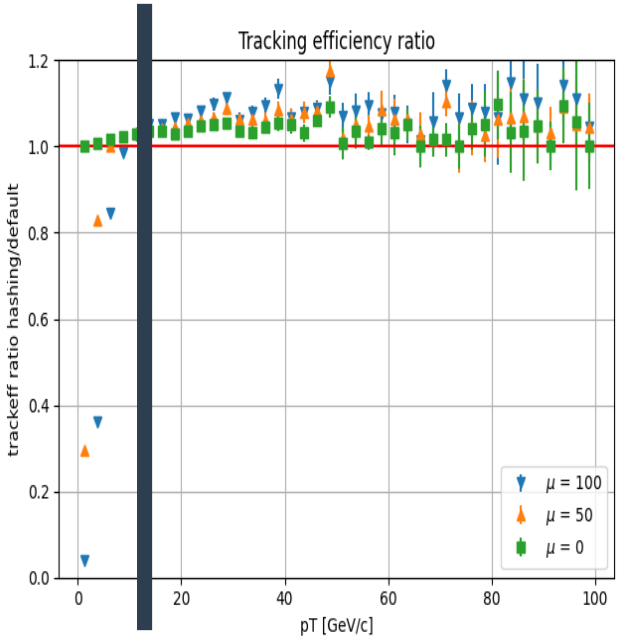


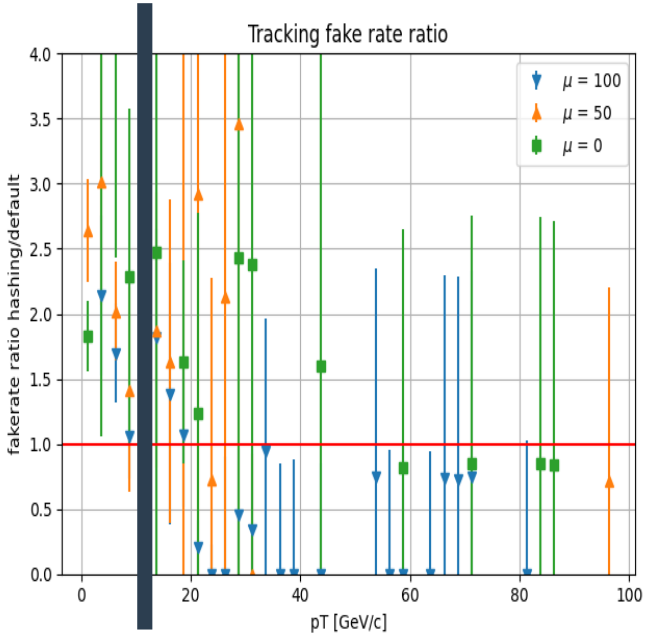


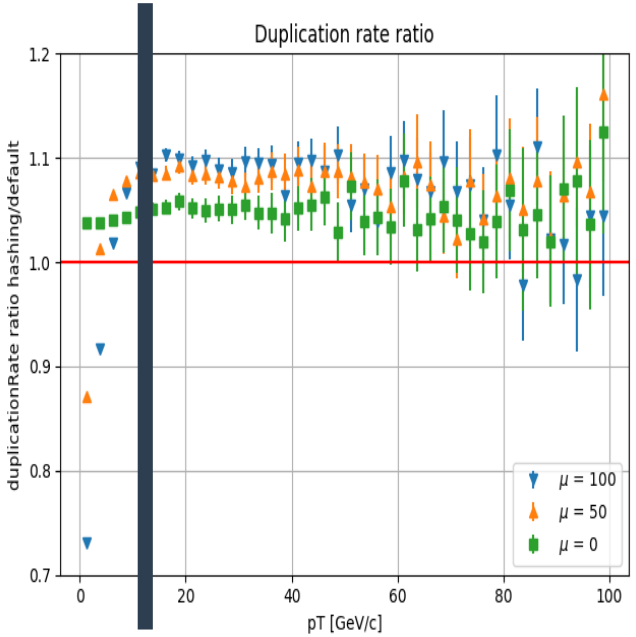


Bucket size = 20 (default)

