Performance optimization for a scintillating glass electromagnetic calorimeter at EIC

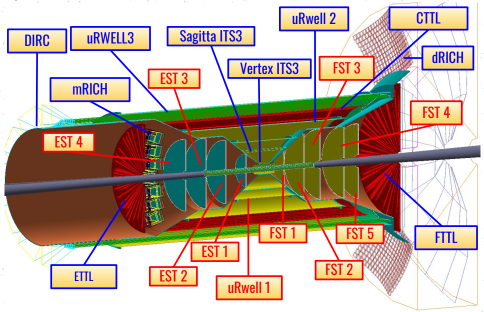
Dmitry Kalinkin for the ePIC collaboration

University of Kentucky

Examples detector optimization at EIC

ECCE Tracker Plane and Disk proportions

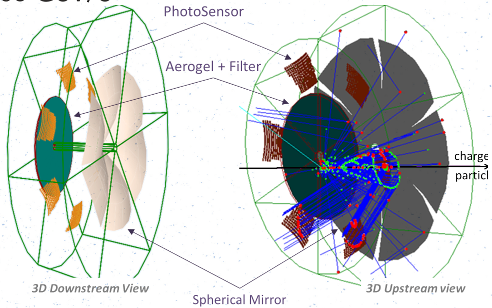
Objectives: KF efficiency, θ and $|\vec{p}|$ resolutions



Multi-Objective Optimization using NSGA-III (see arXiv:2205.09185) and Bayesian MOO (see talk by Karthik Suresh from last year)

dual-radiator RICH

Objective: FOM over $N\sigma_{K\pi}$ at $p = 14$ and 60 GeV/c

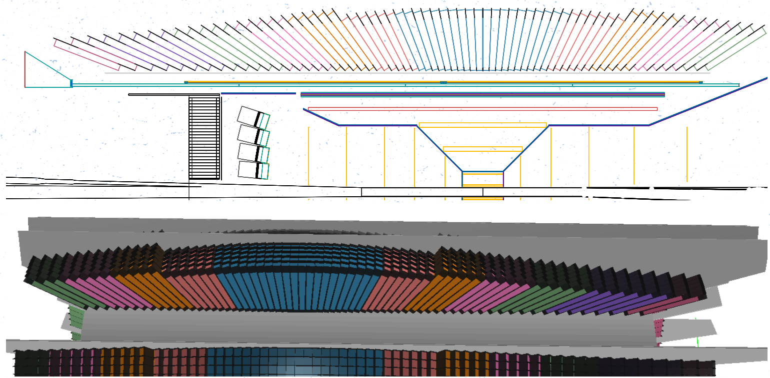


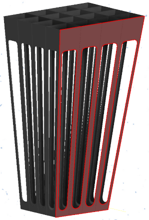
Single-Objective Optimization using BO (see 2020 JINST 15 P05009)

University of Kentucky

SciGlass projective calorimeter

Tower dimensions and placement implemented based on mechanical design: pyramidal towers, angles tweaked by hand

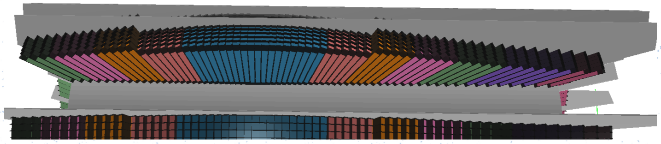


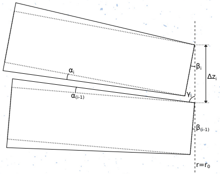


SciGlass lengths of 45.5 and 40 cm ($\approx$ 16.3 and 14.3 X_0)
– originally chosen for a calorimeter that fits inside the BaBar solenoid (Detector I)
Main purpose: measurement of $e^{\pm}/\gamma$ energy, $e^{\pm}/\pi^{\pm}$ and $(\pi^0 \to \gamma\gamma)/\gamma$ discrimination

University of Kentucky

Parametrization





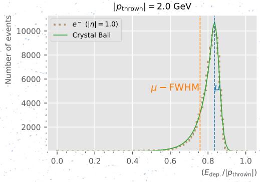
- 7 shapes of cells – “families”
- Families are stacked from $\eta = 0$ in $-\hat{z}$ and $+\hat{z}$ directions
  - 5 integer numbers (0-9) of towers for negative
  - 7 integer numbers (0-9) of towers for positive
- Each shape is a “G4Trap”
  - azimuthal flaring and at-face flaring angles can be calculated to preserve fixed gaps
  - 7 floating point (0.0-2.0 degrees) longitudinal flaring angles determine η projectivity
- Altogether 19 parameters considered
- DD4hep allows detector configuration from “compact” XML files

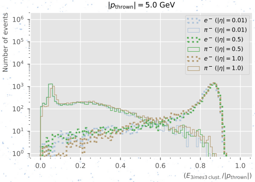
Optimization objectives

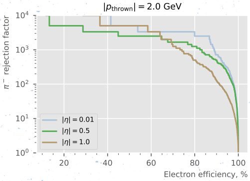
» Single particle simulations: e^-, π^- ($p_T = 2$ GeV)

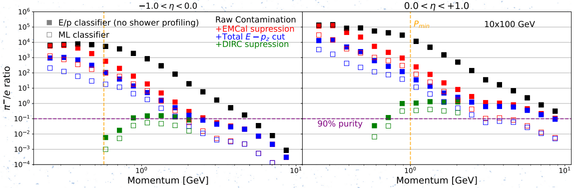
» Key metrics:

