Parameter Optimization in ACTS Using Orion

Rama Salahat

An Najah National University - IJCLab
Nablus, Palestine
Supervised by: David Rousseau, Corentin Allaire
Acts Machine Learning Meeting
06/03/2022

Outline

- Why is Optimization needed?
- Orion Python Library
- The Seeding Optimization Pipeline Specifications
- Results
- Baseline comparison

Introduction

Track Reconstruction is a vital part of all high-energy physics experiments

- Very computationally expensive and memory intensive

- Solution needs to be based on the geometry and material type
  => *a lot of parameters*.

- Good efficiency is essential, but avoiding duplicate tracks is a big challenge too

Challenge:

Having a lot of complex characteristics that vary among different detectors and experimental conditions, thus the need for a flexible on-the-spot optimization tool.

Parameters are currently optimized by hand, automating the process would save a lot of time and manpower.

Orion

Oríon is an open-source python framework for black-box function optimization.

- Easy to integrate and deeply customizable, as it’s code agnostic.
- Asynchronous, enabling parallelisation
- Offers a set of easily plugable algorithms
- Provides several plotting helpers

Orion Algorithms

- Orion impeded algorithms:
  - **Random Search**
  - **Grid Search**
  - **Hyperband**
  - **ASHA**: Asynchronous Successive Halving Algorithm
  - **TPE**: Tree-structured Parzen Estimator
  - **Evolution-ES**
- Algorithm Plugins
  - **Scikit-Optimize**: providing a wrapper for *skopt* optimizers, e.g Scikit Bayesian Optimizer
  - **Robust Bayesian Optimization**: providing a wrapper for *RoBO* optimizers, e.g: Gaussian Process, Gaussian Process with MCMC, Random Forest, DNGO and BOHAMIANN.

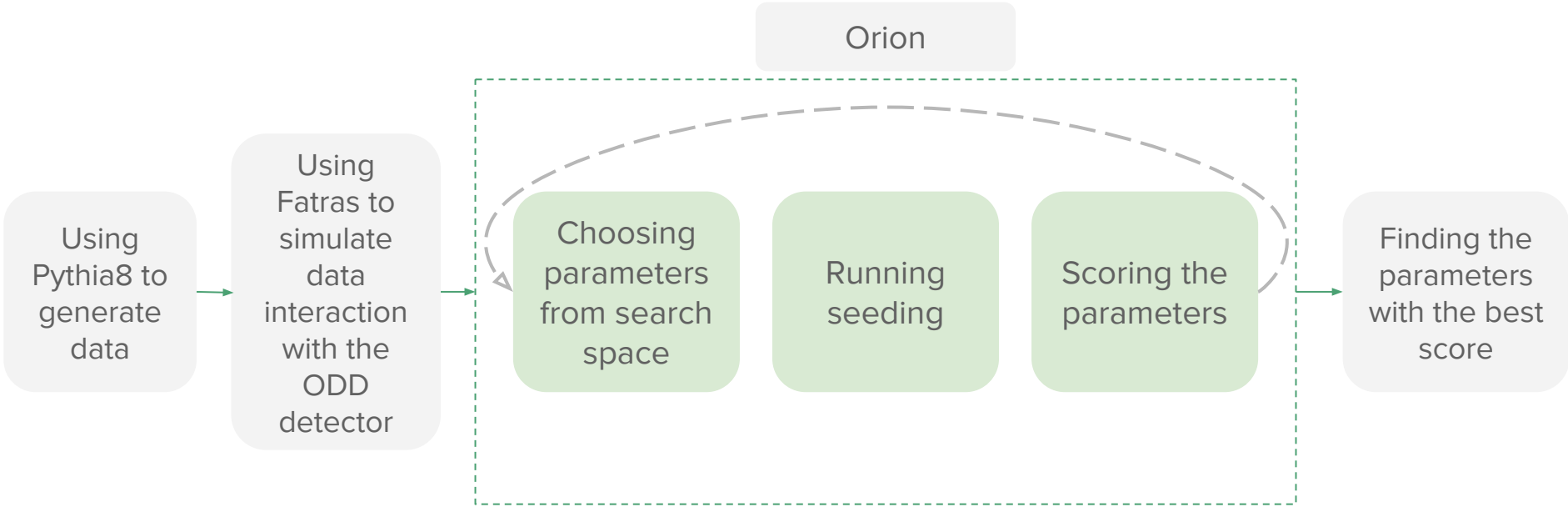
Orion Parallel Strategies

- Informatively picking parameters from the search space is usually an iterative process
- Orion is able to use “lies” or unfinished trials with fake objectives to pick the next params (with a temporary algorithm that gets discarded later)
  => Allowing **parallelisation** with minor efficiency affects