Physics performance ratio







- Improved efficiency for high pT > 15 GeV
- Higher fake rate for low pT < 15 GeV at high μ
- Higher duplication rate for μ = 0

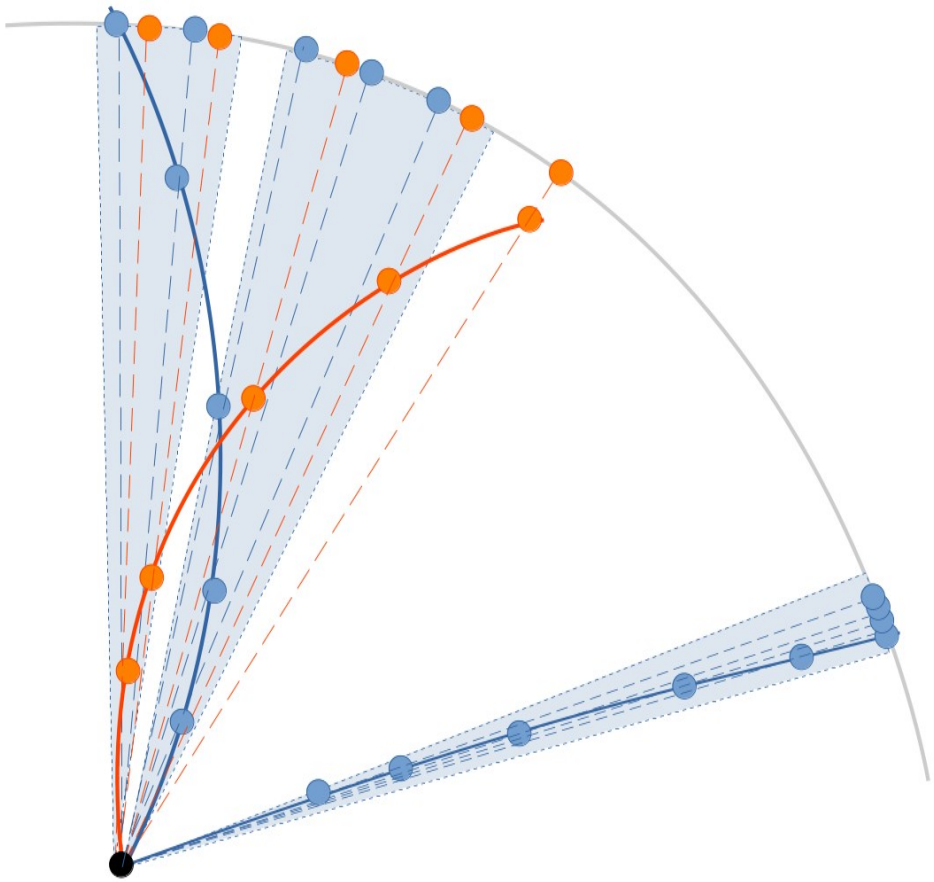
  → More seeds?
- Lower reconstruction efficiency at low pT with high μ

  → Bucket size? Metric?

More seeds with hashing even after removing identical ones → Why?

```
seedfinderConfigArg = SeedfinderConfigArg(
        r=(None, 200 * u.mm),  # rMin=default, 33mm
        deltaR=(1 * u.mm, 60 * u.mm),
        collisionRegion=(-250 * u.mm, 250 * u.mm),
        z=(-2000 * u.mm, 2000 * u.mm),
        maxSeedsPerSpM=1,
        sigmaScattering=50,
        radLengthPerSeed=0.1,
        minPt=500 * u.MeV,
        bFieldInZ=1.99724 * u.T,
        impactMax=3 * u.mm,
    )
```

Metric issue?



