Auto-Tuning of Seeding and Vertexing algorithms in ACTS Software Framework

ROCKY BALA GARG

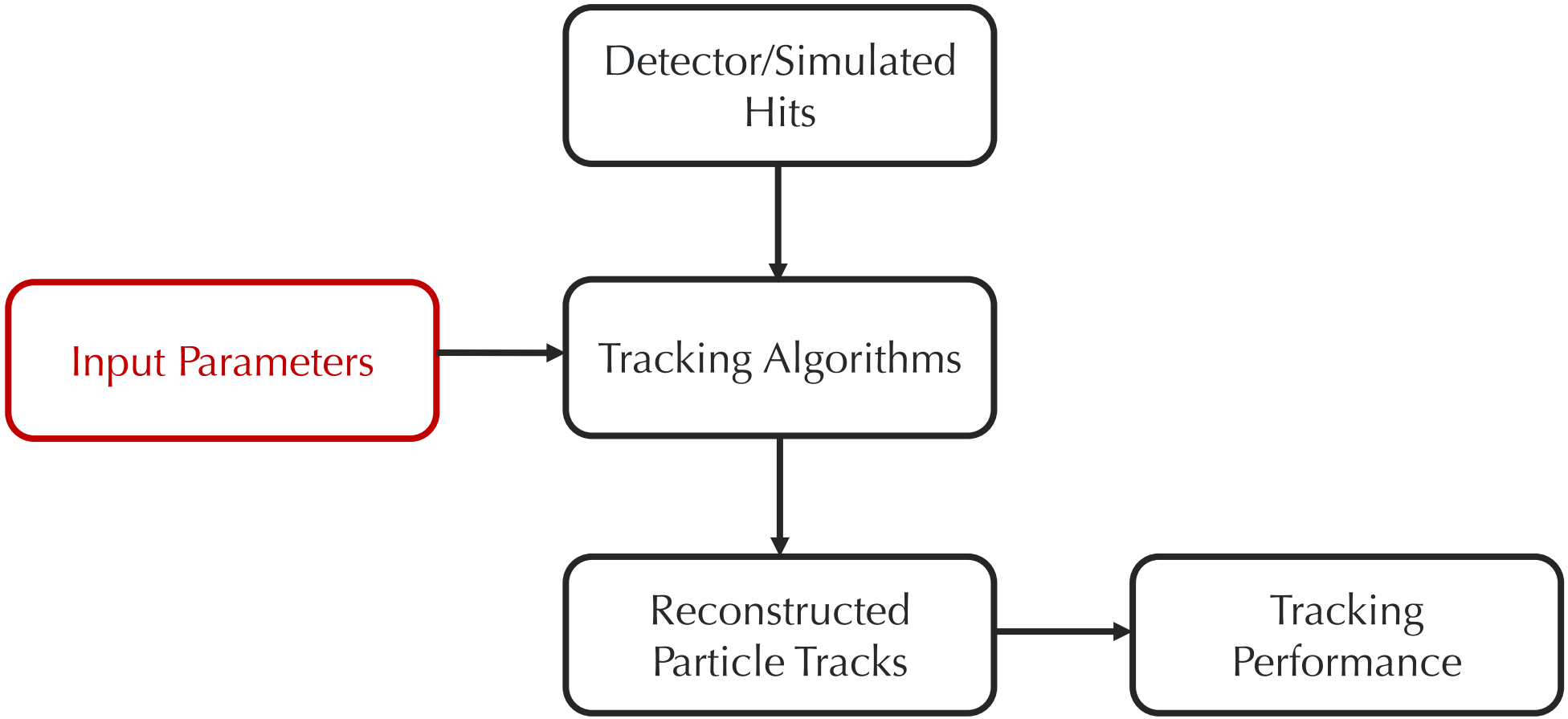
STANFORD UNIVERSITY

(Results presented in CTD2022)

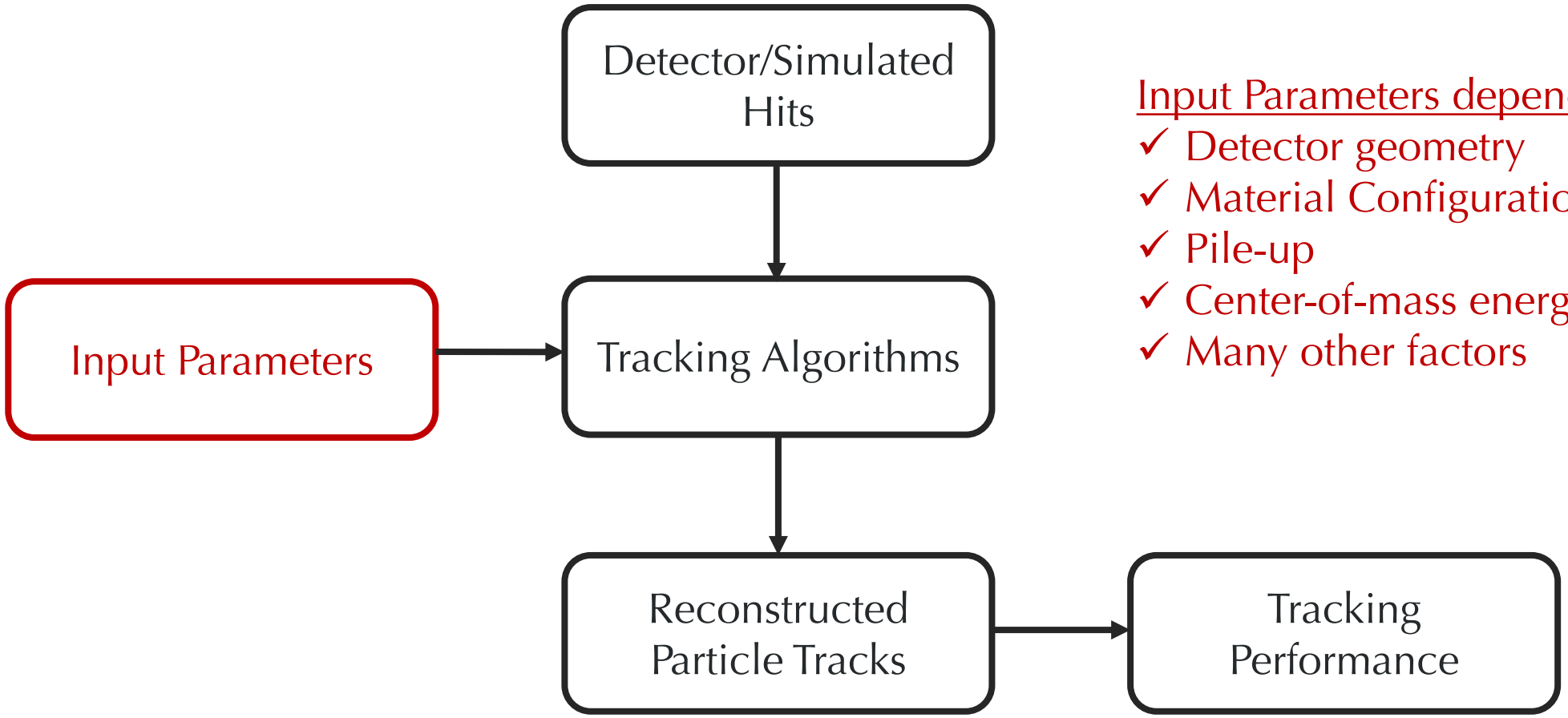
ACTS Developers Workshop, CERN

Sept 26th – 30th

Motivation


Detector/Simulated Hits
Input Parameters
Tracking Algorithms
Reconstructed Particle Tracks
Tracking Performance

Motivation

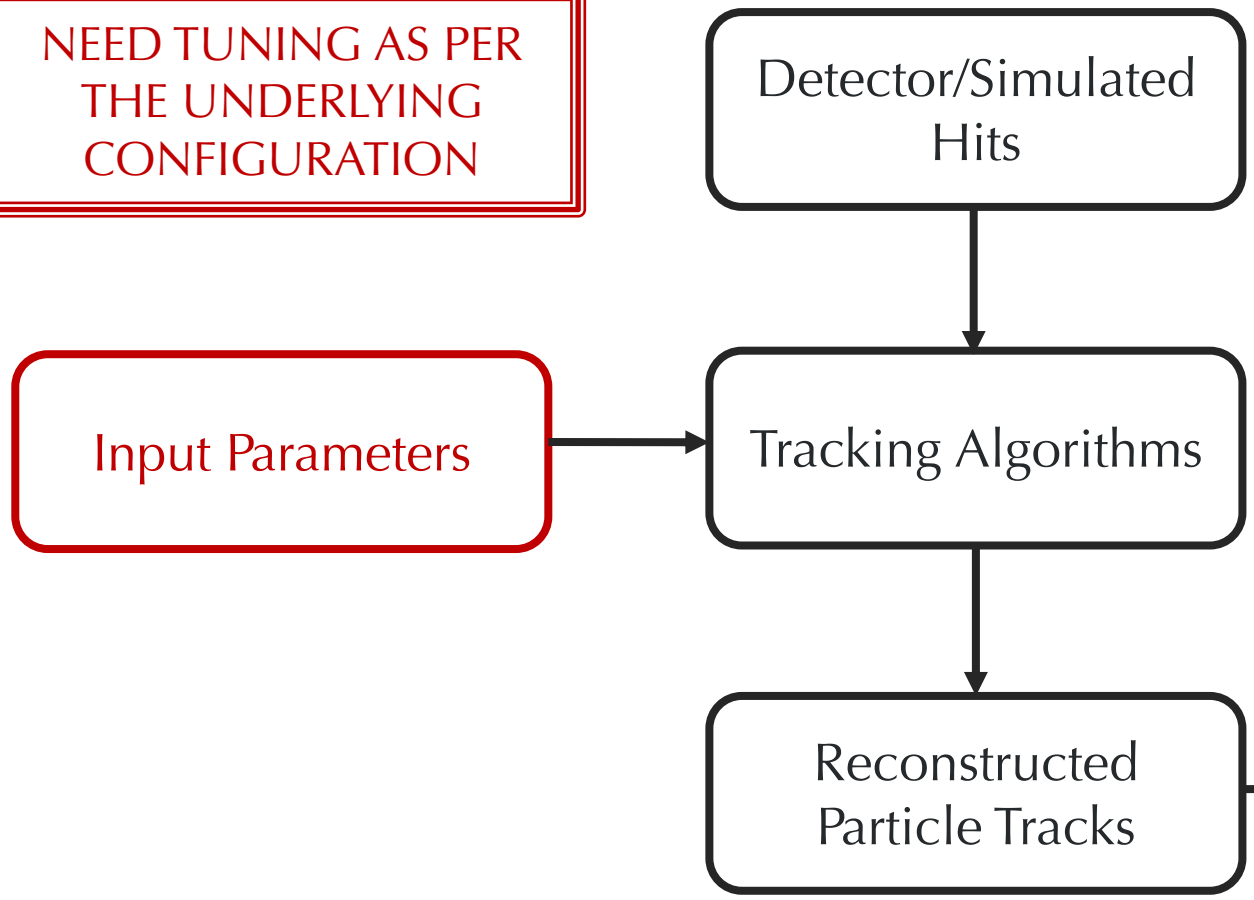


Input Parameters depend on:

- ✓ Detector geometry
- ✓ Material Configuration
- ✓ Pile-up
- ✓ Center-of-mass energy
- ✓ Many other factors