Parallel Strategies:

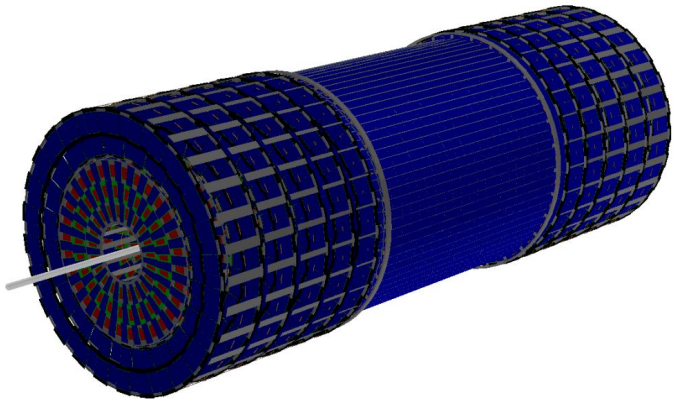
- NoParallelStrategy
- StubParallelStrategy: assign lies a certain objective value
- MaxParallelStrategy: assign lies the current max value
- MeanParallelStrategy: assign lies the current mean value

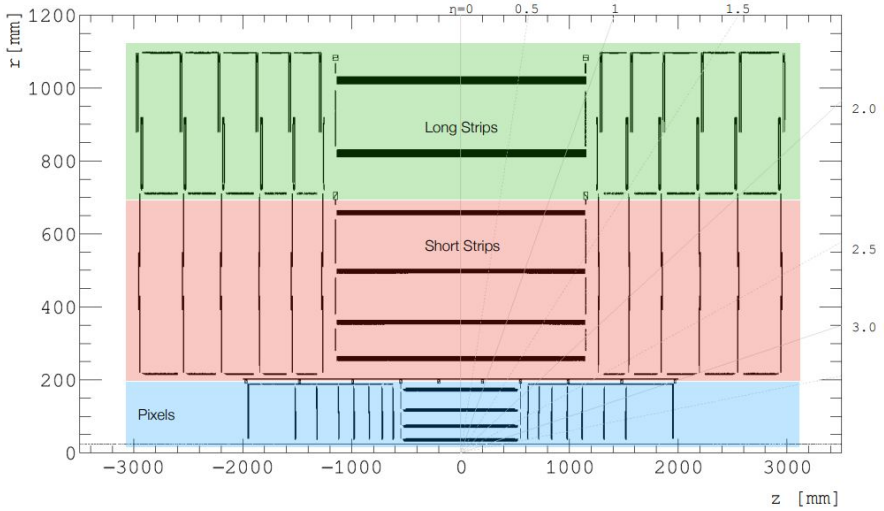
Optimization Pipeline Example



Open Data Detector (ODD)

- A general purpose, all-silicon, virtual detector
- ODD is similar to the ATLAS ITK upgrade, but without inclined modules
- ODD was used in the data simulation of the optimization pipeline





Scoring

- The algorithm tries to minimize the following equation:

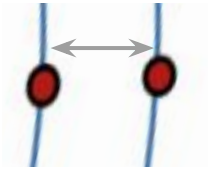
Score = 1 - (efficiency - (((100 * fakeRate) + avgDuplicate) / 1000))

- efficiency = totalMatchedParticles / totalParticles
- fakeRate = (totalSeeds - totalMatchedSeeds) / totalSeeds
- avgDuplicate = (totalMatchedSeeds - totalMatchedParticles) / totalMatchedParticles

Chosen Parameters to Tune The Seeding

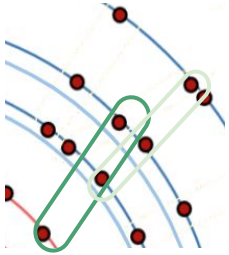
***Seed Finding Params*:**

- **minPt**: : pT(transverse momentum) cut off
- **deltaRMin**: min distance between 2 measurements
- **deltaRMax**: max distance between 2 measurements
- **radLengthPerSeed**: estimated radiation length encountered per seed



