Seeding Optimization

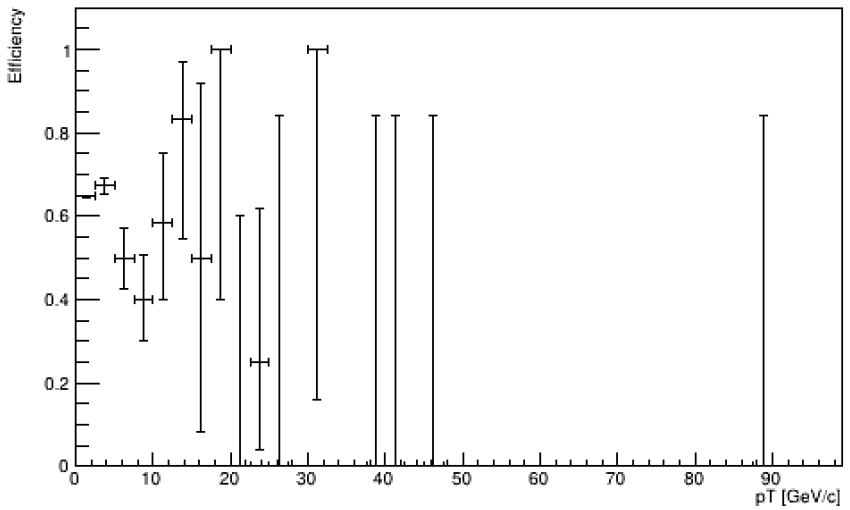
By Peter Chatain

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

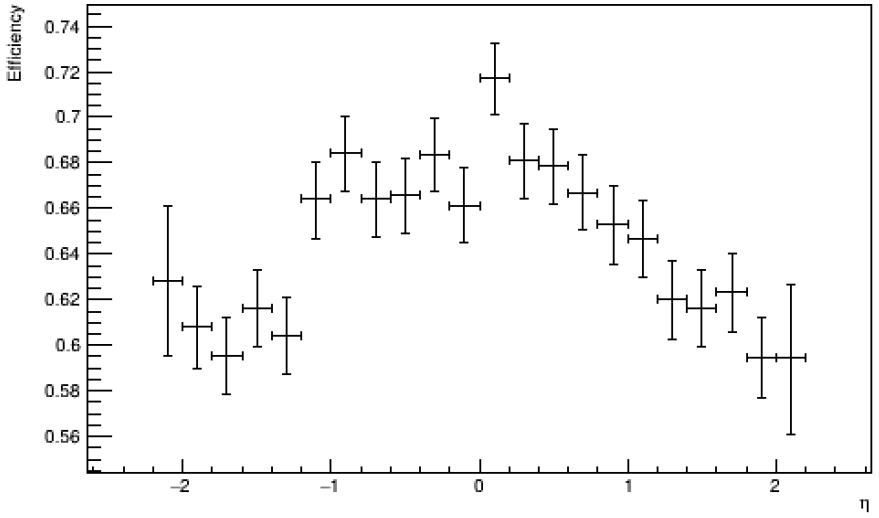
Background

- Summer project was to write the seed finding example for ACTS
- Originally found very poor performance ~50% efficiency on ttbar sample
  - 200 pileup, generic detector
  - Efficiency = fraction of true particles with a matched seed
- Tried filtering out particles that don't have 3 hits in the pixel detector
  - Only small improvement seen ~65% efficiency
- Tried to configure it by hand

Original Config on ttbar sample



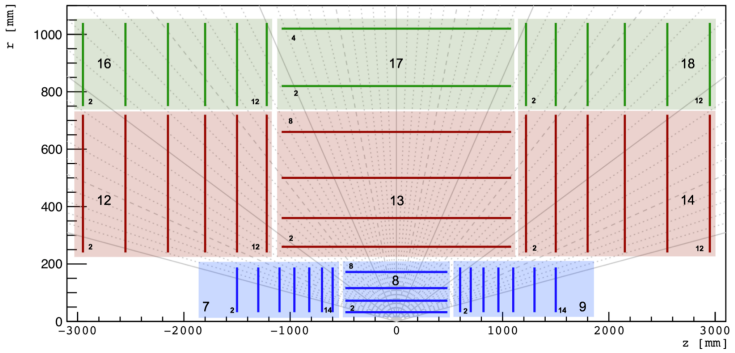
Original Config on ttbar sample



Hand Configuration

- Edited a seeding algorithm ”TODO” statement
  - Line 208 here
  - Maximum transverse momentum to apply sigma scattering cut on
- Certain parameters should be known by the user
  - Magnetic field strength
  - Where to look in the detector for hits
- Others are less obvious
  - Sigma Scattering
  - MaxPtScattering
    - Units = ~4*MeV
  - Impact max
- Configured by testing