Motivation

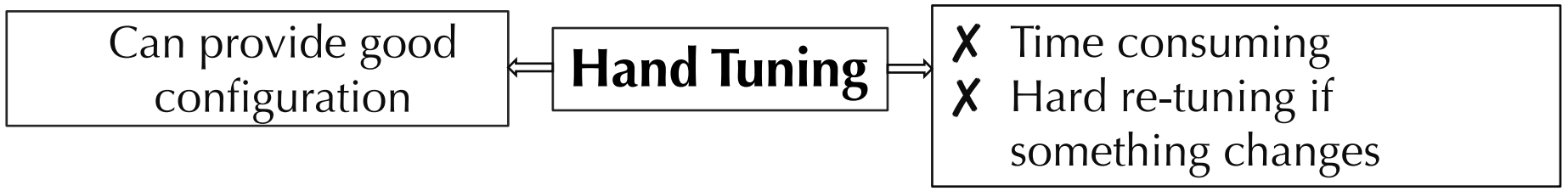
NEED TUNING AS PER THE UNDERLYING CONFIGURATION



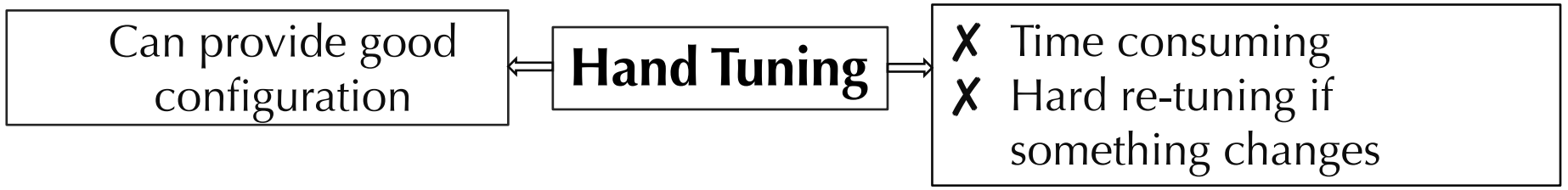
Input Parameters depend on:

- ✓ Detector geometry
- ✓ Material Configuration
- ✓ Pile-up
- ✓ Center-of-mass energy
- ✓ Many other factors

Current/Most Popular Approach



Current/Most Popular Approach



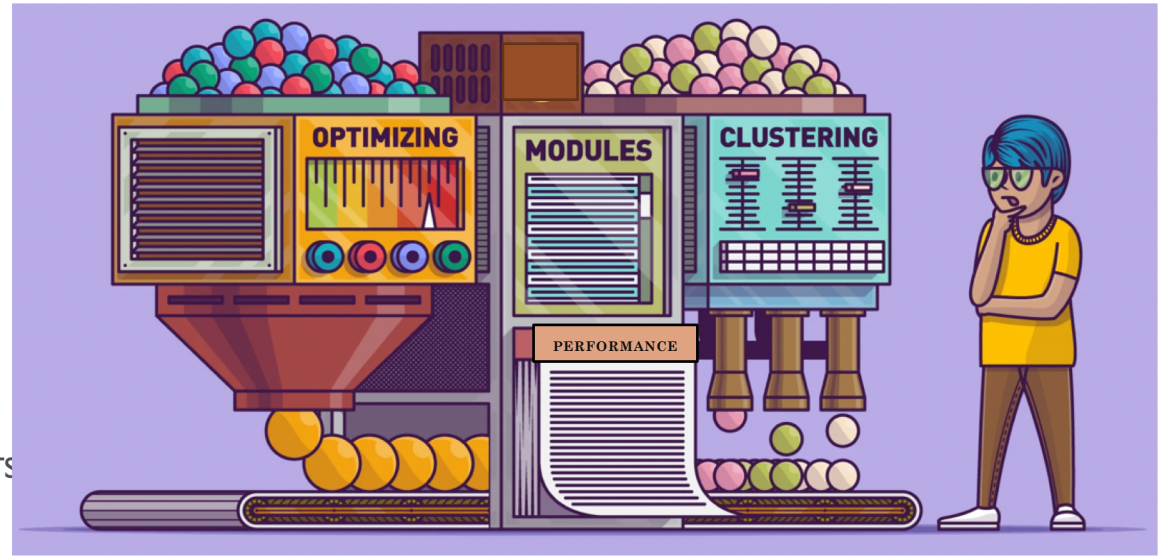
New Approaches Are Needed!!

Current/Most Popular Approach

Can provide good configuration ⟺ **Hand Tuning** ⟹

✗ Time consuming
✗ Hard re-tuning if something changes

Automatic Parameter Tuning???



✓ Efficient
✓ Less time consuming
✓ Easy re-tuning

Studied Different Optimization Frameworks
Derivative-free approaches

DISTRIBUTED EVOLUTIONARY ALGORITHMS IN PYTHON

EA: Based on genetic evolution in living organisms

Optuna: Open source software for automatic hyperparameter search

Lipschitz Optimization

LIPO: Simple algorithm based on a mathematical model with no hyperparameters of its own

Skopt: Optimization framework built in Python sklearn library

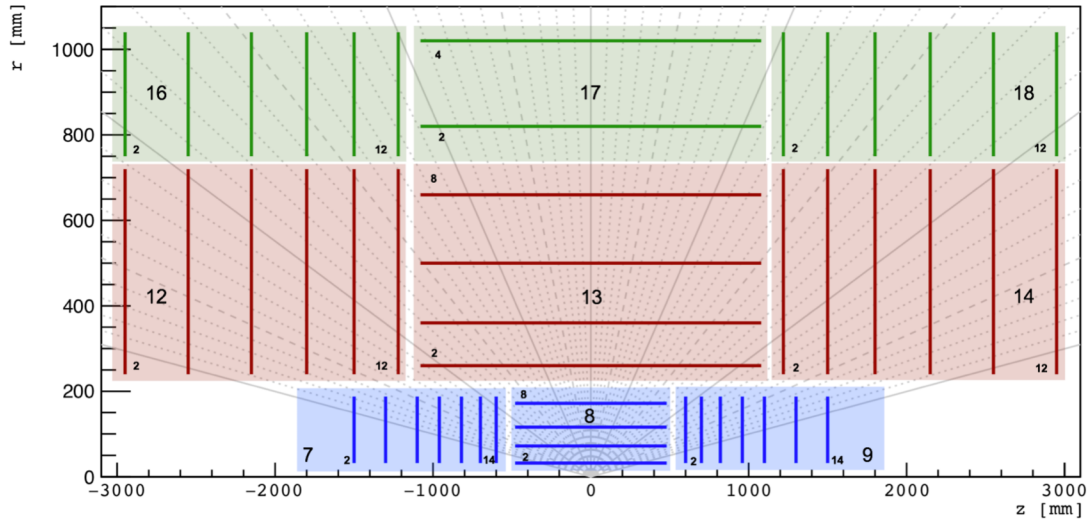
Orion: Black box optimization framework with different optimization techniques

Test Bed for our studies:
A Common Tracking Software (ACTS)

Input Dataset:

- ✓ ttbar process for pile-up = 140 & 200 at 14 TeV
- ✓ Generated using Pythia8 and simulated through Generic detector within ACTS
- ✓ Min P_T of particles > 1 GeV
- ✓ $|\eta|$ of particles < 2.5

In-built Generic/TrackML detector
Equivalent to a typical full silicon LHC detector



Tracking Algorithms studied

Track Finder/Fitter

Combinatorial Kalman Filter (CKF)

- ✓ In ACTS, main bottleneck in CKF performance: Track Seeding (~ 98% efficiency with truth seeds)
- ✓ Optimized 8 parameters of Track Seeding algorithm
- ✓ Evaluated the performance on CKF

Vertex Finder/Fitter

Adaptive Multi Vertex Finder (AMVF)

- ✓ CKF tracks with truth seeds used as input to AMVF algorithm
- ✓ Optimized 5 parameters of AMVF algorithm
- ✓ Evaluated the performance on AMVF

Performance Evaluation: Score/Objective Function

- ✓ Based on the performance metrics of underlying tracking algorithm
- ✓ Positive weights are given to quantities we want to increase
- ✓ Negative weights are given to quantities we want to decrease

Performance metrics for CKF

- ✓ Tracking Efficiency
- ✓ Duplicate Rate
- ✓ Fake Rate
- ✓ Run-time

Performance metrics for AMVF

- ✓ Total number of reconstructed vertices
- ✓ Reconstructed vertices tagged as
  - ✓ Clean
  - ✓ Merged
  - ✓ Split
  - ✓ Fake
- ✓ Vertex Resolution in x, y and z

Output from an optimization method is highly dependent on the score function used

Performance Evaluation: Score/Objective Function

Combinatorial Kalman Filter (CKF)

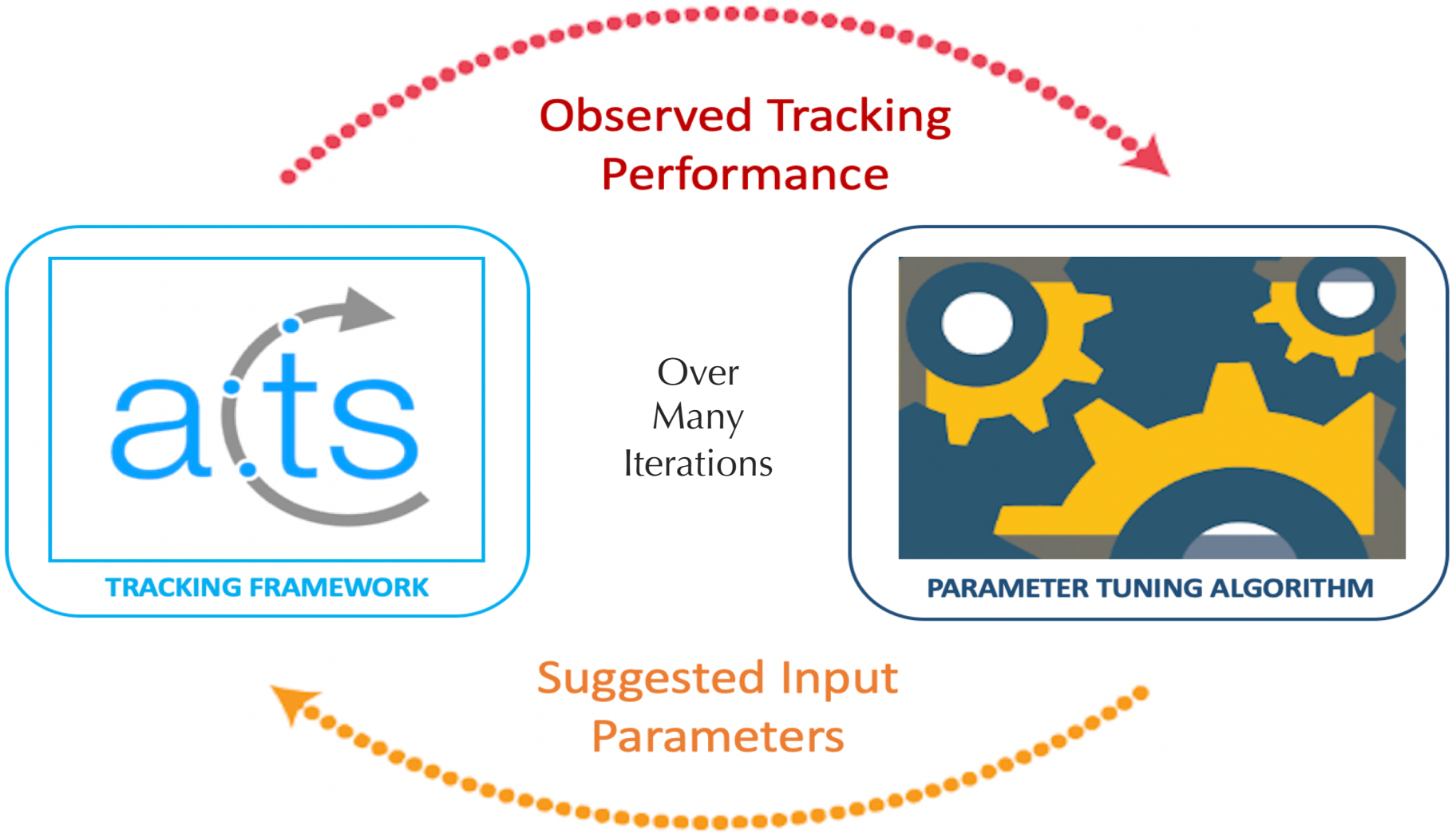
$$\text{Score Function} = \textit{Efficiency} - (\textit{FakeRate} + \frac{\textit{DuplicateRate}}{K} + \frac{\textit{RunTime}}{K}),$$