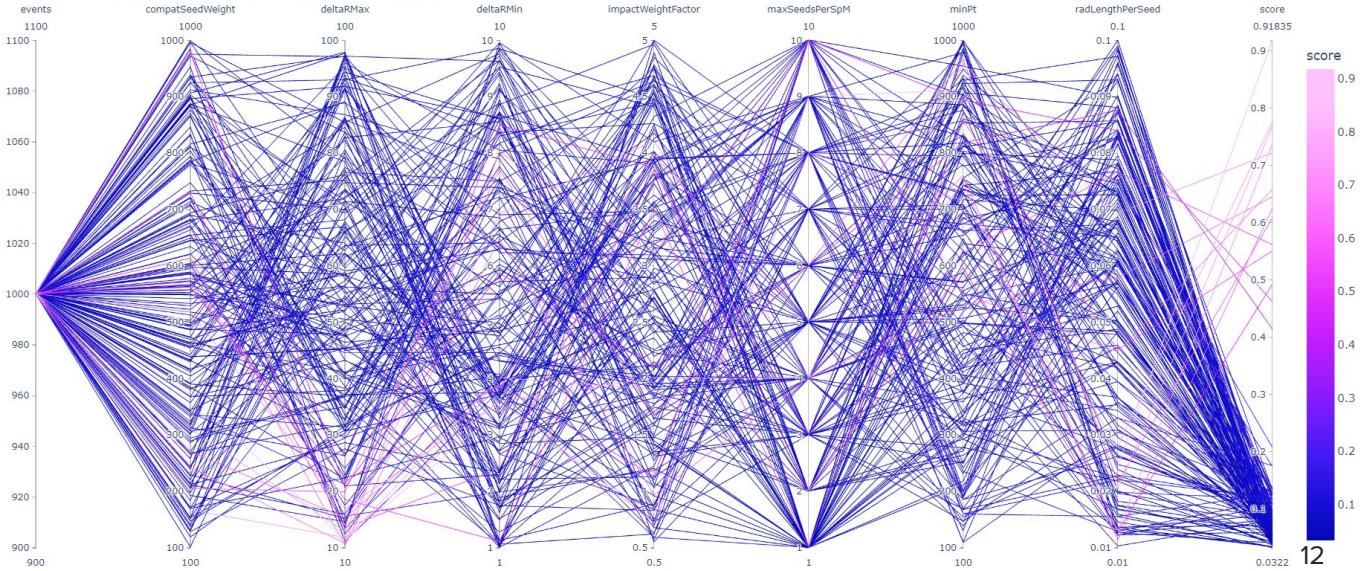
***Seed Filtering Params*:**

- **maxSeedsPerSpM**: maximum number of seeds considered, per space point
- **compatSeedWeight**: boost for compatible seed
- **impactWeightFactor**: effect of the impact param on the weight