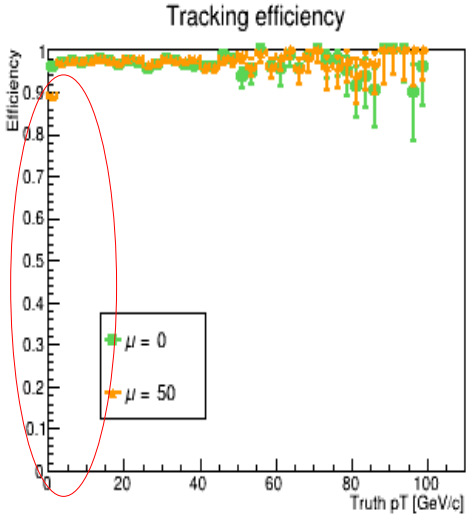
1. Metric not adapted to low pT:
Mixed hits of several tracks in the same bucket
→ Need to improve the metric

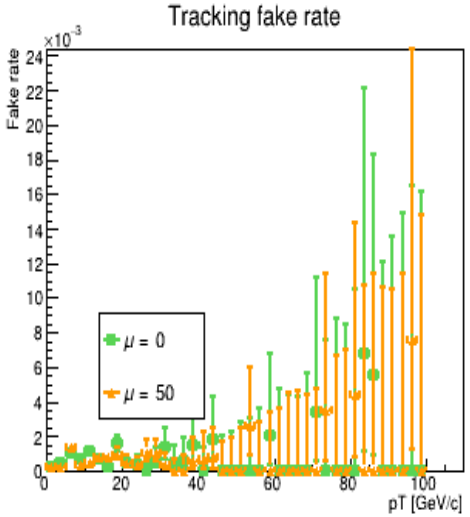
2. Possible improvement:
Increase bucket size to contain a full track even at low pT

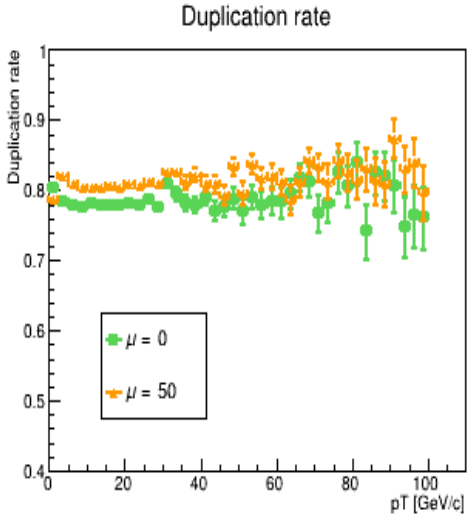
We tested..

