Using Evolutionary Algorithms to Optimize Parameters for Track Reconstruction

By Peter Chatain

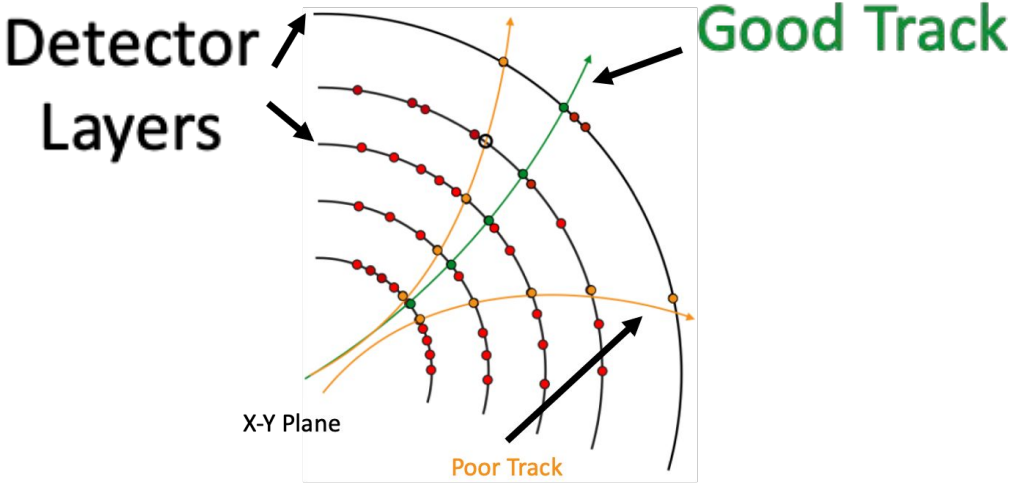
Mentors: Dr. Rocky Bala Garg & Dr. Lauren Tompkins

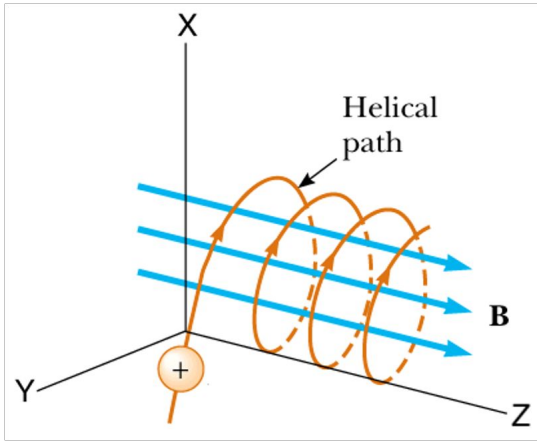
APS Meeting, Session T19: HEP Data Analysis In the Post-Moore Era

April 19th, 2021

What is Tracking?

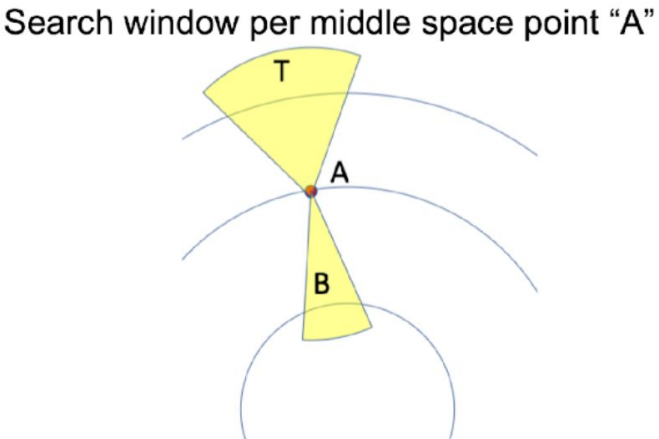
- Goal: Reconstruct the particle trajectories from detector hits
  - A “Hit” or “space point” = x,y,z coordinates of where a charged particle was registered on a detector layer