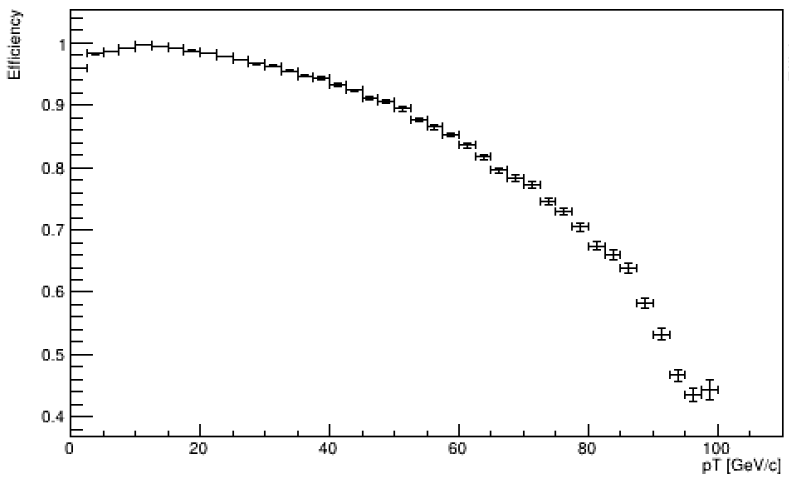
  one at a time



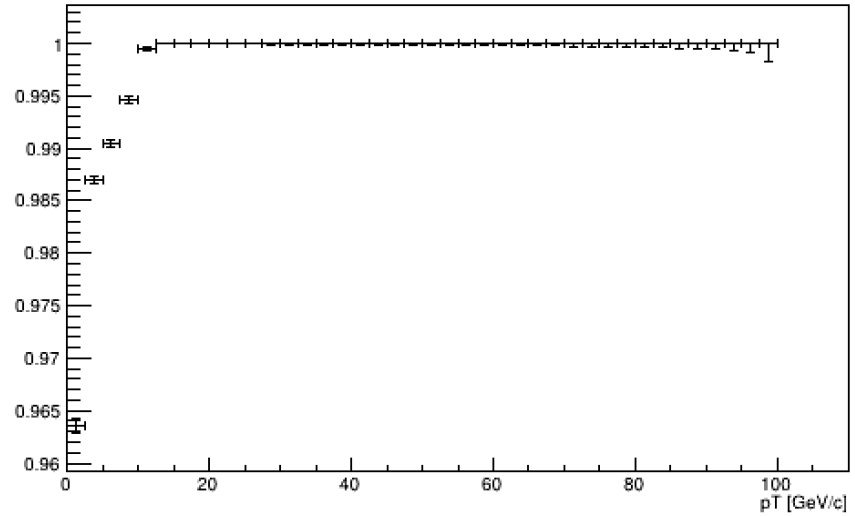
For these plots, particles must have:

- 3 hits in pixel “blue” volumes
- 1 hit in middle “red” volumes
- 1 hit in outer “green” volumes
- 0.1GeV <|pT| < 100GeV