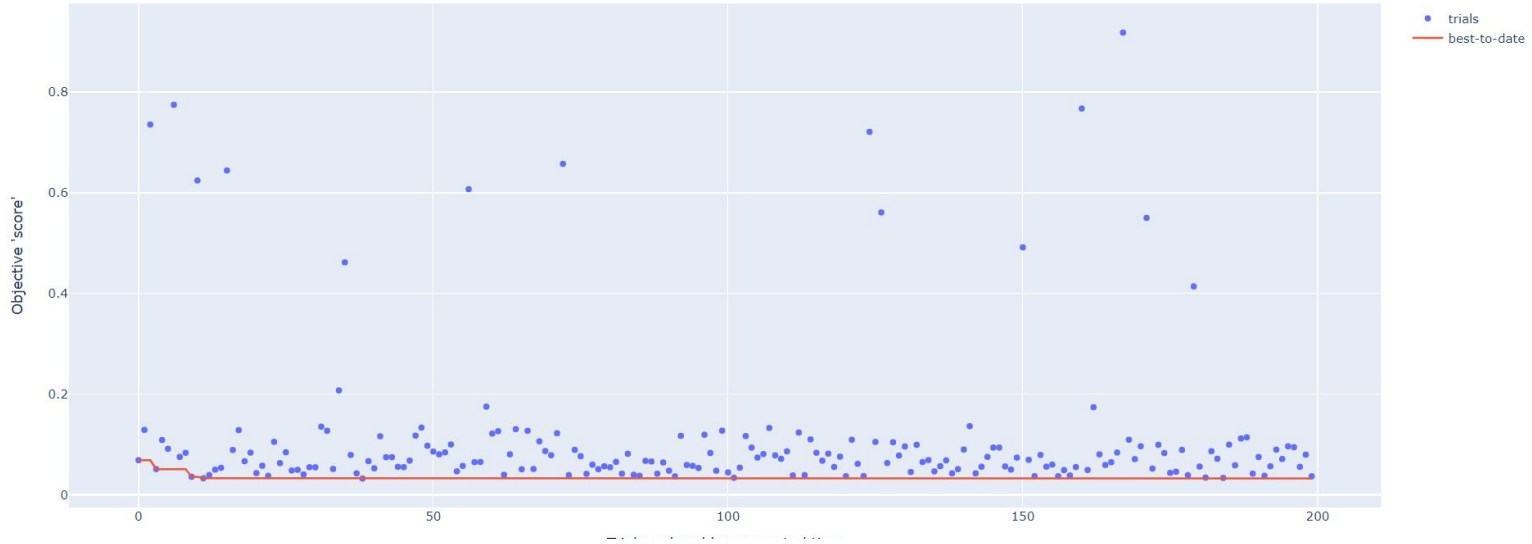
Algorithm: Random Search

- Random Search: samples randomly from the problem's space
- The parallel coordinates plot below displays the params selection for each trial and their corresponding score, and here it looks indeed random



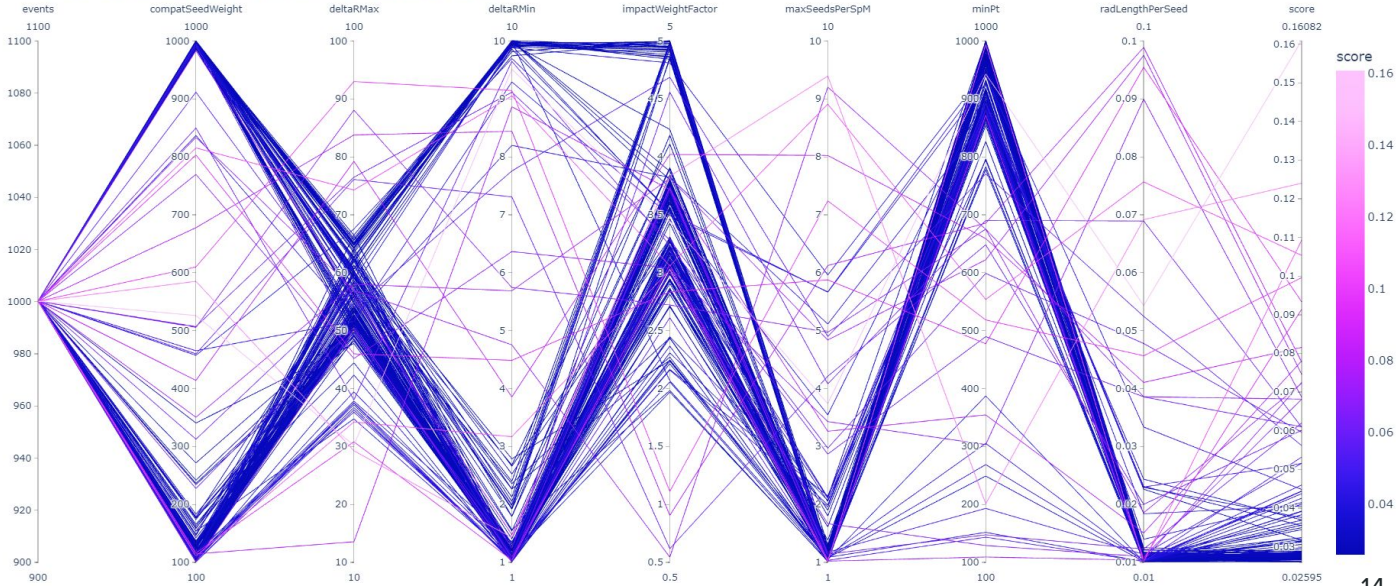
Algorithm: Random Search

The Regret Plot displays trials and their scores, here it shows that even though the search is random, the algorithm was able to find a lot of good solutions!