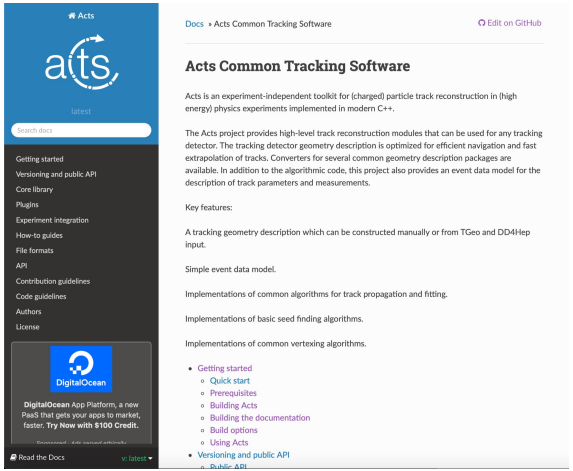
Track Seeding

- **Seeds are tracks with only 3 hits**
  - Reduce computational complexity and provide a starting point to build a full track
- Complexity is $O(n^3)$ since a seed is made for every possible combination of three points within search parameters.



ACTS - A Common Tracking Software

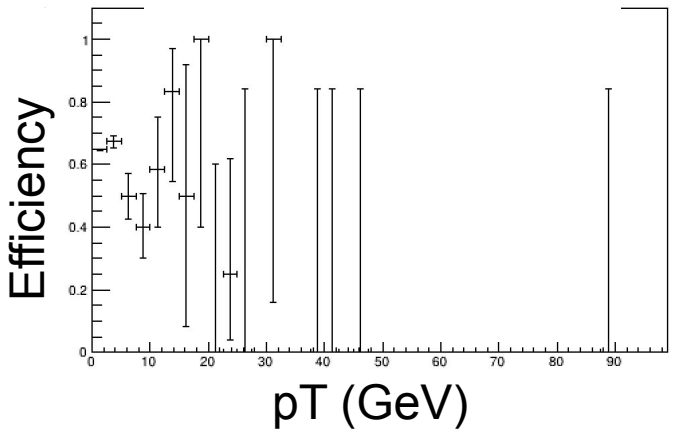
- Goals:
  - Prepare for an upgrade to the LHC where more data will need to be processed
  - Develop geometry agnostic software for tracking
- Originally ports over tracking software based on ATLAS
- Reference



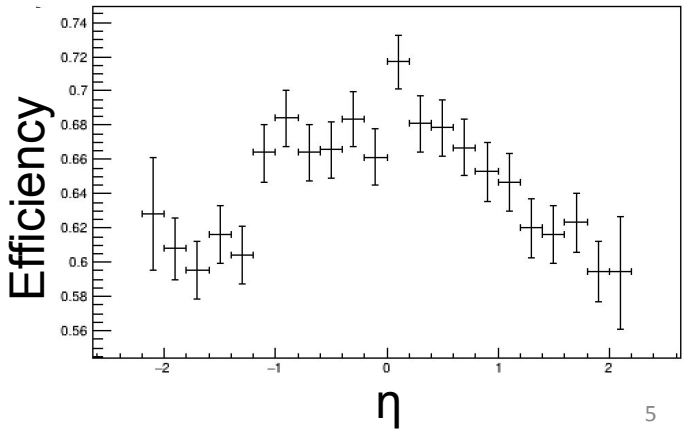
Background

- Summer project was to write the seed finding example for ACTS
- Out of the box ~50% efficiency on ttbar sample
  - Details in backup slides
- Configuration is heavily dependent on the detector geometry