Hand Tuned Config on single muon sample with 10,000 charged particles

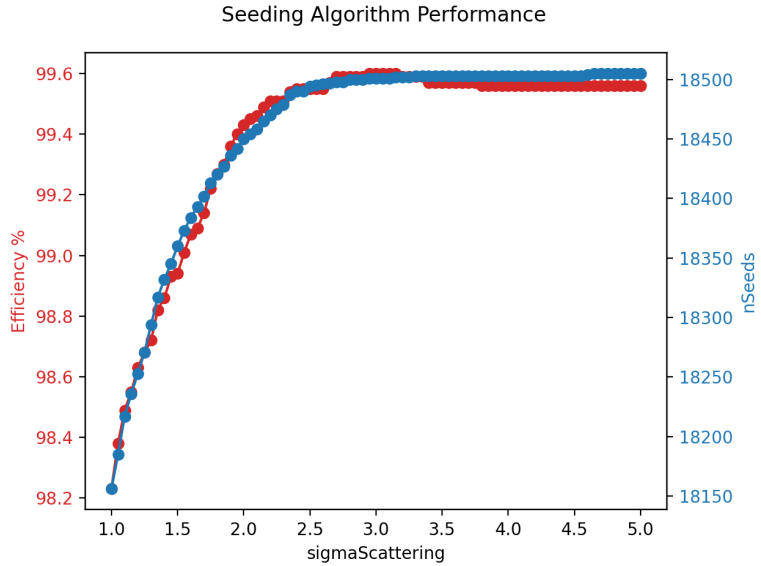


Removed sigma scattering cut entirely

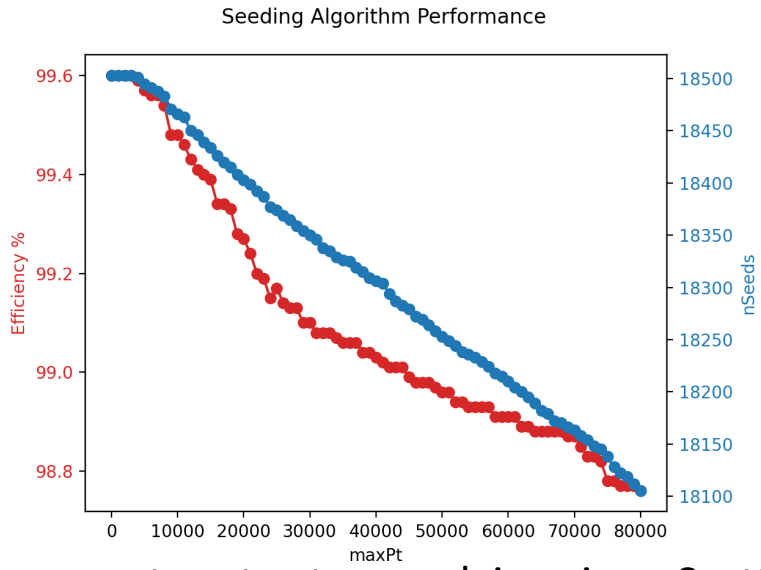


Hand Tuning

- Wrote a script using multi-processing to analyze which configuration to use
- I added boost command line options to read in parameters
- Downsides:
  - Parameters depend on each other so takes many iterations
  - Inefficient exploration of high dimensional space
  - Unclear whether configuration is optimal



(How many standard deviations of scattering to include)



(Which momentum to stop applying sigmaScattering cut)

Hyperparameter Tuning

- Reminds me of hyperparameter tunning from Machine Learning!
  - In ML, parameters are learned through training, and hyperparameters are defined outside of training ”by hand.” e.g. learning rate.
  - I’m taking a class in ML, so I was excited to try this
- Common Hyperparameter Tunning Strategies
  - Grid search (brute force all combinations of parameters)
  - Random Search
  - Evolutionary algorithms
  - Derivative based approaches
- I decided on DEAP
  - Easily use multiprocessing

DISTRIBUTED EVOLUTIONARY ALGORITHMS IN PYTHON

Problem Statement

- Given an initial guess at the best configuration, and any geometry (including ITK), find the optimal configuration for the seed finder
  - Should find > 99.4% efficiency on single muon gun sample. Ideally a < 10% fake rate and < 60% duplicate rate
    - This is what the seedfinder was able to obtain on generic detector with single muon sample and 200 pileup as generated here

  Efficiency = true particles matched to a seed / true particles

  Fake Rate = seeds that don't correspond to a particle / seeds

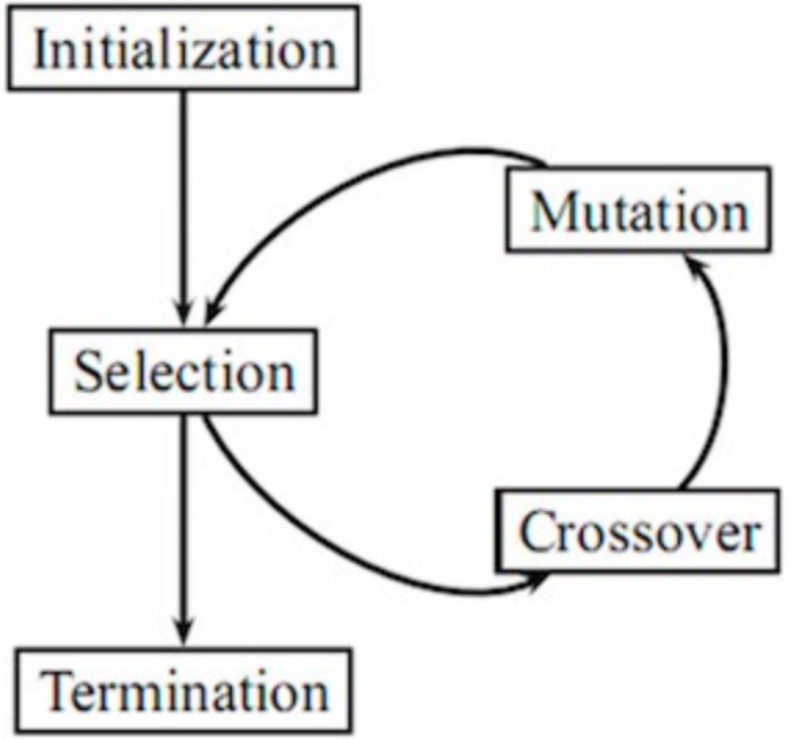
  Duplicate Rate = seeds that re-identify a particle / seeds