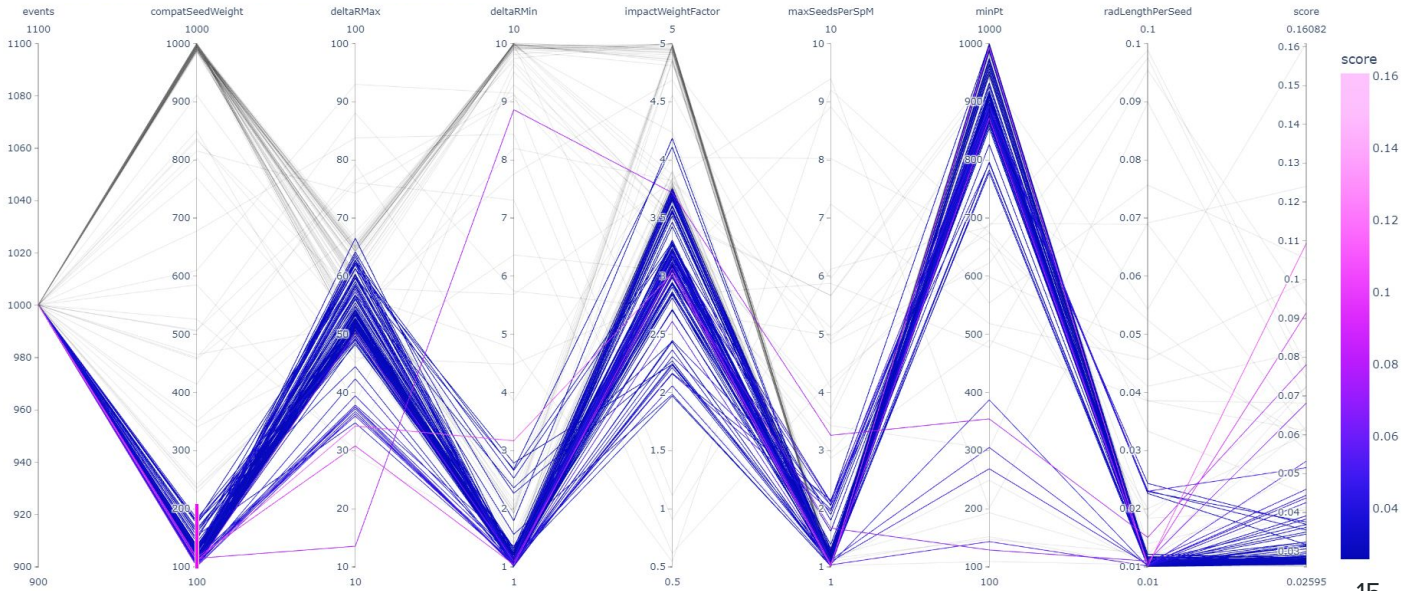
Algorithm: Tree-structured Parzen Estimator (TPE)

- TPE chooses parameters from space according to previous trials' results
- There's more distinguished converging in this experiment



Algorithm: Tree-structured Parzen Estimator (TPE)