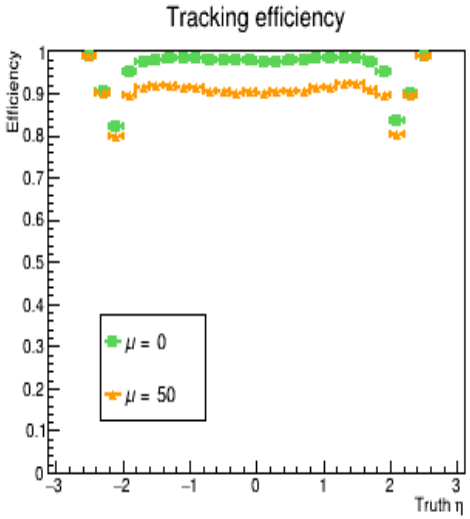
Impact of the bucket size

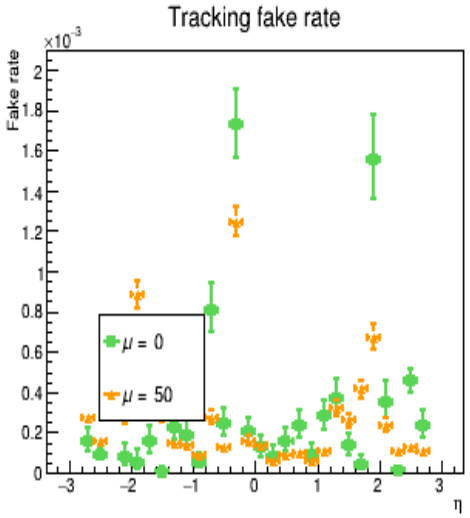
Bucket size = 50

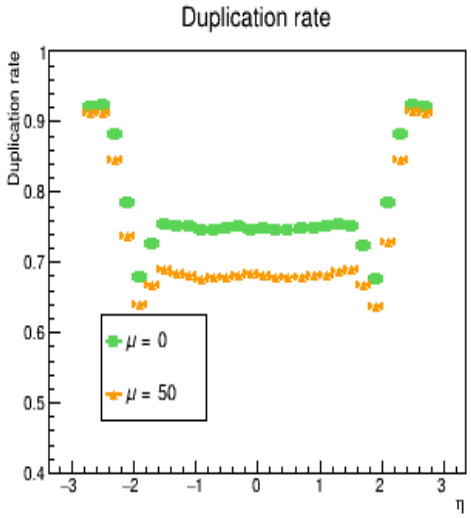