- First approach: Evolutionary Algorithm

Evolutionary Algorithm

DISTRIBUTED EVOLUTIONARY ALGORITHMS IN PYTHON

- Initialization
  - Provide a good guess, create N copies of it
  - Individual = one seedfinder configuration
- Selection
  - Evaluate the population
  - randomly delete poor performing individuals
  - replicate good performing individuals to keep pop size constant
- Mutation
  - Each individual has a 0.3 chance of being mutated
  - If mutated, each value in an individual has a 0.2 chance of being mutated
  - Mutation is drawn from gaussian distribution centered at 0
  - Numbers hand chosen before running the algorithm
- Termination
  - Ether max gen reached or ideal (> 99.4% efficiency, < 10% fake rate, < 60% duplicate rate)



Individual Evaluation

DISTRIBUTED EVOLUTIONARY ALGORITHMS IN PYTHON

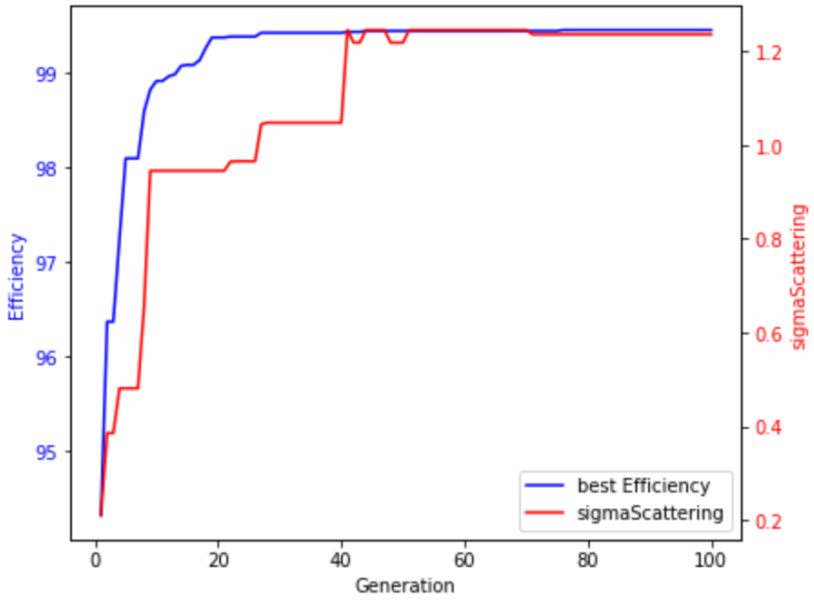
- Individual = a seedfinder config = a tuple of parameters,
  - e.g. --sf-maxPt 12000 --sf-impactMax 0.99 --sf-deltaRMin 1 --sf-sigmaScattering 2.25 --sf-deltaRMax 60 --sf-collisionRegionMin -300 --sf-collisionRegionMax 300 --sf-maxSeedsPerSpM 1
- Evaluated by running the seeding algorithm with that configuration
- Loss function is difficult to choose
  - I chose to score by efficiency, fake rate, and duplicate rate with priority in that order (i.e. 99.3% efficiency > 99.2% efficiency regardless of fake rate)
  - Other options are to combine terms
- Each parameter is bounded above and below during mutation