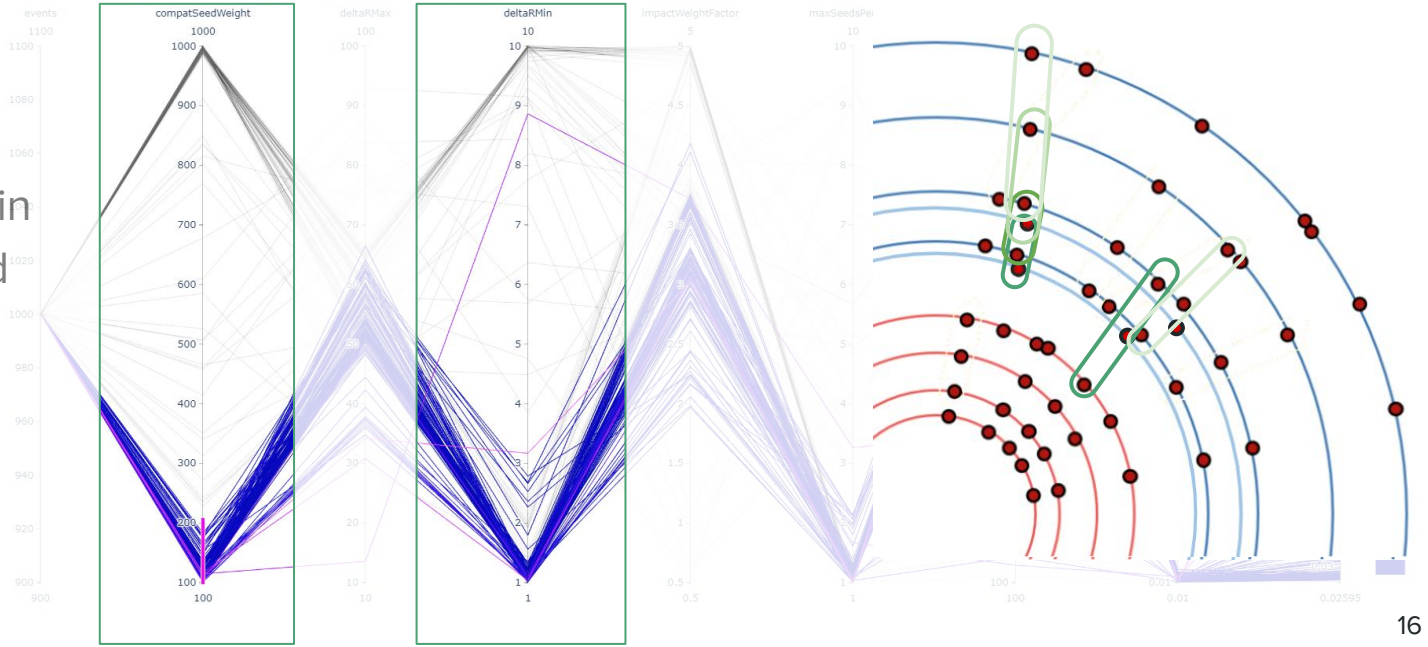
- It seems like there's 2 converged patterns!
- One with a low compatSeedWeight and low deltaRMin and the other with a high compatSeed Weight and high deltaRMin



Algorithm: Tree-structured Parzen Estimator (TPE)

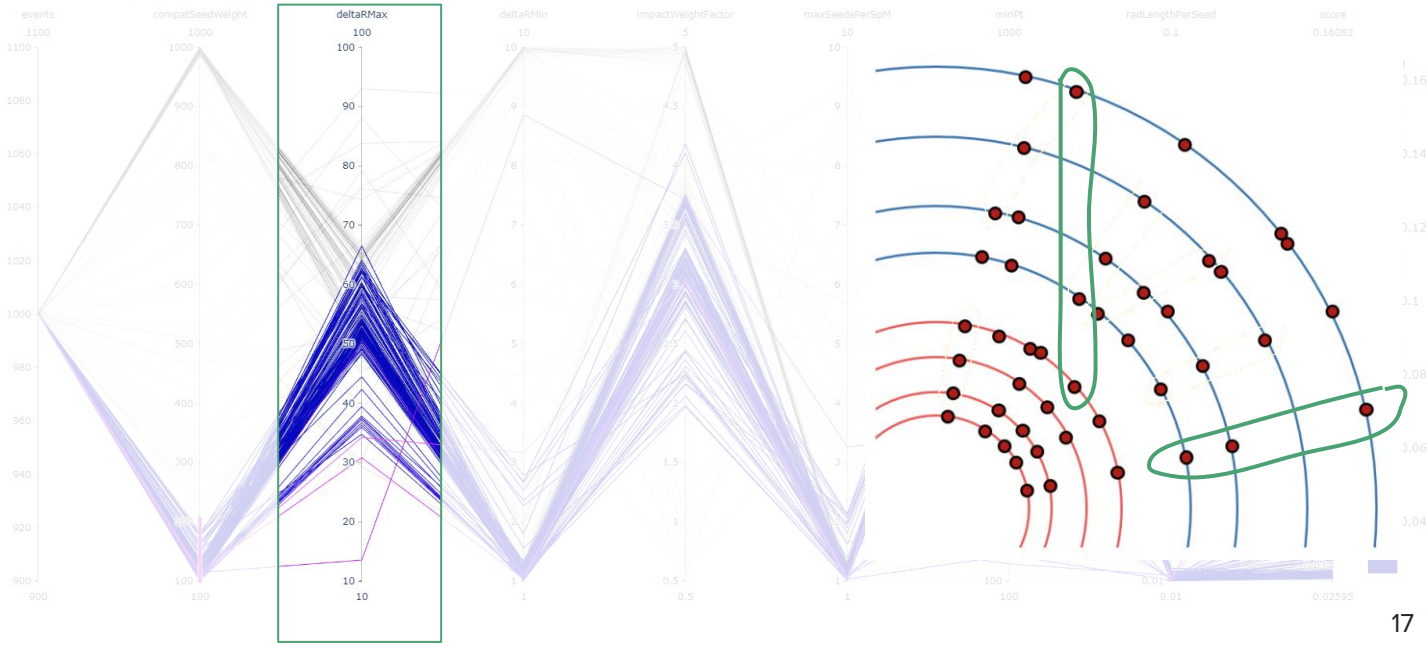
- When compatSeedWeight is low, creating the duplicate seeds from the same layer *stops boosting the score*, so having a low value for deltaRMin would include a lot of good seeds.