(K = 7 for all algorithms)

Adaptive Multi Vertex Finder (AMVF)

$$\text{Score Function} = (\textit{Eff}_{\textit{Total}} + 2\textit{Eff}_{\textit{Clean}}) - (\textit{Merged} + \textit{Split} + \textit{Fake} + \textit{Resolution})$$

Integration of two algorithms

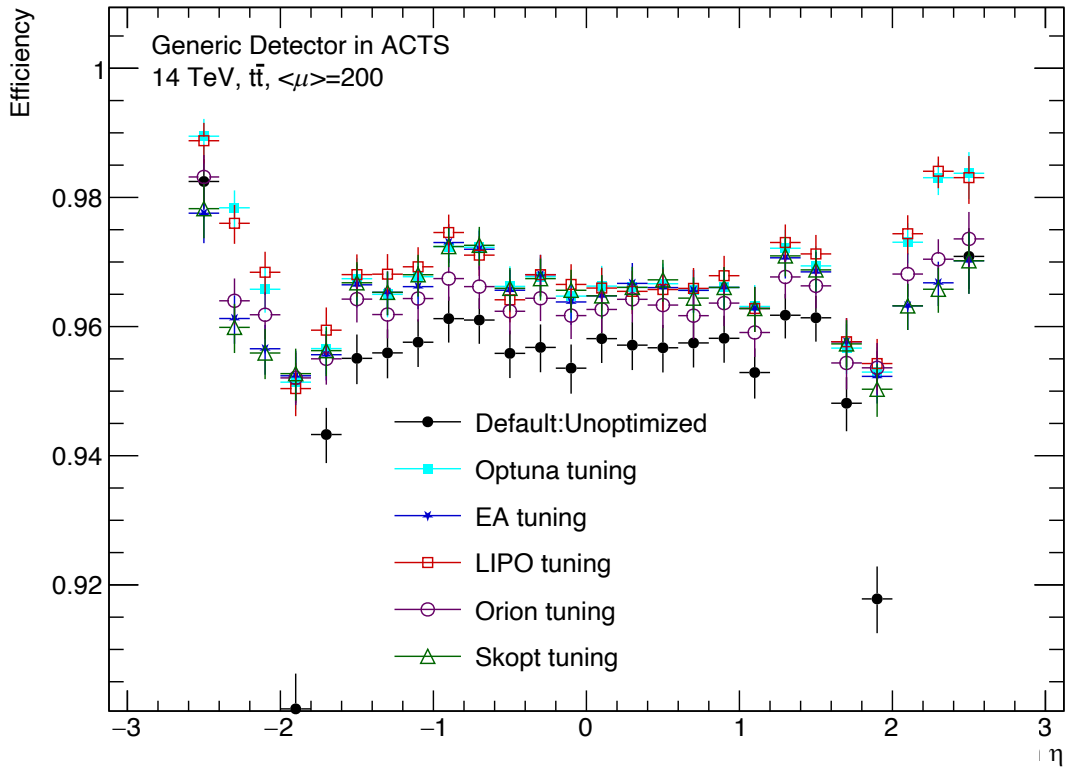


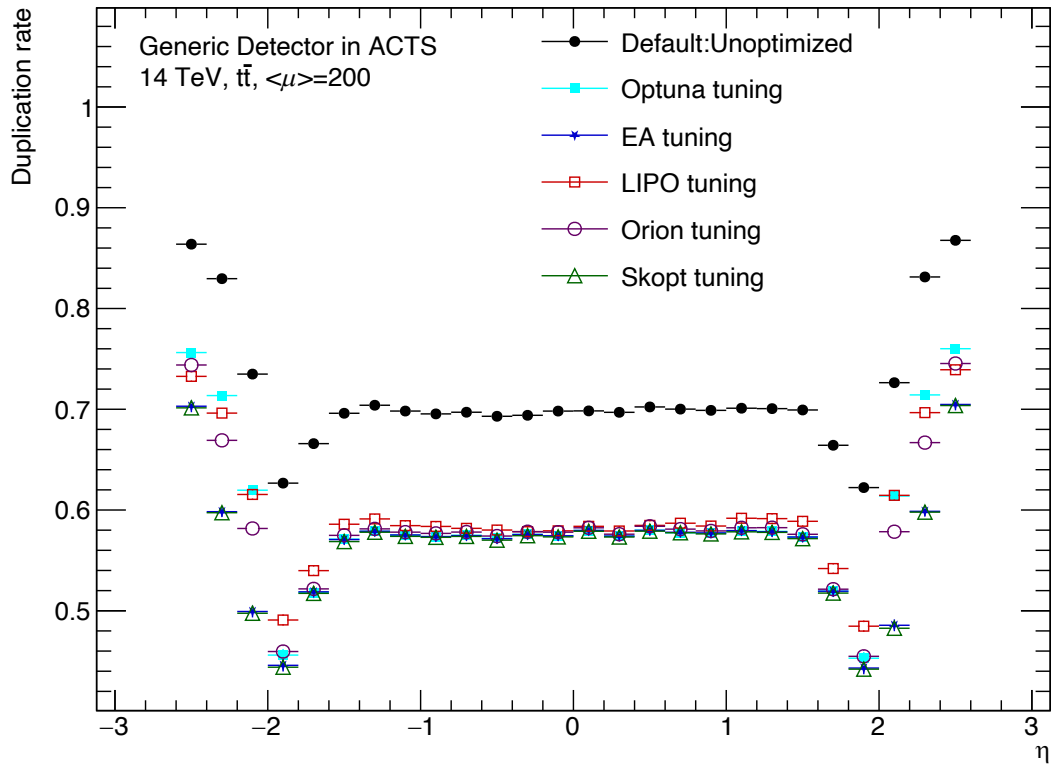
CKF Optimization Results

CKF performance metrics are evaluated and compared for default and optimized parameters!!

Track Efficiency and Duplicate Rate vs η

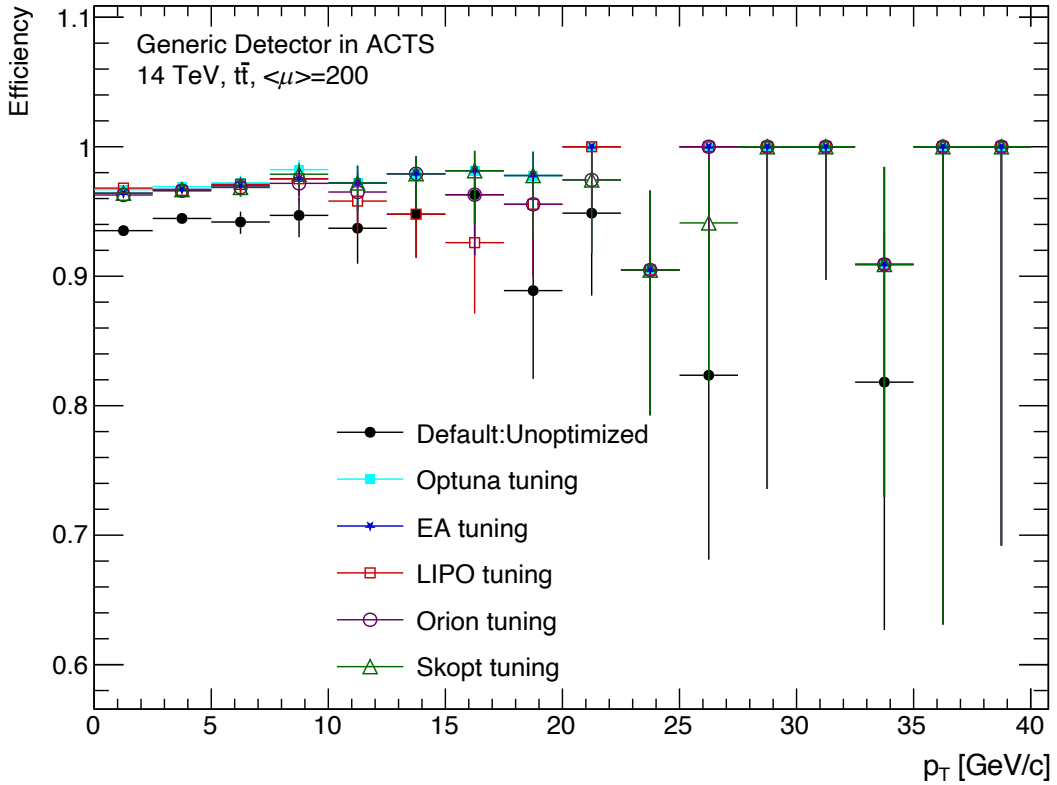
Visible improvement in performance with tuned parameters