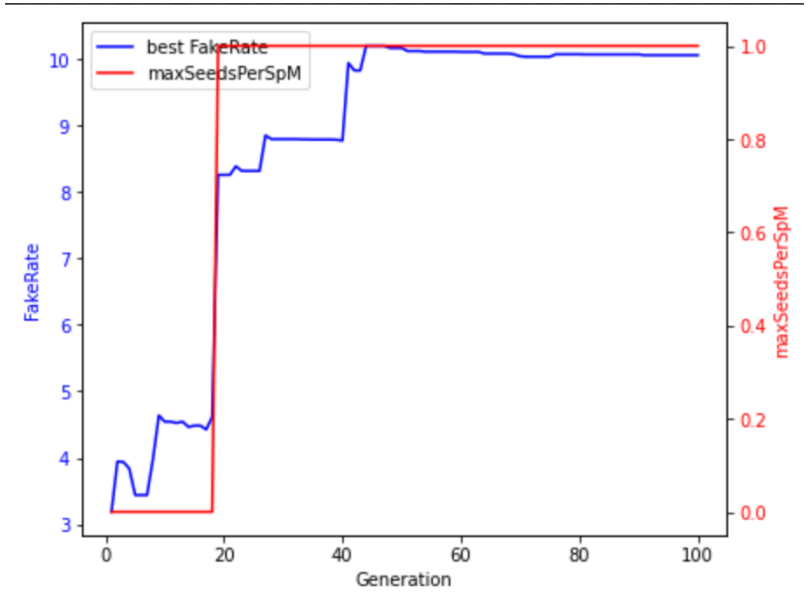
Evolutionary Algorithm Test 1

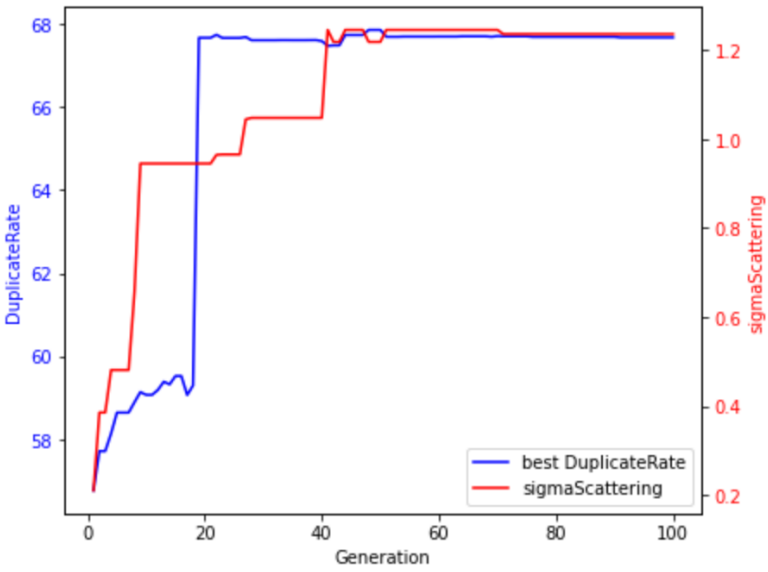
- Wanted to test whether the algorithm can learn at least one parameter
  - Edited sigma scattering from 2.0 to 0.2.
    - 99.1% efficiency changed to 93%
  - Goal: recover 2.0 sigma scattering
- Algorithm Set Up
  - Individual had 8 parameters
    - Including my maxPt cut on sigma scattering calculation
  - Population size 50
    - Around 7-15 mutated per generation (16 cores)
  - 100 generations (~15 minutes)
  - Trained on single muon gun sample with 10,000 particles

Performance Measured on Best Individual

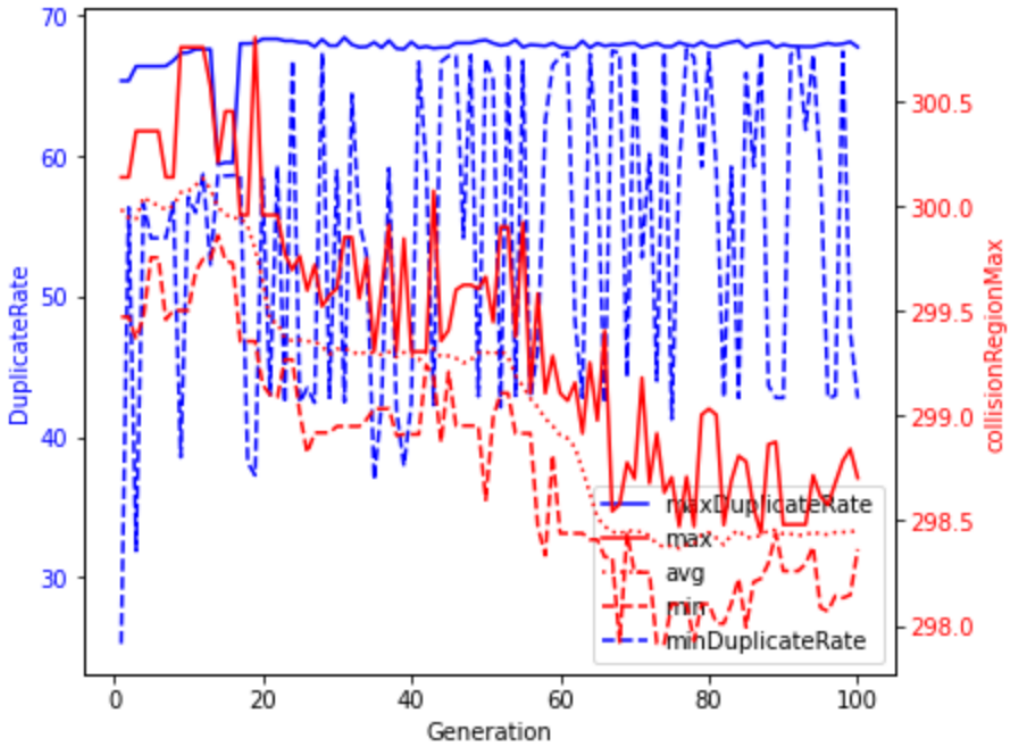
- Why was sigma scattering of 2.0 not recovered?
- Partly due to maxSeedsPerSpM increase from 1 to 2 at gen 19
  - Seeds contain 3 space points. This cut determines the number of seeds to consider per middle space point.
  - In the algorithm, 1 is added to the input (so 0 is really 1)
  - Increases efficiency by considering more seeds, but higher duplicate rate