- And vice versa, the high deltaRMin would be needed to exclude such seeds



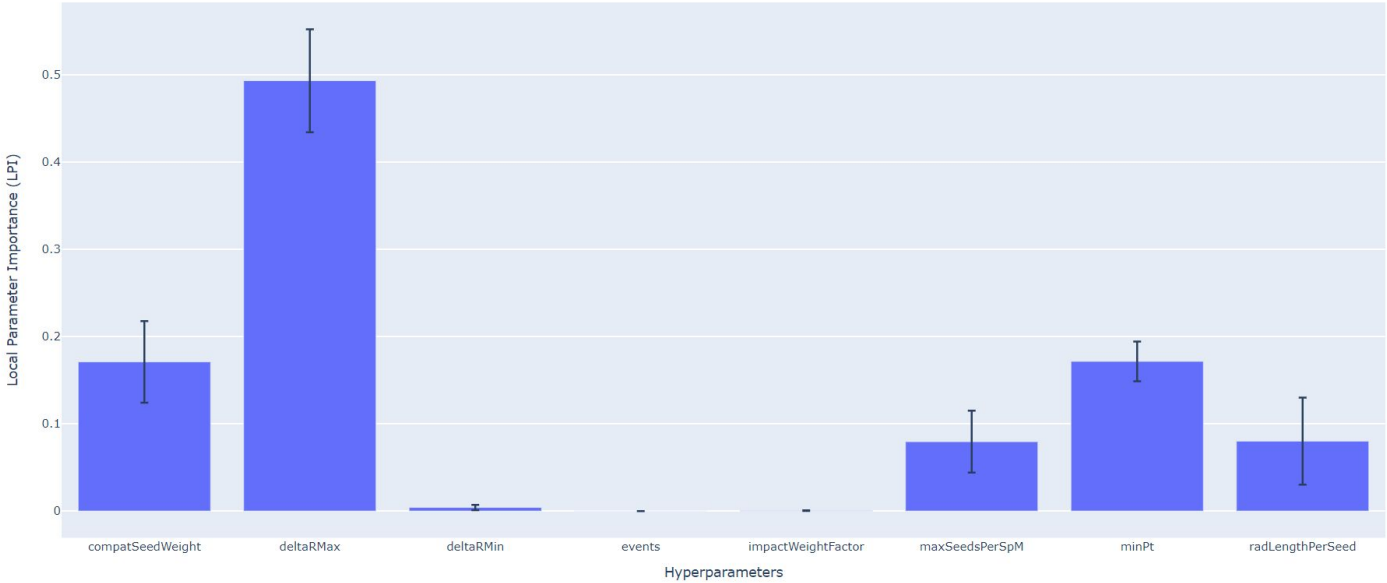
Algorithm: Tree-structured Parzen Estimator (TPE)

- The value of deltaRMax converged to a balanced range, that would enable the seeding to work with detector holes, but also not cause faulty seeds



Algorithm: Tree-structured Parzen Estimator (TPE)