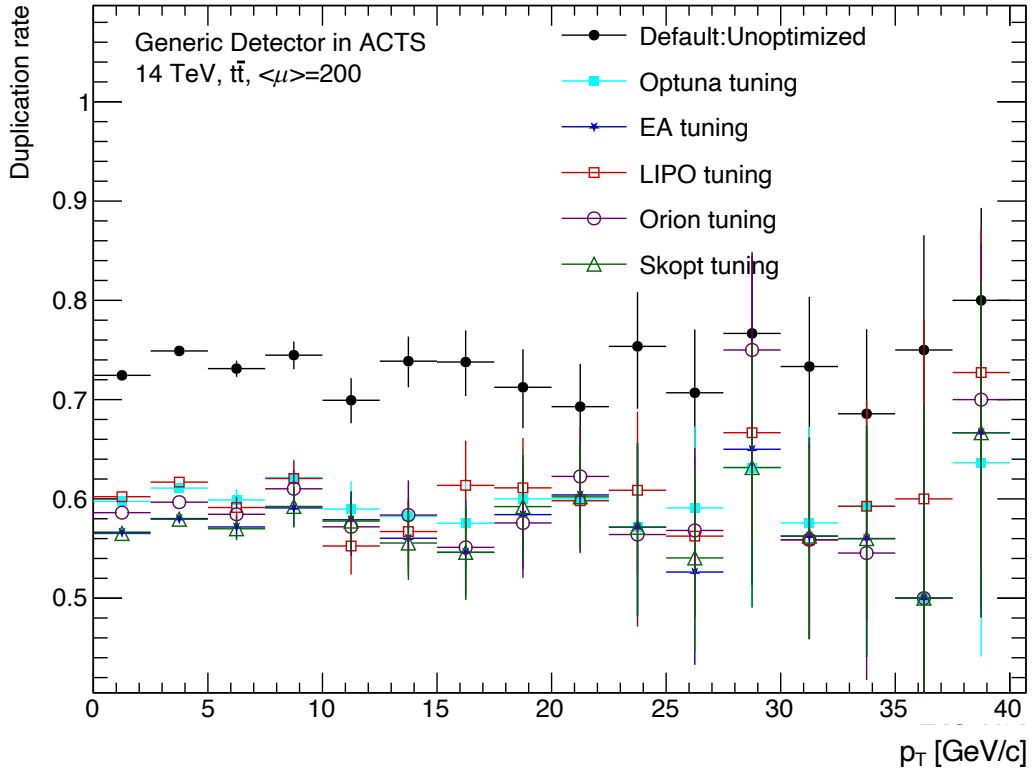




Track Efficiency and Duplicate Rate vs P_T

Visible improvement in performance with tuned parameters





CKF Optimization Results

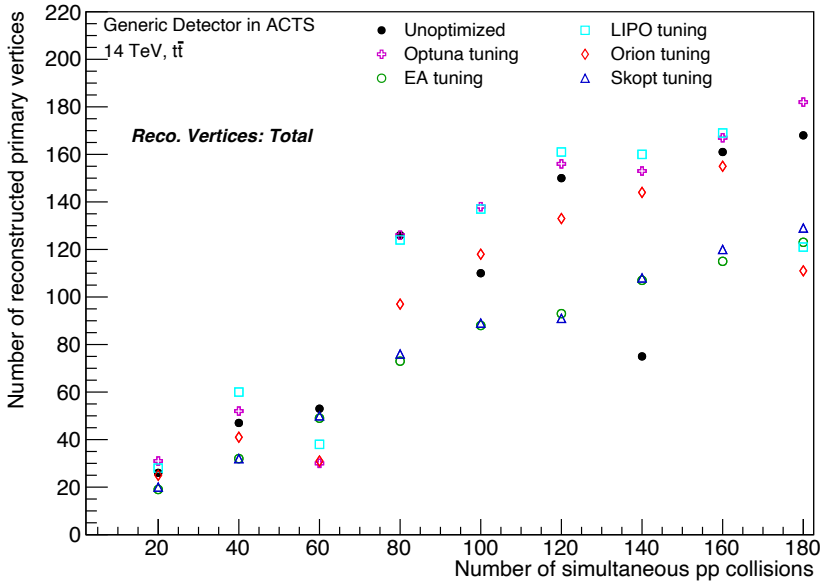
	Default	EA	Optuna	LIPO	Skopt	Orion
Score	69.06	78.88	75.38	77.28	77.05	77.79
Efficiency	0.936	0.965	0.967	0.968	0.965	0.963
Duplicate Rate	0.726	0.568	0.598	0.603	0.566	0.587
Fake Rate	5.56E-05	6.2E-05	5.2E-05	6.8E-05	5.7E-05	8.8E-05
Total(sec) time/event	50.2	31.1	46.8	37.2	40.2	33.9

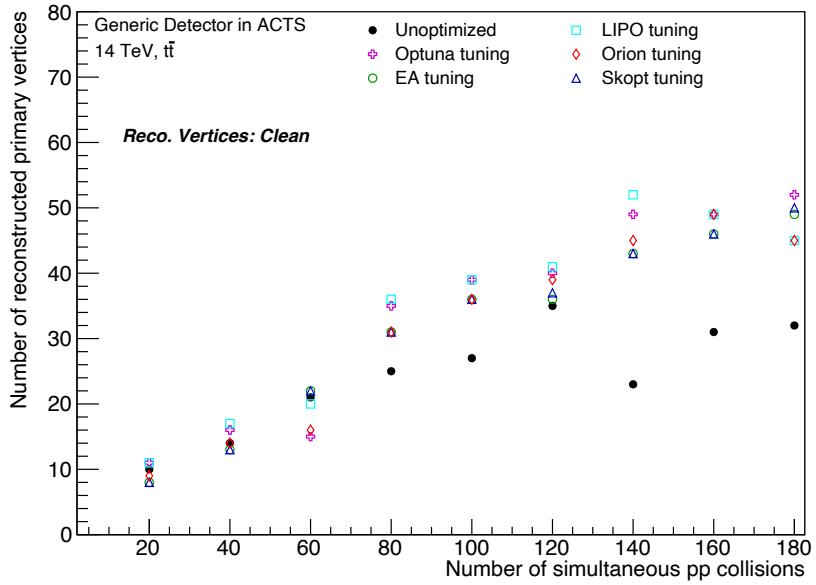
AMVF Optimization Results

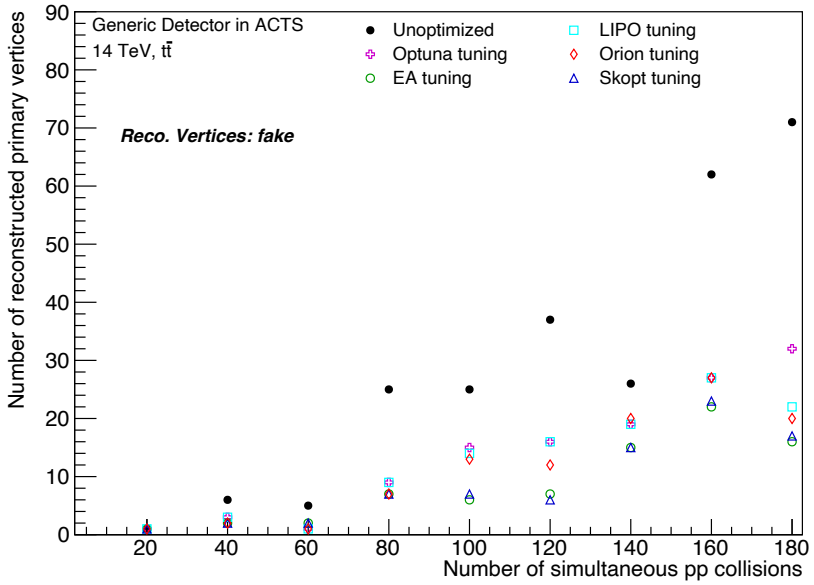
AMVF performance metrics are evaluated and compared for default and optimized parameters!!

Reconstructed Vertices: Total/Clean/Fake

Tuned parameters resulted into more clean vertices and less fake vertices

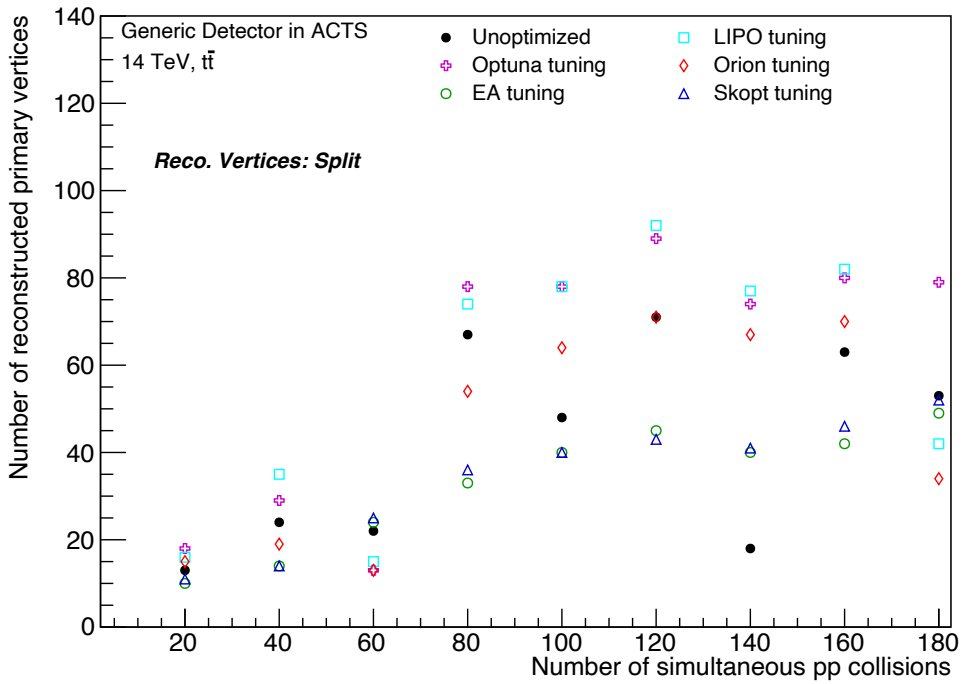


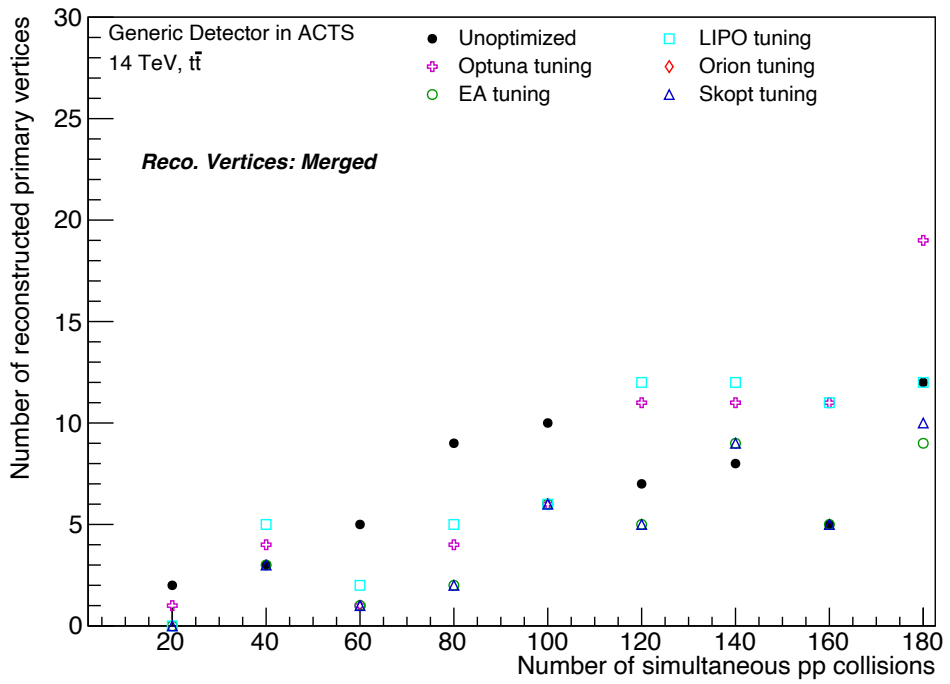




Reconstructed Vertices: Split/Merged

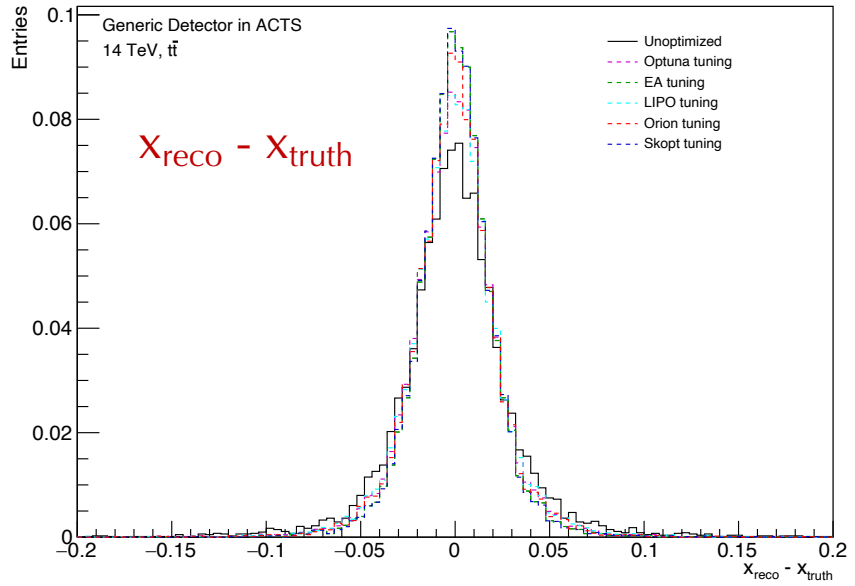
Less Split vertices with tuned parameters from EA and Skopt
Merge vertices already small, remains small with tuned parameters