Original Config on ttbar sample

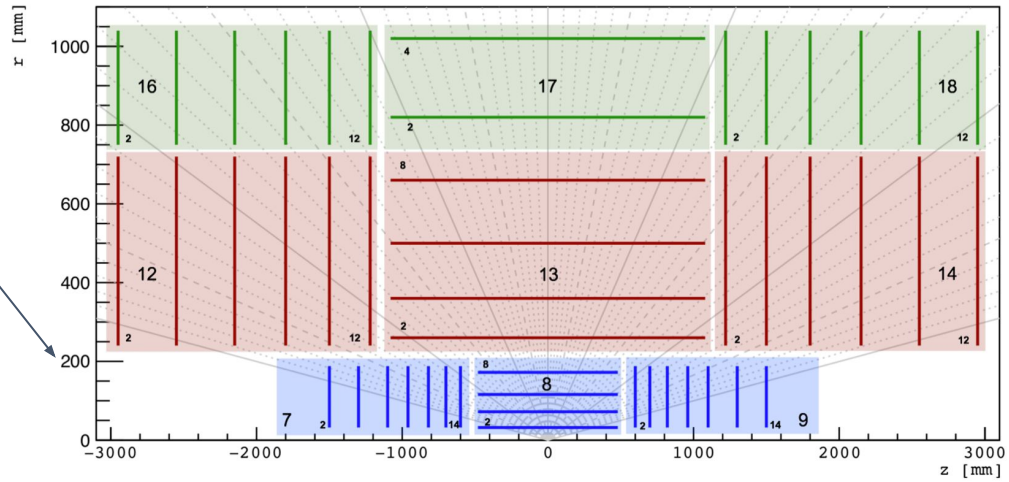


Original Config on ttbar sample



Seed Finding Configuration

- Certain parameters should be known by the user
  - Magnetic field strength
  - Maximum radius = 200
- Others are less obvious
  - Delta R Max
  - Sigma Scattering
  - MaxPtScattering
    - Units = ~4*MeV



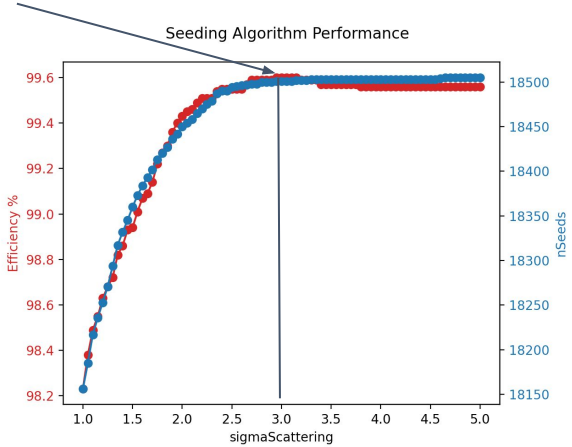
Hand Tuning

- Wrote a script using multi-processing to analyze which configuration to use
- Downsides:
  - Ad hoc. Is this really the best configuration?
  - Not generalizable to new detectors
  - Consumes physicists' valuable time

Red = Efficiency (fraction of particles identified by a seed)

Blue = Number of seeds generated

Eye-ball 3.0 for sigma-scattering



(How many standard deviations of scattering to include)