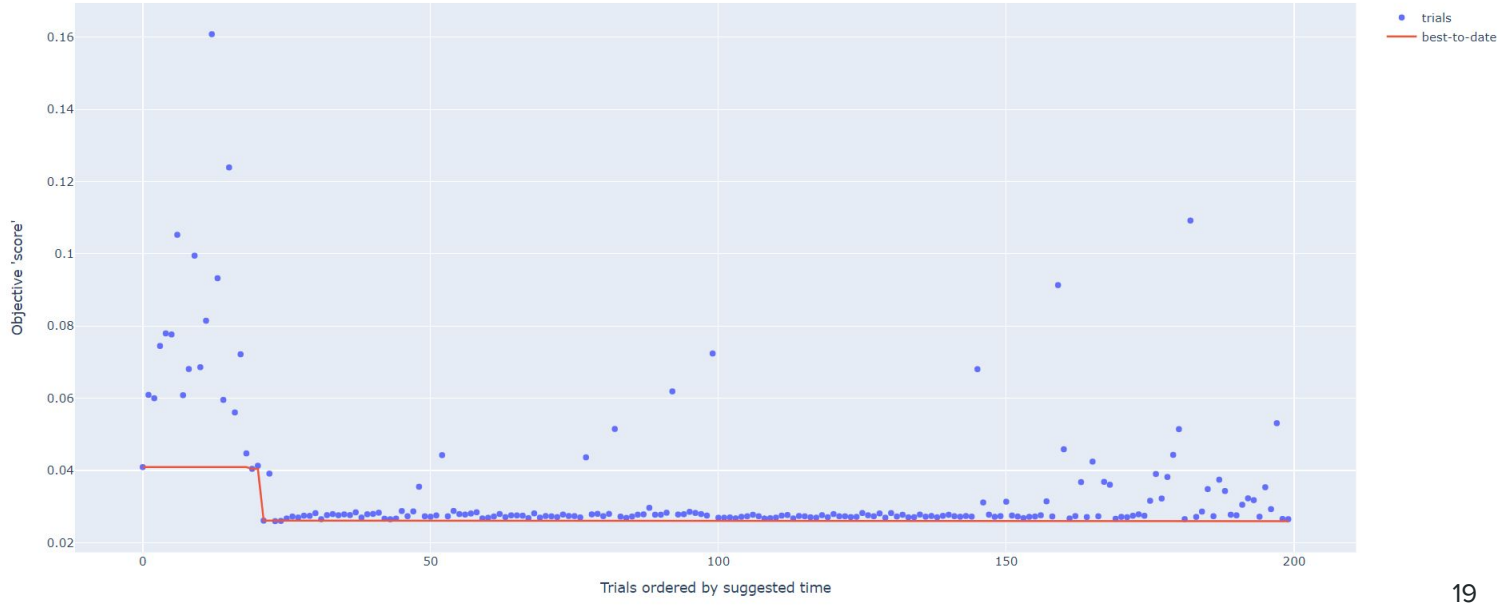
Local Parameter Importance Plotting: measures the variance of the results when varying one hyperparameter and keeping all other fixed



Algorithm: Tree-structured Parzen Estimator (TPE)

TPE seems to have converged really early(~trial 25)

There was still some bad trials after converging.



Baseline Comparison

	Efficiency	Fake Rate	Duplicate Rate*	Average Duplicate	Objective*
Optimised Params with TPE	0.985	0.0513	0.965	28.670	0.034
Baseline1*	0.994	0.029	0.962	29.014	0.036
Baseline2*	0.984	0.074	0.965	27.556	0.050

Baseline1* is the default seeding parameters for the seeding python bindings in ACTS
Baseline2* is the default seeding values of seeding.py python example in ACTS
Duplicate Rate* wasn't used in the scoring methodology
Objective* is the value the algorithm was trying to minimize

Trials' Parameters

	Optimised with TPE	Baseline1	Baseline2
maxSeedsPerSpM	1	1	1
minPt	257.9	400.	500.
deltaRMax	45.75	270	60
deltaRMin	2.06	5	1
radLengthPerSeed	0.01121	0.05	0.1
compatSeedWeight	105.0	200	200
impactWeightFactor	2.614	1	1

Conclusion

- Orion was used to optimize the seeding process parameters, with the openDataDetector.
- Re-running the optimization pipeline with a more realistic detector, and with other algorithms(Orion provides a long list of algos) could result in computing better parameters
- Applying the pipeline to other processes than the seeding