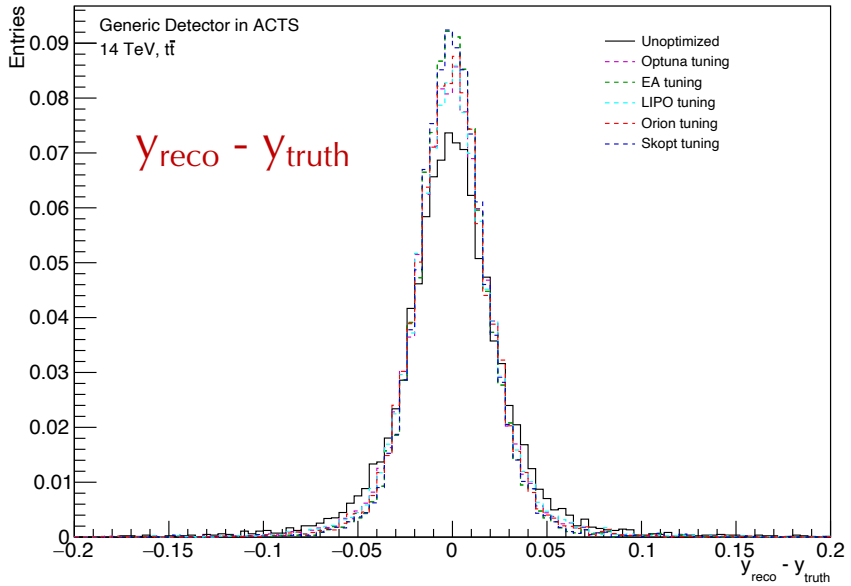


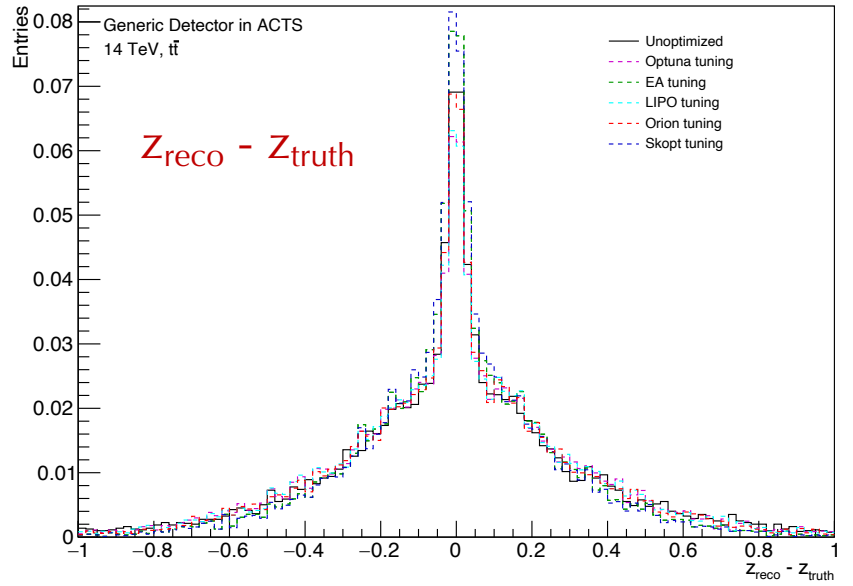


Vertex Resolution in x,y and z

Resolution more centered towards zero for x and y
Not much effect in z



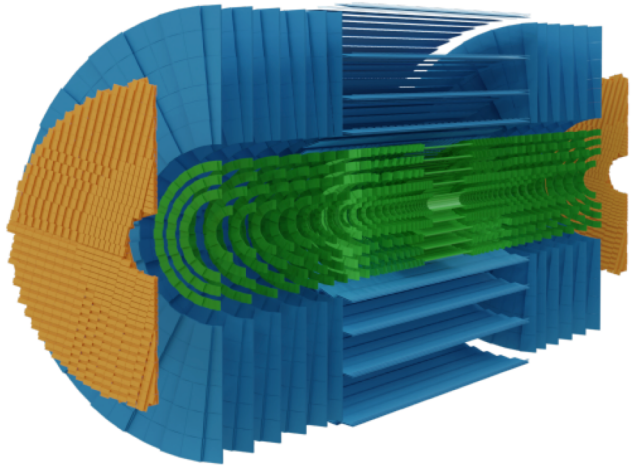




Studies with a Real Detector Geometry: ATLAS ITk

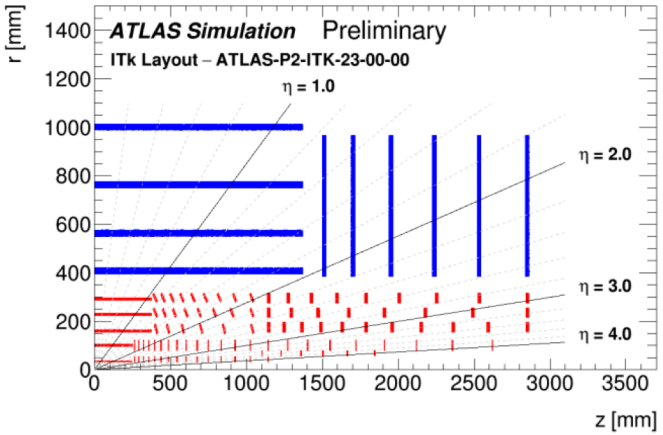
Preliminary Studies with ATLAS-ITk geometry integrated within ACTS

- ✓ New ATLAS ITk geometry plugged-in with ACTS
- ✓ Preliminary studies done using ITk geometry within ACTS for CKF parameters:
  - ✓ Generated ttbar data for pile-up = 200 using Pythia8 and simulated with ITk geometry inside ACTS Framework
  - ✓ Min P_T of Particles > 1 GeV
  - ✓ $|\eta|$ of particles < 4.0



- ✓ Score Function = $Efficiency - FakeRate - \frac{DuplicateRate}{K} - \frac{RunTime}{K}$

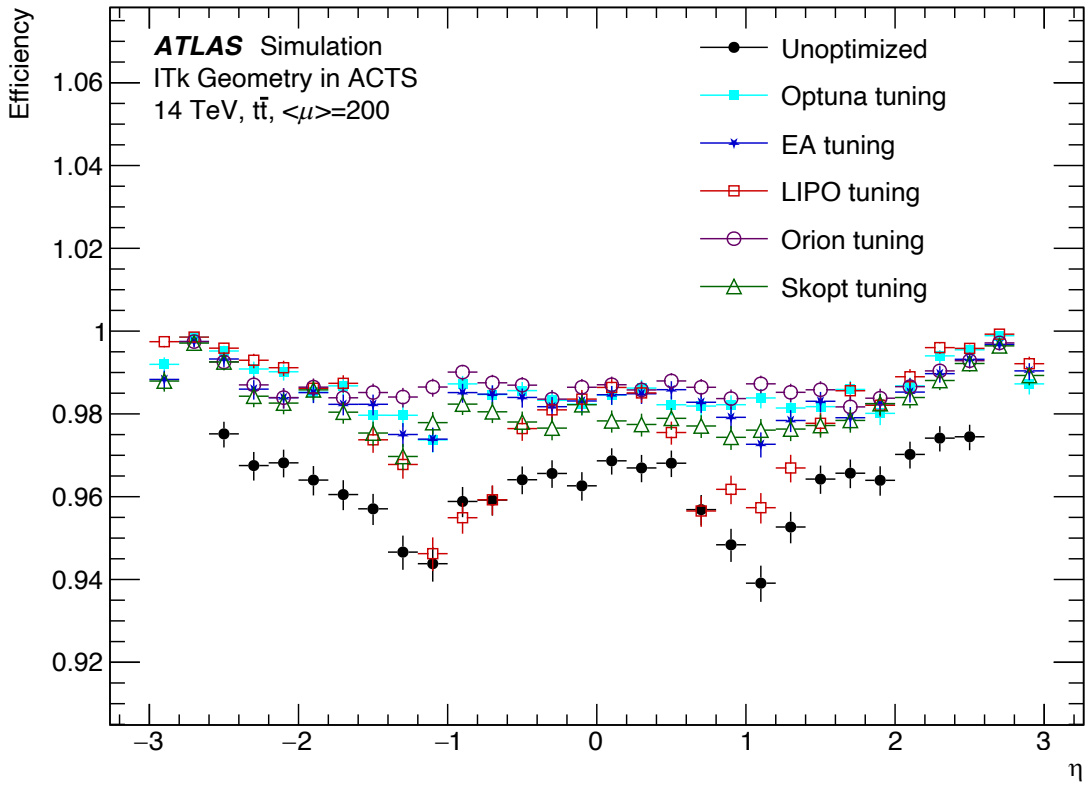
(K = 5 for all algorithms)