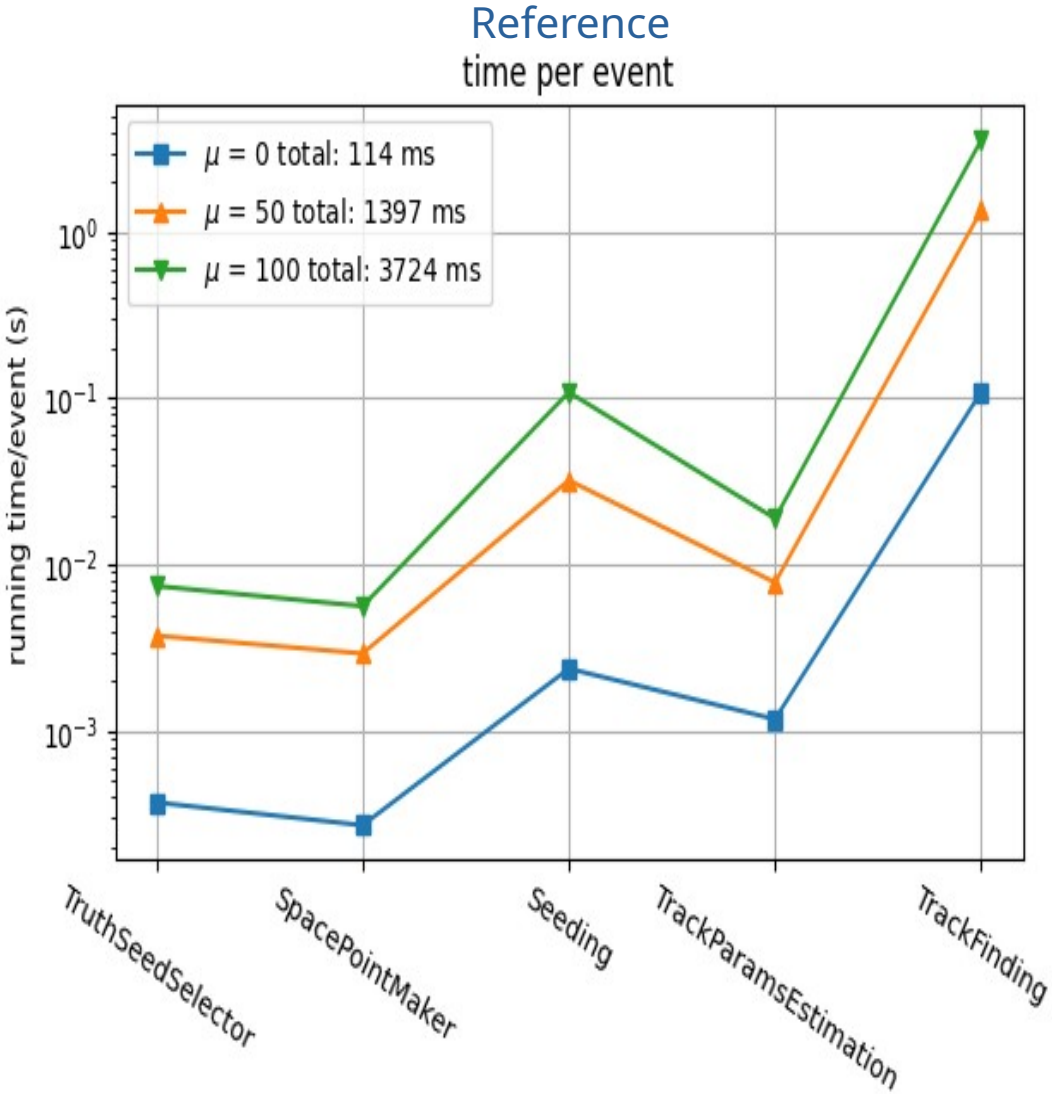








CPU time (no hashing)

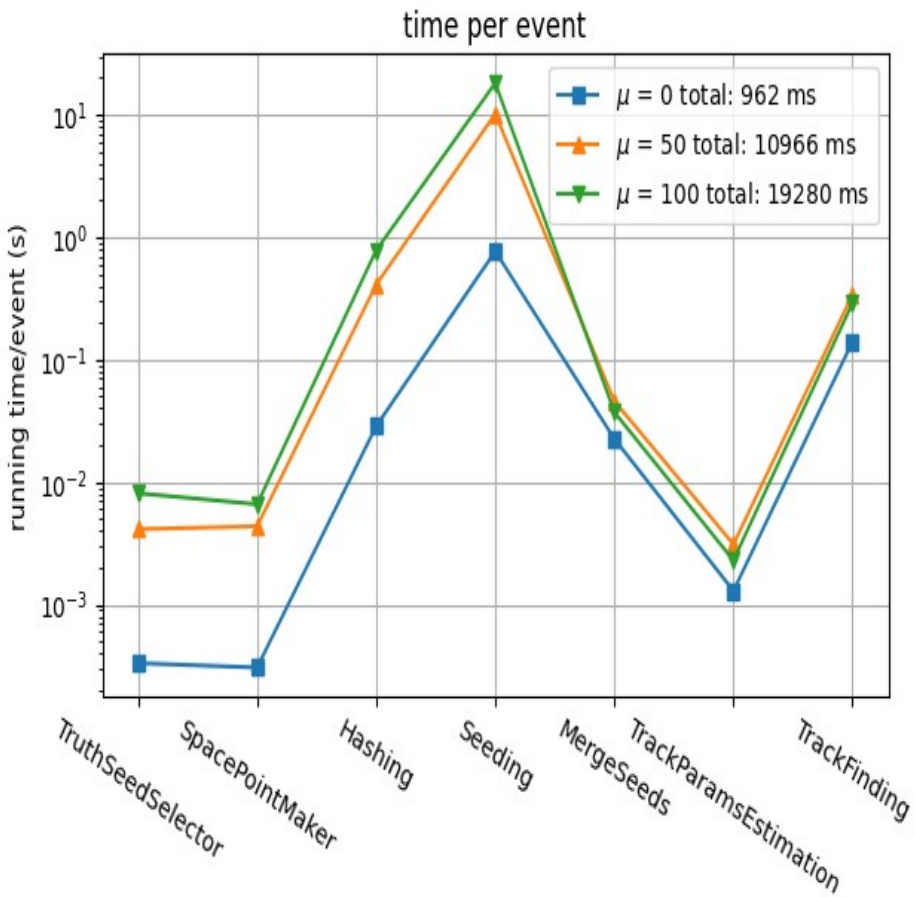


Complexity:

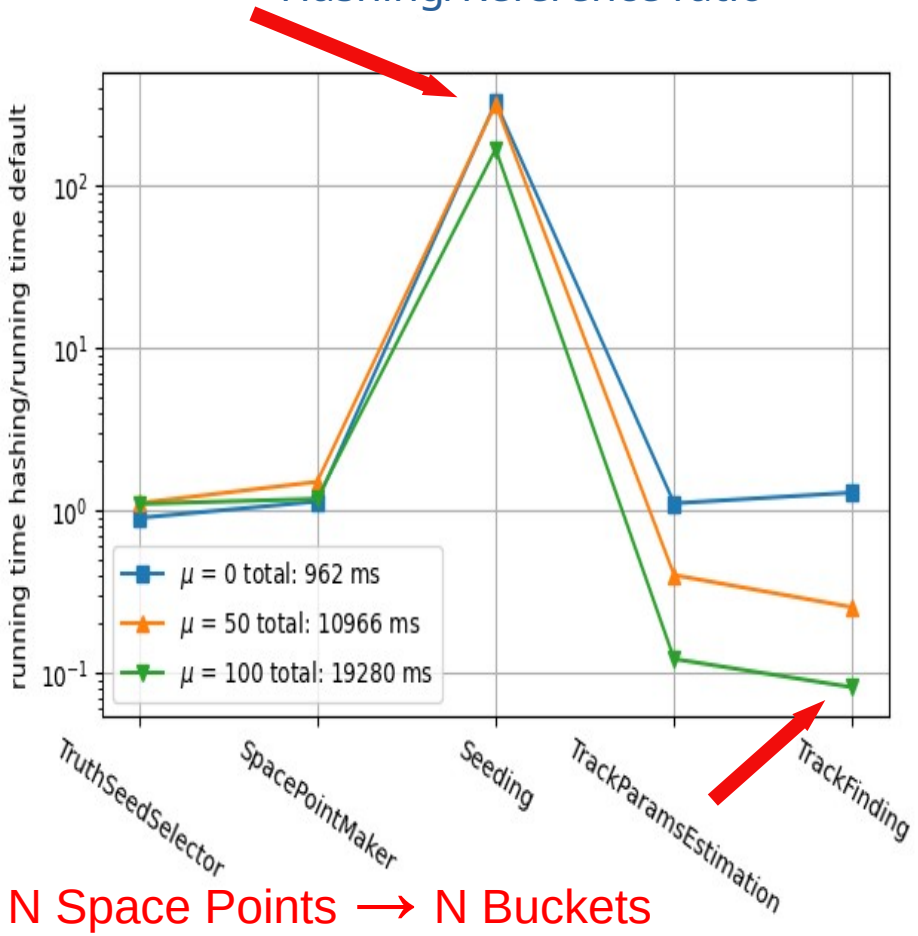
- Space Point Maker is linear
- Seeding: worst case $O(n^3)$
  observed ~$O(n^2)$
- Track Finding is combinatorial O(n!) and linear with the number of seeds

CPU time with hashing

Hashing



Hashing/Reference ratio



N Space Points ⟶ N Buckets

(we need something more clever...)

Mean seeding time per bucket (size = 20): 0.9 ms (constant with μ)

Summary

- Implementation of Annoy in ACTS
- We can compare performances with and without Annoy
- First observations with hashing:
  - Speed is dominated by the seeding
  - Physics performance get worst at low pT with increasing μ

Issues with the sequencer

- One sequencer ; each event has a different number of Space Points → number of buckets → number of seeding needed

  ⟶ have to find the maximal number of buckets among the events

- Was not able to run a sequencer until hashing and

  rerun the same after adding the remaining algorithm ⟶ had to run a copy

- Was not able to run the seeding in parallel

Outlooks

- Seeding parallelization in ACTS to be worked on
- Ambiguity resolver not implemented yet in ACTS
  - cannot compare duplicate rates
  - cannot do a full parallelization

- Need ML optimization (bucket size, new metric)
- Metric learning from data or simulated MC samples?
- Replace Annoy by a “novel” Neural Network architecture (Transformers?)

QUESTIONS?