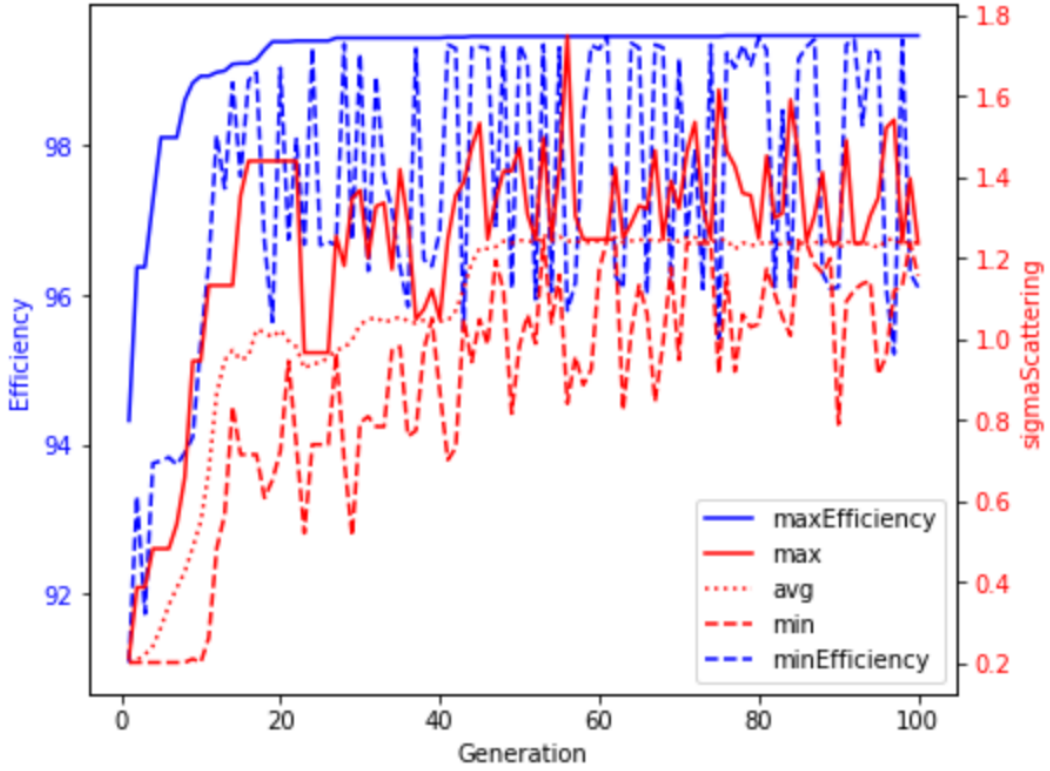




Population Graphs



- More robust than hand tunning
- Population stays relatively centered
- Collision Region Max remained relatively unchanged as expected