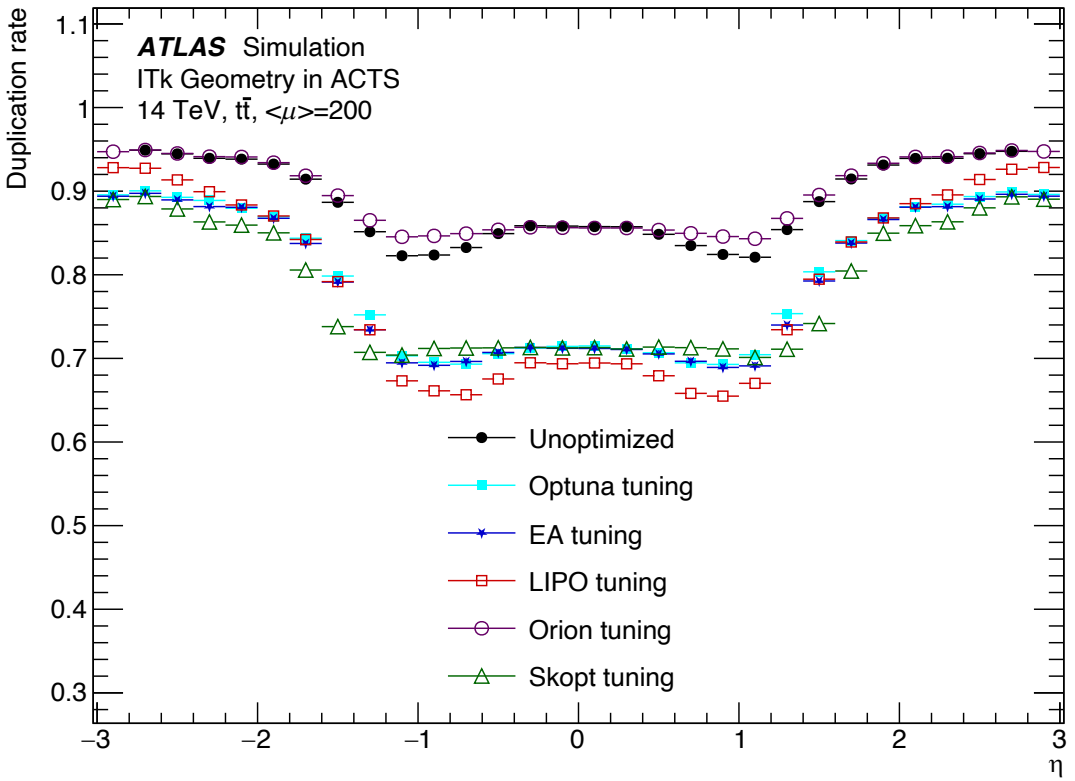


CKF Optimization Results with ATLAS ITk

Track Efficiency and Duplicate Rate vs η

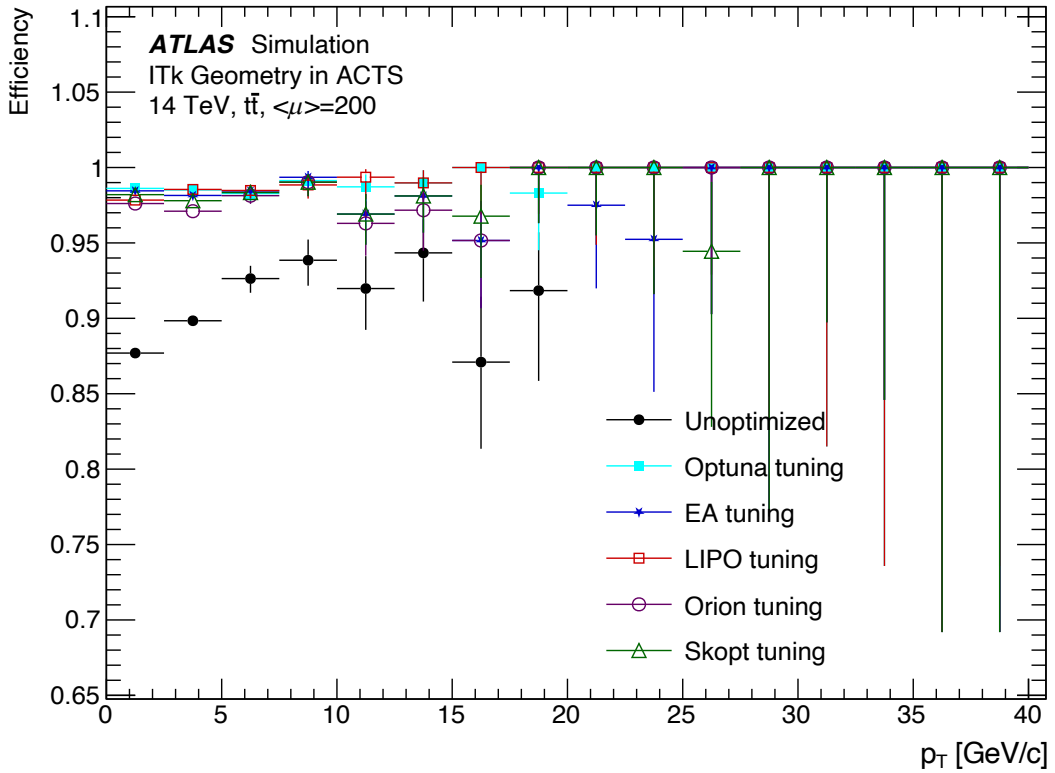
Good improvement in performance with tuned parameters over unoptimized parameters

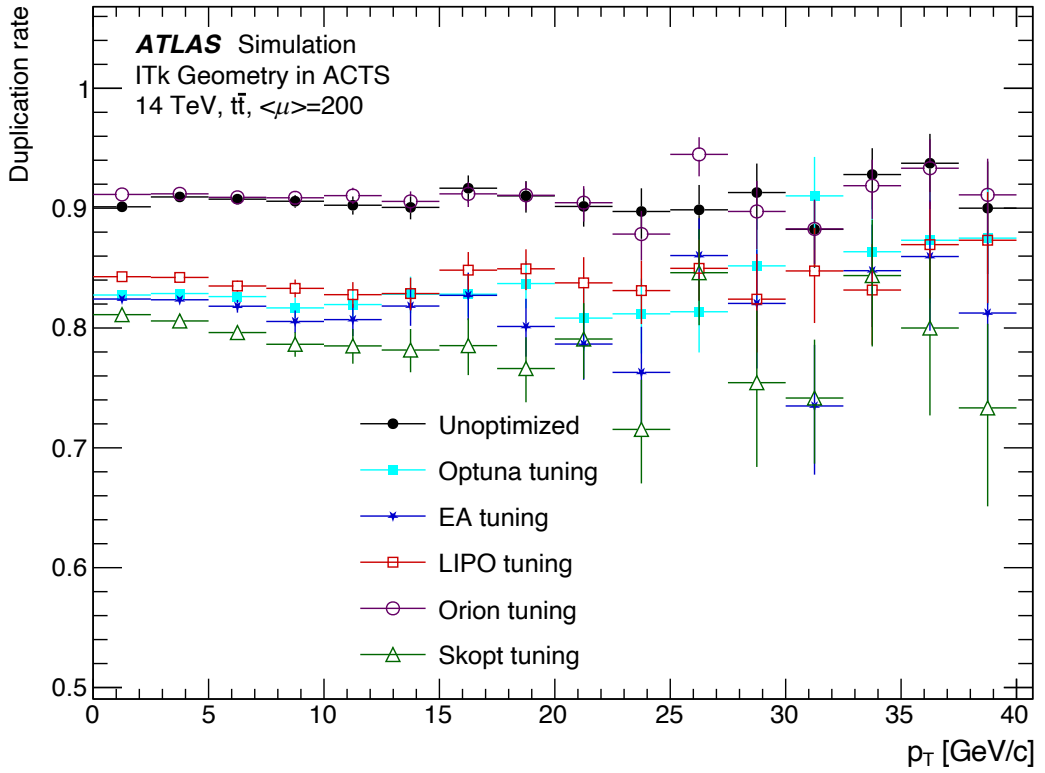




Track Efficiency and Duplicate Rate vs P_T

Good improvement in performance with tuned parameters over unoptimized parameters





Auto-Tuning Scripts and documentation

- ✓ The auto-tuning scripts for Optuna and Orion for seeding algorithm:

  acts/Examples/Scripts/Optimization

- ✓ The documentation explaining how to run these scripts:

  https://acts.readthedocs.io/en/latest/howto/run_ckf_auto_tuning.html

Conclusions

- ✓ Different derivative free optimization methods studied
- ✓ Significant improvement in performance with optimized parameters
- ✓ Optimization algorithms can be tuned as per the need by providing different score functions

Future Work & Plans

- ✓ More work to improve the integration of optimization algorithms with ACTS
- ✓ Explore other possible approaches, if needed
- ✓ Detailed studies with ATLAS ITk geometry
- ✓ Study impact of tuned parameters within ATLAS framework

Back-up

Optimization Algorithms

Evolutionary Algorithms:

- ✓ Based on genetic evolution in living organisms
- ✓ Initializes a population, each individual of population refers to one parameter configuration
- ✓ In each generation, use simultaneous selection and mutation to choose better performing individuals to be kept for next generation
- ✓ Maximizes the score/objective function
- ✓ Link to docs: https://deap.readthedocs.io/en/master/index.html

Optimization Algorithms

Optuna Hyperparameter Tuning Framework:

- ✓ Choose parameter values from the initial range provided
- ✓ Multiple hyperparameter sampling algorithms for parameter selection at each trial are available
- ✓ **Tree-structured Parzen Estimator (TPE):** On each trial, for each parameter, TPE fits one Gaussian Mixture Model (GMM) l(x) to the set of parameter values associated with the best objective values, and another GMM g(x) to the remaining parameter values. It chooses the parameter value x that maximizes the ratio l(x)/g(x)
- ✓ Maximizes the score/objective function
- ✓ Link to docs: https://optuna.org/

Optimization Algorithms

Lipschitz Optimization (LIPO):

- ✓ Do not have its own hyperparameters
- ✓ Maintain a piecewise linear upper bound of score function f(x) and use that to decide which x to evaluate at each step of the optimization
- ✓ Basically picks out points at random, checks if the upper bound for the new point is better than the best point seen so far, and if so selects it as the next point to evaluate
- ✓ Maximizes the score/objective function
- ✓ Link to docs: http://blog.dlib.net/2017/12/a-global-optimization-algorithm-worth.html

Optimization Algorithms

Scikit Optimization:

- ✓ Optimization framework in-built in Python's Sklearn library
- ✓ Includes a number of optimization algorithms
- ✓ Skopt forest_minimize: sequential optimization using decision trees
- ✓ Minimizes the score/objective function
- ✓ Link to docs: https://scikit-optimize.github.io/stable/

Optimization Algorithms

Orion Hyperparameter Tuning Framework:

- ✓ Random search for parameter configurations from uniform priors in the provided range
- ✓ Link to docs: https://orion.readthedocs.io/en/stable/index.html

List of parameters considered for CKF optimization

- ✓ **maxPtScattering:** upper PT limit for scattering angle calculations
- ✓ **impactMax:** maximum value for impact parameter
- ✓ **deltaRMin:** minimum distance in r between two measurements within one seed
- ✓ **deltaRMax:** maximum distance in r between two measurements within one seed
- ✓ **sigmaScattering:** number of sigma used for scattering angle calculations
- ✓ **radLengthPerSeed:** average radiation lengths of material on the length of a seed
- ✓ **maxSeedsPerSpM:** number of space-points in top and bottom layers considered for compatibility with middle space-point
- ✓ **cotThetaMax:** maximum cotTheta between two space-points in a seed to be considered compatible

List of parameters considered for AMVF optimization

- ✓ **tracksMaxZinterval:** maximum z-interval used for adding tracks to multi-vertex fit
- ✓ **maxVertexChi2:** maximum chi2 value for tracks to be compatible with fitted vertex
- ✓ **maxMergeVertexSignificance:** maximum significance on the distance between two vertices to allow merging
- ✓ **minWeight:** minimum track weight for the track to be considered compatible with vertex candidate
- ✓ **maximumVertexContamination:** maximum vertex contamination value

Performance metrics of Tracking algorithms

CKF:

- ✓ Track reconstruction efficiency: fraction of generated particles that have created at least 9 measurements on the traversed detectors and are associated with tracks
- ✓ Fake rate: Fraction of reconstructed tracks not associated with any truth particle
- ✓ Duplicate rate: Fraction of multiple reconstructed tracks associated with same truth generated particle

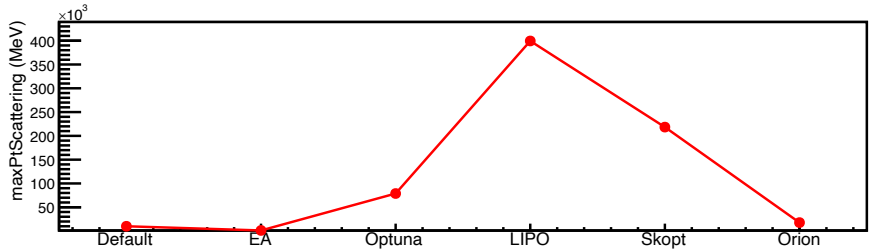
AMVF:

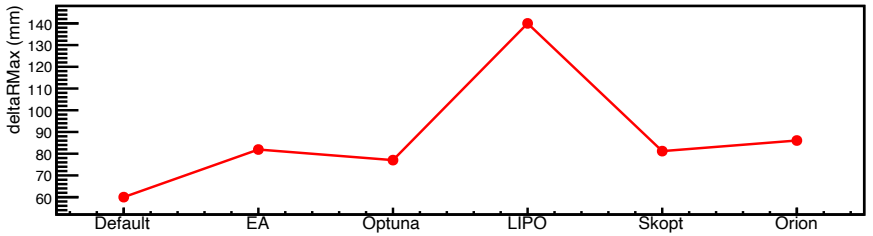
- ✓ Eff_total: No. of vertices reconstructed by AMVF out of total detector accepted vertices
- ✓ Clean: reconstructed vertices associated to one truth generated particle
- ✓ Split: More than one reconstructed vertices associated with same truth particle
- ✓ Merge: One reconstructed vertex associated with more than one truth particle
- ✓ Fake: reconstructed vertices not associated to any truth particle
- ✓ Resolution: deltaR/R = $((reco_x\text{-}true_x)^2 + (reco_y\text{-}true_y)^2 + (reco_z\text{-} true_z)^2)/(true_x^2 + true_y^2 + true_z^2)$

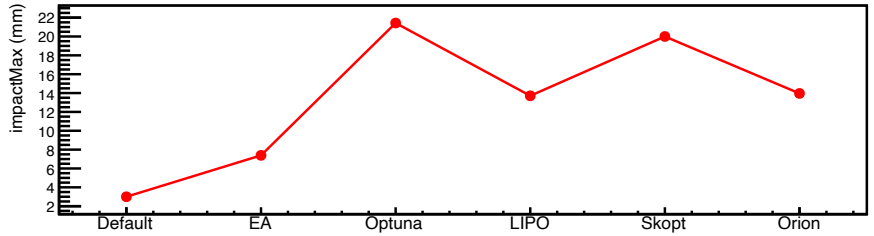
CKF Parameters for Generic Detector: Default and Optimized

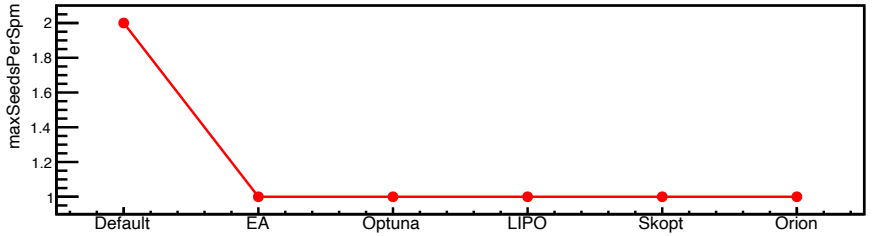
	Default	EA	Optuna	LIPO	Skopt GP	Orion
maxPtScattering (MeV)	10000	1200	78834.5	399441	218413.9	17750
impactMax (mm)	3	7.3899	21.4300	13.7073	20	13.96
deltaRMin (mm)	1	30	12.9795	0.6098	30	23.98
sigmaScattering	50	49.618	4.1008	0.2004	5.7554	5.266
deltaRMax (mm)	60	81.919	77.0430	139.980	81.1772	86.07
maxSeedsPerSpM	1	0	0	0	0	0
radLengthPerSeed	0.1	0.001	0.08154	0.0487	0.1	0.03798
cotThetaMax	7.40627	10	9.3836	8.4210	9.3314	9.068

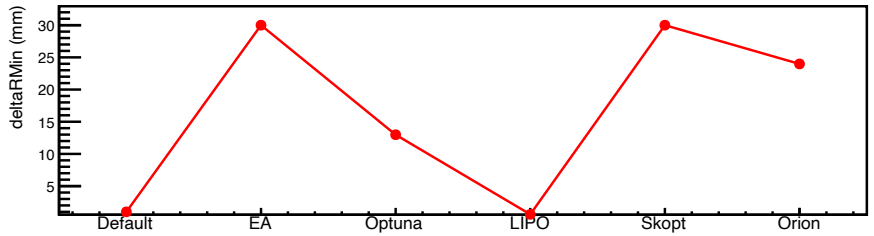
CKF Parameters for Generic Detector: Default and Optimized