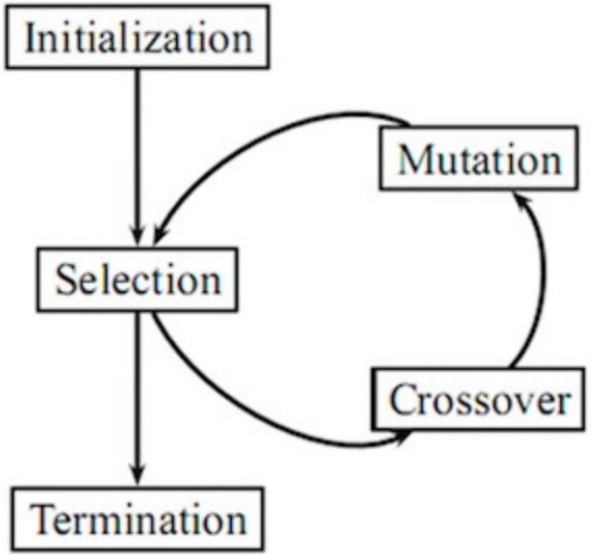
Hyperparameter Tuning

- Reminds me of hyperparameter tuning from Machine Learning!
  - Similar problem in ML. Hyperparameters are defined outside of training ”by hand.” e.g. learning rate.
- Common Hyperparameter Tuning Strategies
  - Grid search (brute force all combinations of parameters)
  - Random Search
  - Evolutionary algorithms
  - Derivative based approaches
- I decided on DEAP
  - Easily use multiprocessing
  - Non-linear search space

DISTRIBUTED EVOLUTIONARY ALGORITHMS IN PYTHON

Evolutionary Algorithm

- Step 1: Initialization
  - Create population
- Step 2: Selection
  - Reproduce better configurations, delete poor configurations
- Step 3: Mutation
  - Randomly mutate parts of each configuration.
- Step 4: Repeat steps 1-3 until Termination Criteria



Problem Statement

1. Efficiency = true particles matched to a seed / true particles
2. Fake Rate = seeds that don’t correspond to a particle / seeds
3. Duplicate Rate = seeds that re-identify a particle / seeds

- Given an initial guess at the best configuration, and any geometry, find the optimal configuration for the seed finder
- Unrealistically perfect scenario is 100% efficiency, 0% duplicate and 0% fake rate.

Results on ACTS Generic Detector

Dataset	Notes	Efficiency %	Fake %	Duplicate %
Generic Muon	Hand tuned	98.9	8	54
Generic Muon	-	99.4	6	70
Generic ttbar	Hand tuned	96.6	38	34
Generic ttbar	-	98.34	47	72.8

Best Configuration per generation

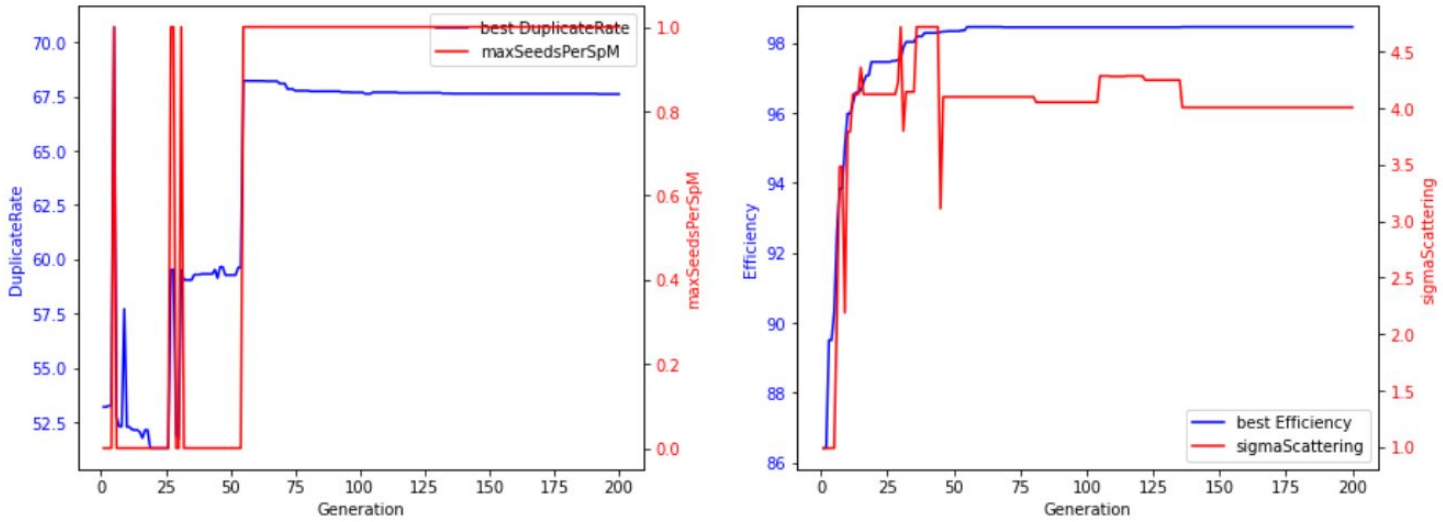
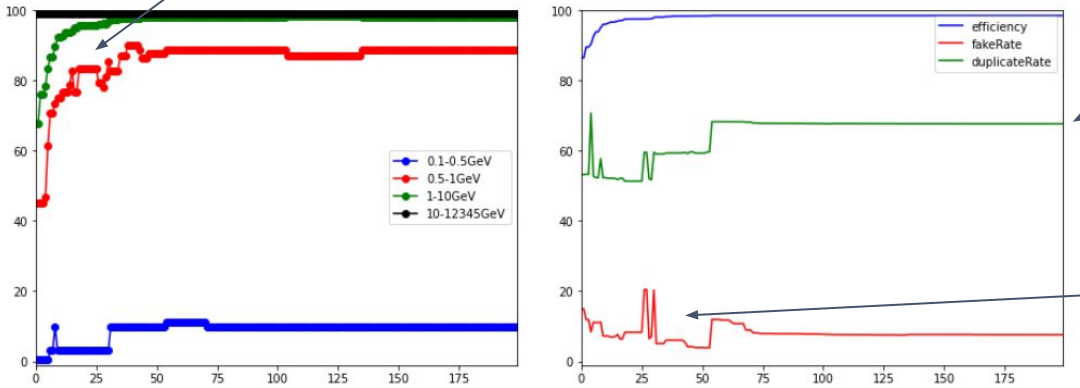