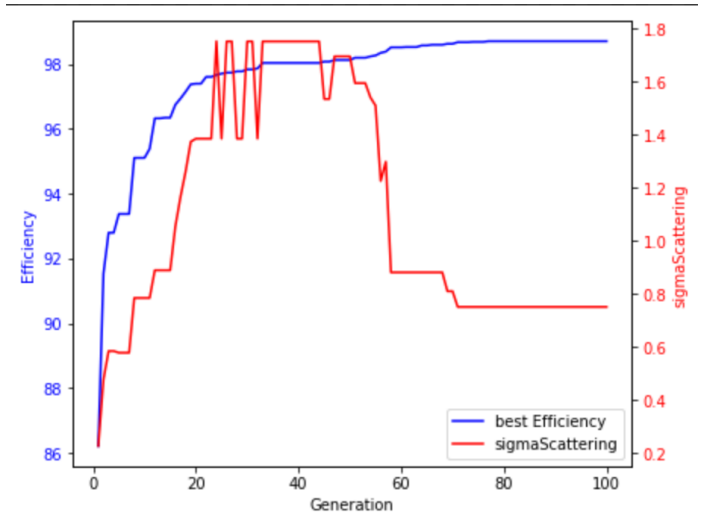
– Highest efficiency in the population
-- Lowest efficiency in the population
– Highest sigmaScattering in the population
-- Lowest sigmaScattering in the population

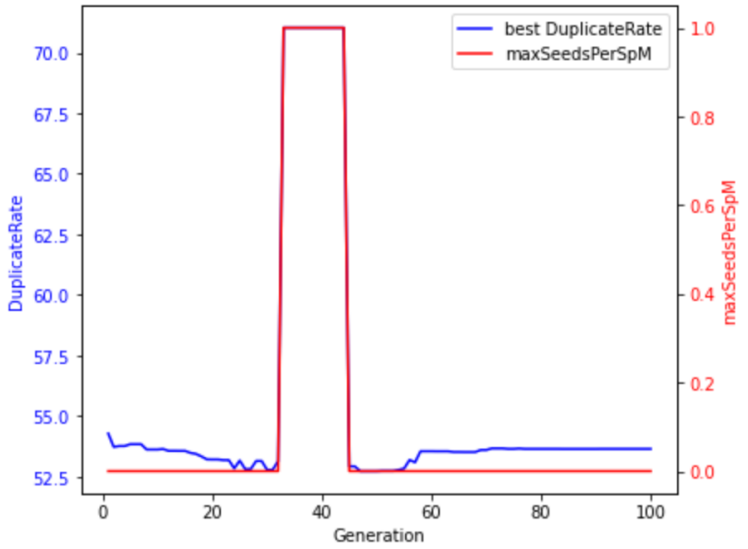
Test 2 With Multiple Bad Parameters

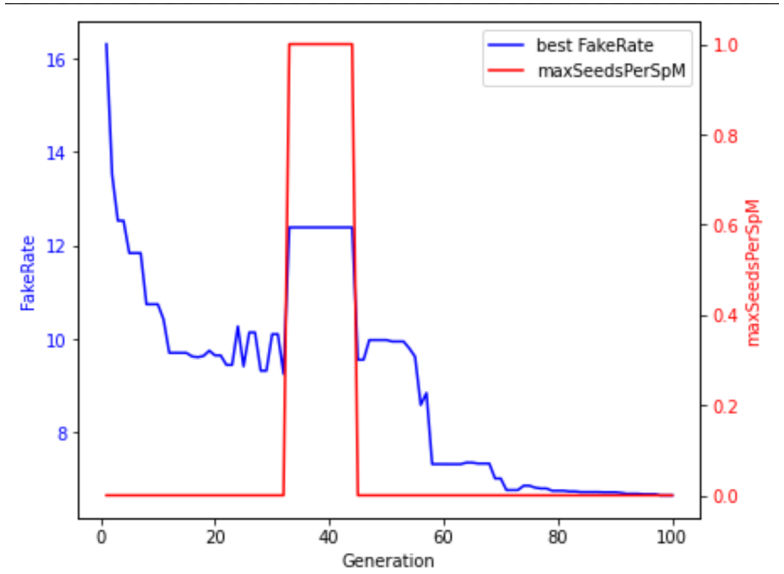
- Goal: Start with several parameters wrong and still find an optimal solution
  - Changed 5 parameters to be off each by a factor >=2
  - Starting point: 85% Efficiency, 17% fake rate, 74% duplicate rate on single muon

Run 2 Best Performing Individual Analysis

- 98.7% Efficiency, 6% fake rate, 54% duplicate rate
- Also, cross checked same configuration on ttbar sample
  - Efficiency 60%->86%, fake rate 76%->64%, duplicate rate 8%->17%
  - Although trained on single muon sample, generalizes to ttbar