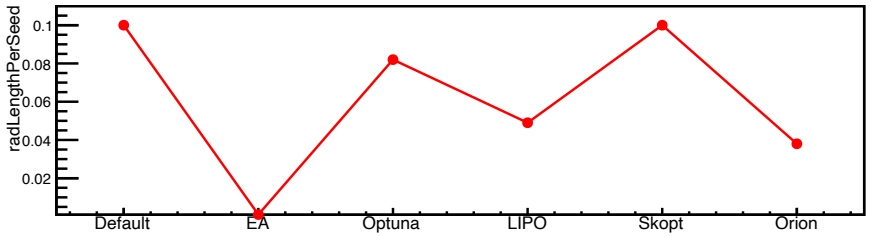


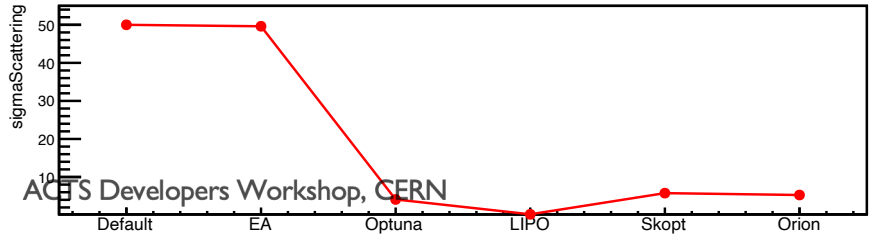


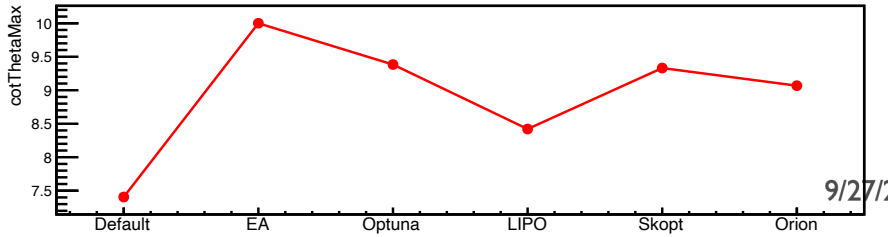








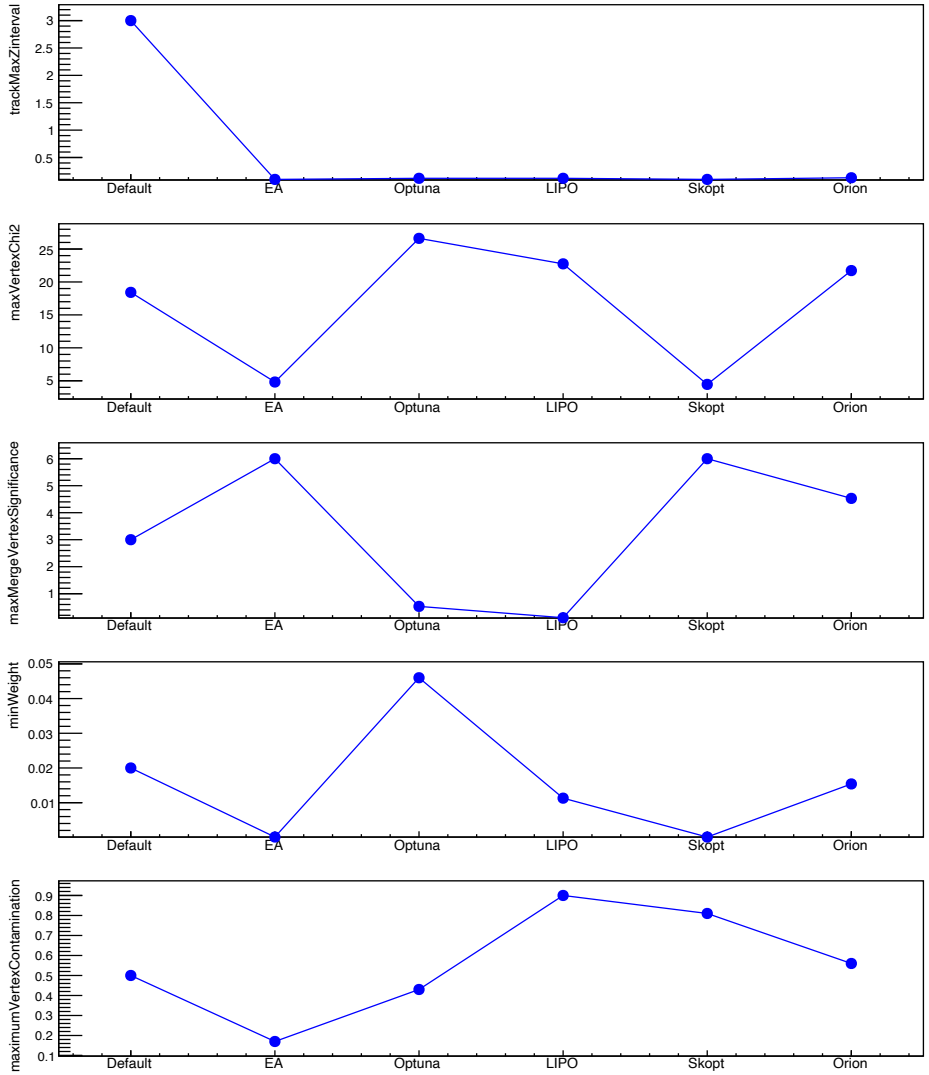




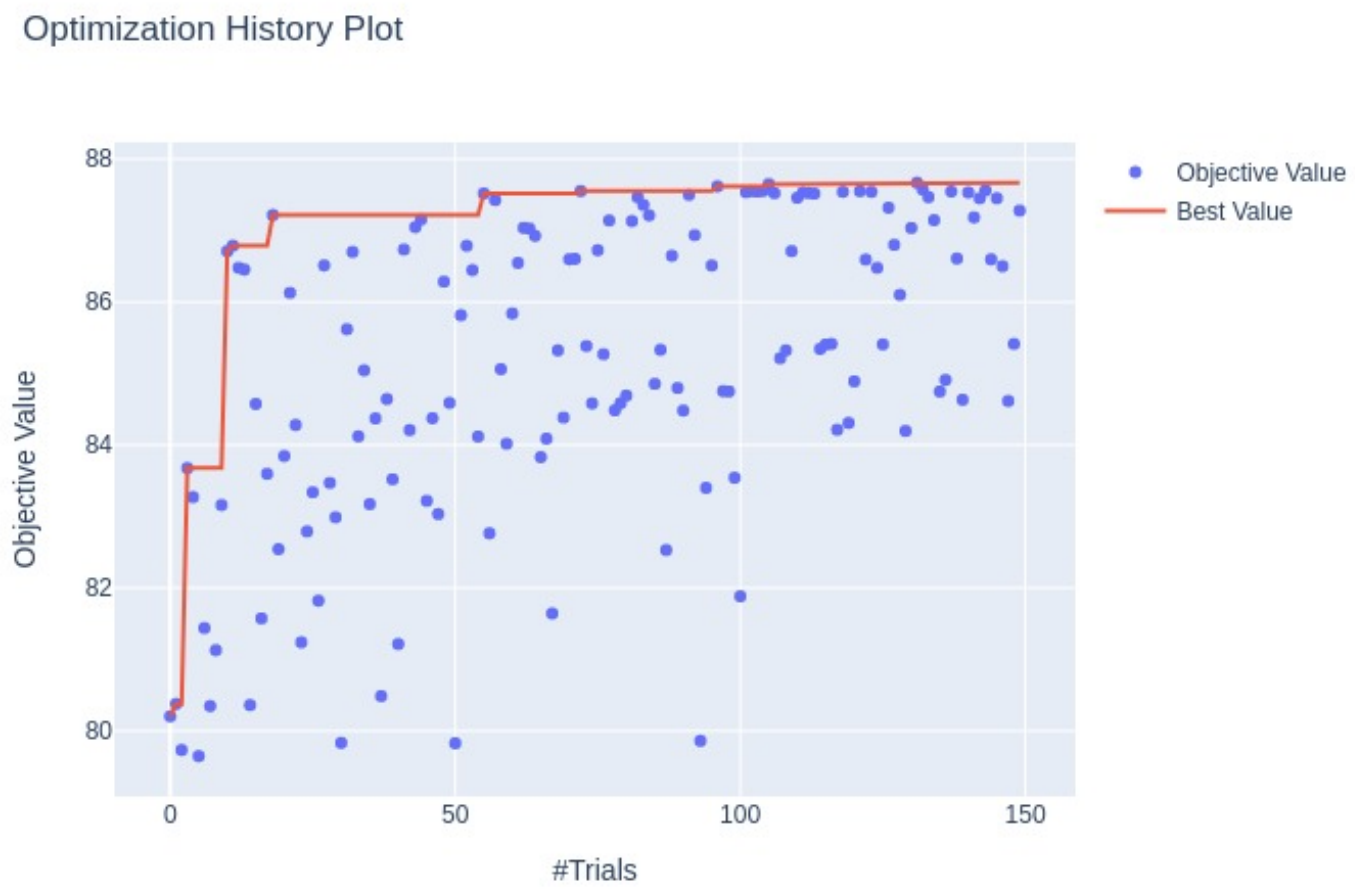
AMVF Parameters: Default and Optimized

	Default	EA	Optuna	LIPO	Skopt	Orion
trackMaxZinterval	3.0	0.1	0.12	0.12	0.1	0.13
maxVertexChi2	18.42	4.81	26.62	22.75	4.45	21.73
maxMergeVertexSignificance	3.0	6.0	0.53	0.11	6.0	4.53
minWeight	0.02	0.0001	0.046	0.0113	0.0001	0.0154
maximumVertexContamination	0.5	0.17	0.43	0.90	0.81	0.56

AMVF Parameters: Default and Optimized

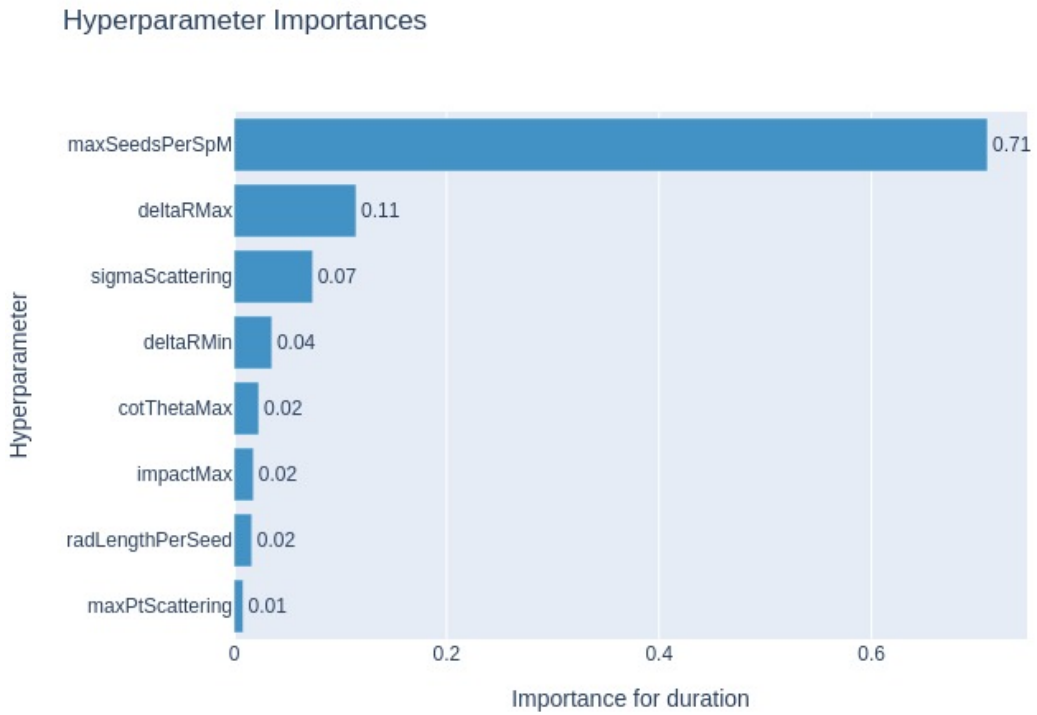


Optuna Performance

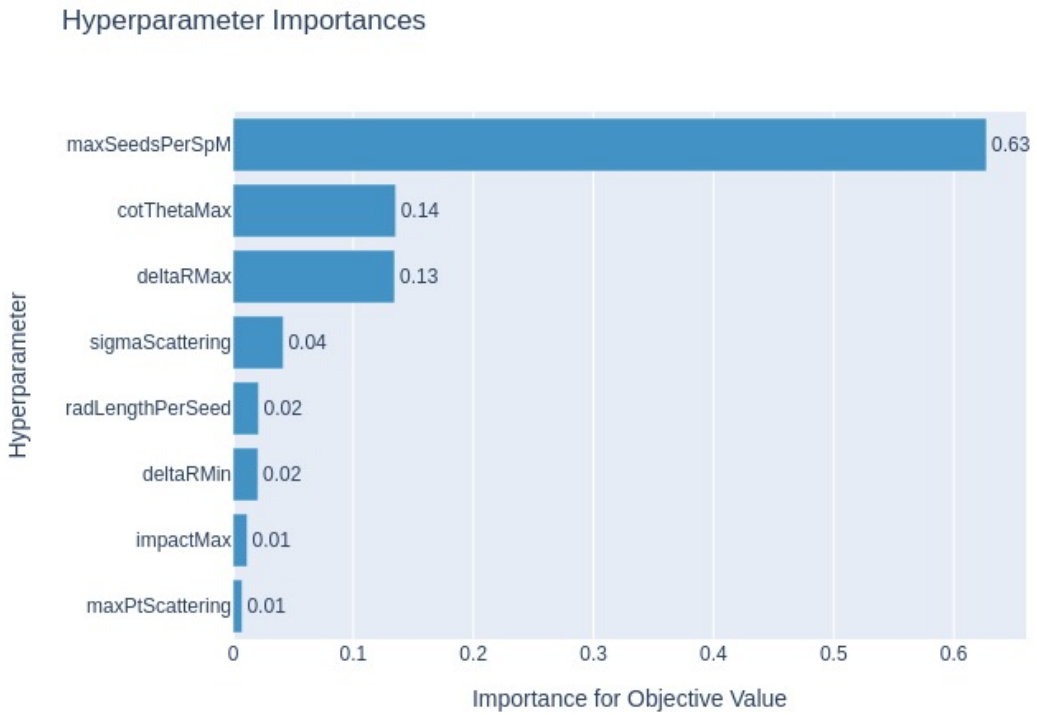


Hyperparameter importance using Optuna

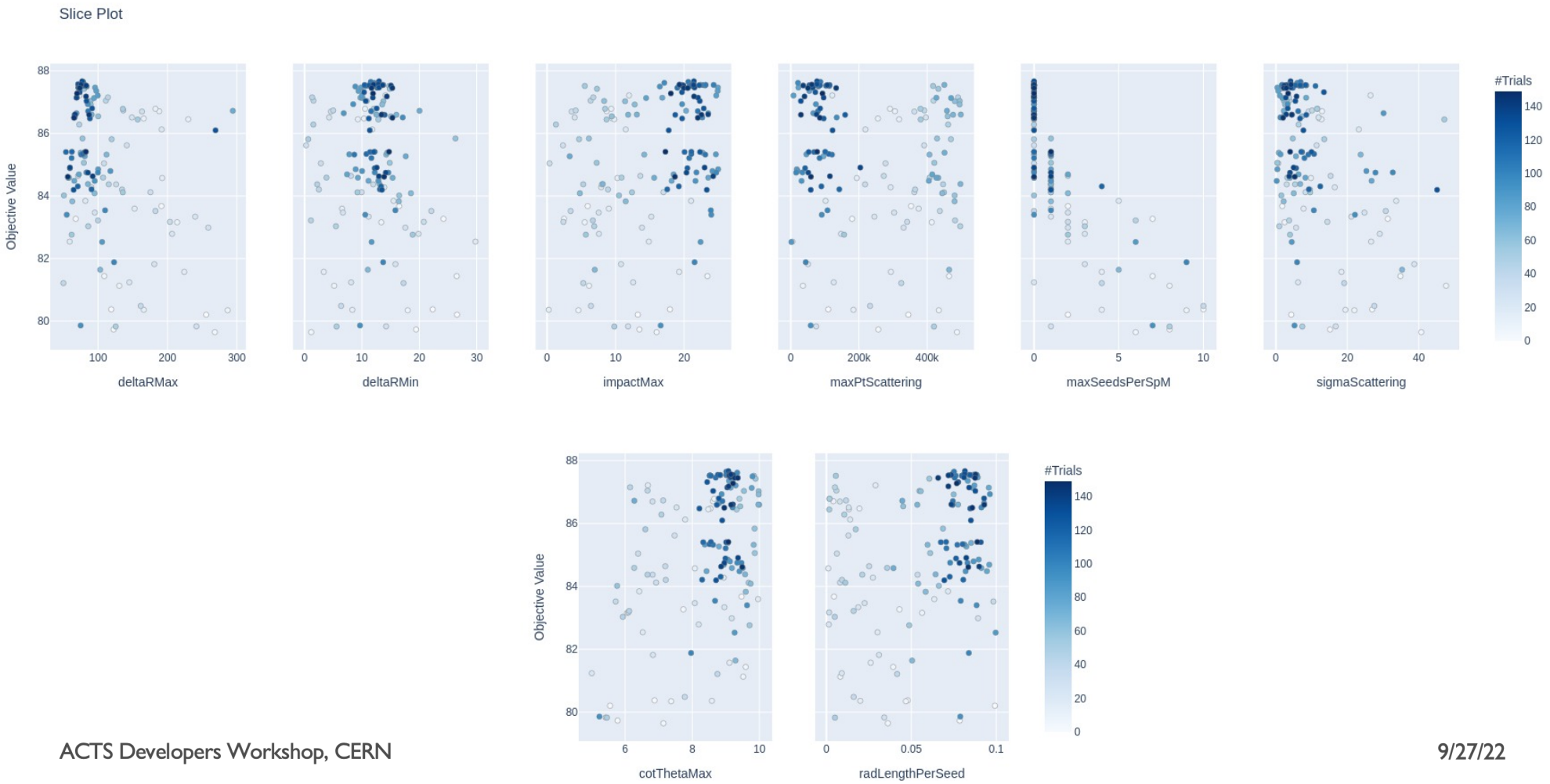
Importance on the basis of run-time



Importance on the basis of score function



Slice plots


Slice Plot
Objective Value
88
86
84
82
80
100
200
300
deltaRMax
0
10
20
30
deltaRMin
0
10
20
impactMax
0
200k
400k
maxPtScattering
0
5
10
maxSeedsPerSpM
0
20
40
sigmaScattering
#Trials
140
120
100
80
60
40
20
0
6
8
10
cotThetaMax
0
0.05
0.1
radLengthPerSeed