Figure 2. Best individual plotted over each generation

Summary Plots

Most efficiency gain was in pT range 0.5-10 GeV.



Duplicate rate kept minimal

Fake rate actually decreased

Figure 3. Efficiency for different pT ranges on muon sample (left). Overall scores (right)

Recap & Questions

1. Seeding algorithms are difficult to hand tune.
2. Evolutionary algorithms are a novel & robust way of optimizing seeding algorithms. Simply provide an evaluation function, and it will find the optimal balance.

- I would love to hear suggestions. My code is located here: https://github.com/Pchatain/seedingWithEA
- Any questions?

Thank you to my amazing mentors:

Dr. Rocky Bala Garg &

Dr. Lauren Tompkins

Individual Evaluation

- **Loss function is difficult to choose**

$$\mathtt{Score} = \mathtt{Efficiency\%} - \frac{\mathtt{Fake\%} \times \mathtt{Duplicate\%}}{K}$$

- K = 1000 worked well.

- Individual = a seedfinder config = a tuple of parameters,
  - e.g. --sf-maxPt 12000 --sf-impactMax 0.99 --sf-deltaRMin 1 --sf-sigmaScattering 2.25 --sf-deltaRMax 60 --sf-collisionRegionMin -300 --sf-collisionRegionMax 300 --sf-maxSeedsPerSpM 1
- Evaluated by running the seeding algorithm with that configuration

Parameter Updates - Mutation

x_j^(i) = parameter j in configuration (i)

s_j^(i) = variance of update on parameter j in configuration (i)

N(0, s_j^(i)) = Normal distribution with mean 0, and variance s_j^(i)

$$x_j^{(i)} := x_j^{(i)} + \epsilon \mid \epsilon \sim \mathcal{N}(0, s_j^{(i)})$$

Each parameter is given loose bounds above and below during mutation.

Original Performance of Seeding Algorithm

- Tried filtering out particles that don’t have 3 hits in the pixel detector
  - Only small improvement seen ~65% efficiency
  - Better performance required a new parameter as well as tuning