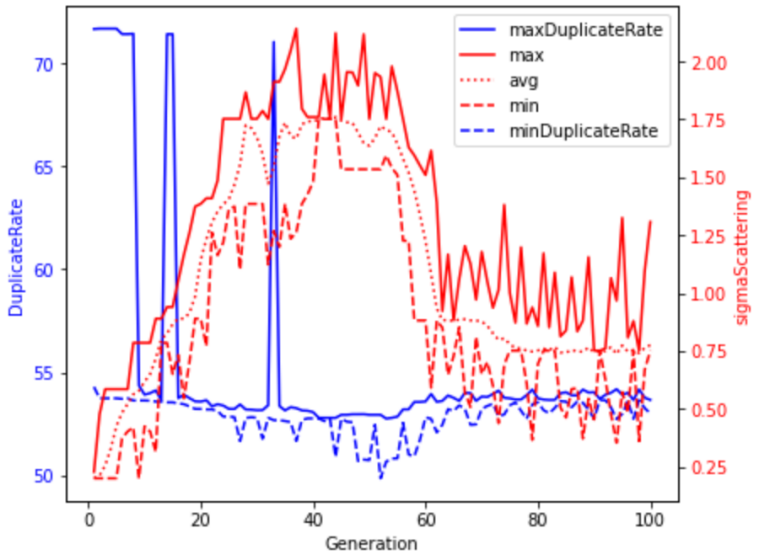


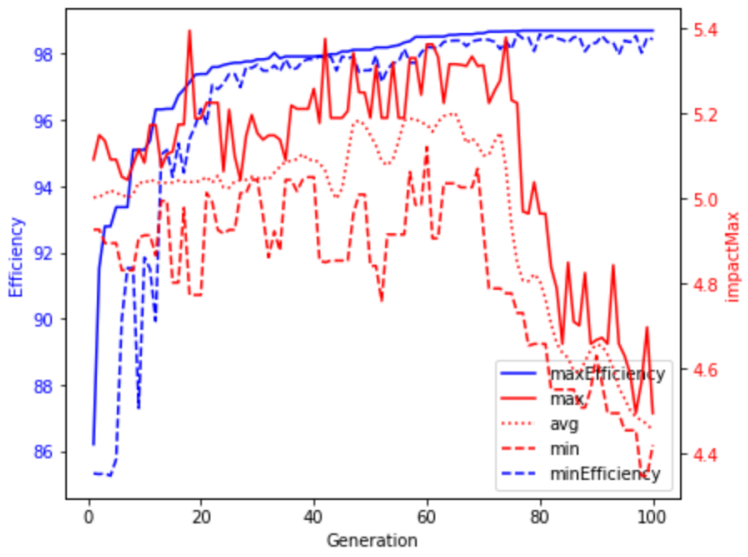


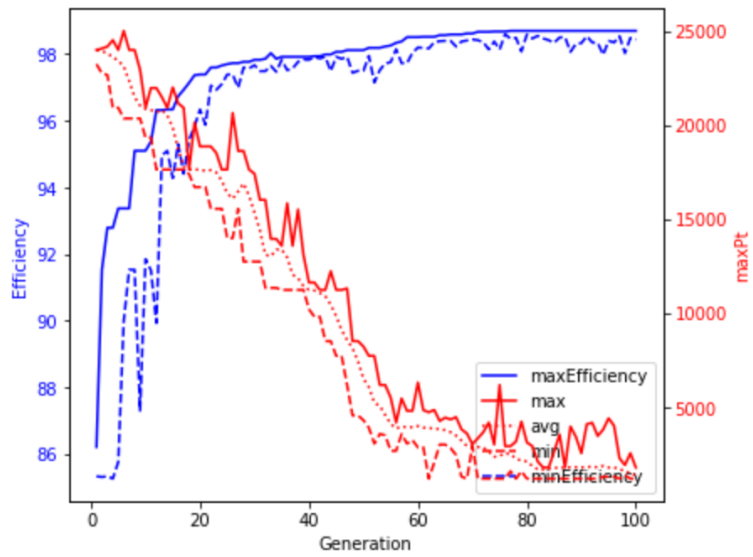


Test 2 Population Analysis

- More generations likely needed for better performance
  - Optimal configuration has impact max of 1.0
  - Impact Max = cut on impact parameter of a seed (how close is the closest point of particle helix to interaction point).







Working On Next

- A hyper parameter tunning algorithm for the seedfinder that can work on the ITK geometry, and 2 other geometries
- Ideas
  - Try a gradient based search
  - Try more evaluation metrics
  - Test on ttbar samples
  - Use a validation set
- Excited to try other seeding or tracking algorithms in the future

Feedback

- Any previous studies on optimizing seeding or tracking algorithms are greatly appreciated
- I would love to hear suggestions. My code is located here: https://github.com/Pchatain/seedingWithEA
- Any questions?