- Energy resolution
- Charged pion rejection factor
- Neutral pion to photon discrimination (not considered yet – relies on ML)









University of Kentucky

Multi-objective Optimization using Genetic Algorithms

1 Initialize population (100 samples $\times$ 19 parameters $\leftarrow$ RNG)

2 Evaluate objective functions $\{f_i\}$ for each specimen in the population
    Pick them so that the minimized $f_i \leq 0$, then for invalid geometries use dominated $f_i \equiv N_{\text{overlaps}}$

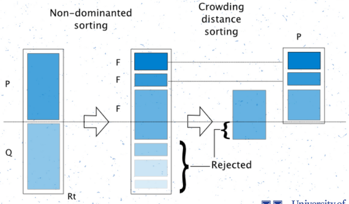
3 Survival, Selection (specified by NSGA-II)

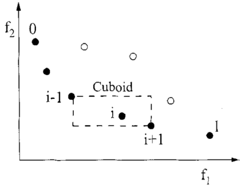
4 Crossover, Mutation (exchange and RNG re-init of individual parameters)

5 Repair (optional)

6 Goto 2

Easy to program with `pymoo`, works out of the box.





Constraints and dimensional reduction

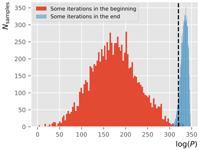
» **Problem:** objective evaluation $\mathcal{O}$(minutes), overlap evaluation $\mathcal{O}$(seconds).
Any way to precompute later for a given geometry?

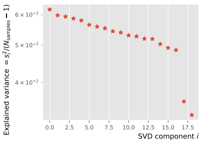
» **Solution:** explore and learn the manifold of valid parameter combinations

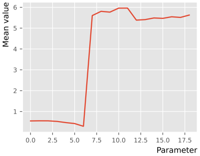
- Could use GA for exploration, but **MCMC** has nice implementations of walkers (we want a "stretch" move)

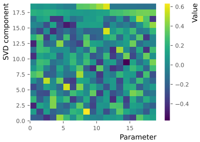
$$\log(P) = \begin{cases} -\infty, \text{ if parameter set doesn't pass overlap check} \\ z_{\text{rightmost tower}} - z_{\text{leftmost tower}} / (1 \text{ cm}), \text{ otherwise} \end{cases}$$

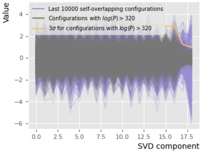
  Run default implementation from `emcee` with $10,000$ walkers for ≈ 1000 iterations
- Almost linear constraints $\Rightarrow$ **PCA** was used to obtain linear transformation and limits (set at 3σ).











2 dimensions effectively removed!

University of Kentucky

Constraints and dimensional reduction

» **Problem:** objective evaluation $\mathcal{O}$(minutes), overlap evaluation $\mathcal{O}$(seconds). Any way to precompute later for a given geometry?

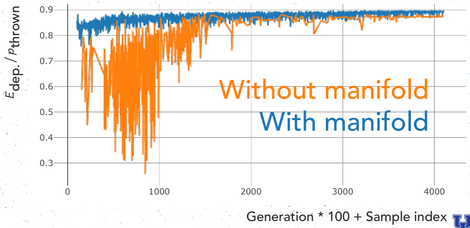
» **Solution:** explore and learn the manifold of valid parameter combinations

- Could use GA for exploration, but **MCMC** has nice implementations of walkers (we want a "stretch" move)

$$\log(P) = \begin{cases} -\infty, & \text{if parameter set doesn't pass overlap check} \\ z_{\text{rightmost tower}} - z_{\text{leftmost tower}} / (1\text{ cm}), & \text{otherwise} \end{cases}$$

  Run default implementation from `emcee` with $10,000$ walkers for ≈ 1000 iterations

- Almost linear constraints $\Rightarrow$ **PCA** was used to obtain linear transformation and limits (set at 3σ).



Nice improvement demonstrated for NSGA!

Bayesian Optimization

Gaussian Process allows for probabilistic surogate modeling

Expected Improvement (EI) acquisition function:

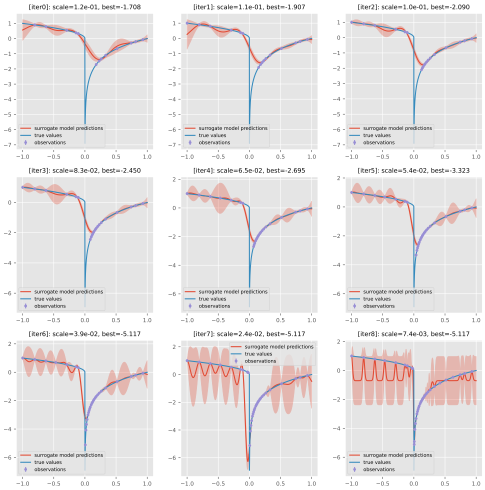
$$\mathbf{EI}(x) = \mathbb{E}[\max(f(x) - f_{\text{best known}}, 0)]$$

Illustration using a toy function:

$$f(x) = \begin{cases} \log(x), \text{ if } x > 0 \\ \sqrt{|x|}, \text{ if } x < 0 \end{cases}$$

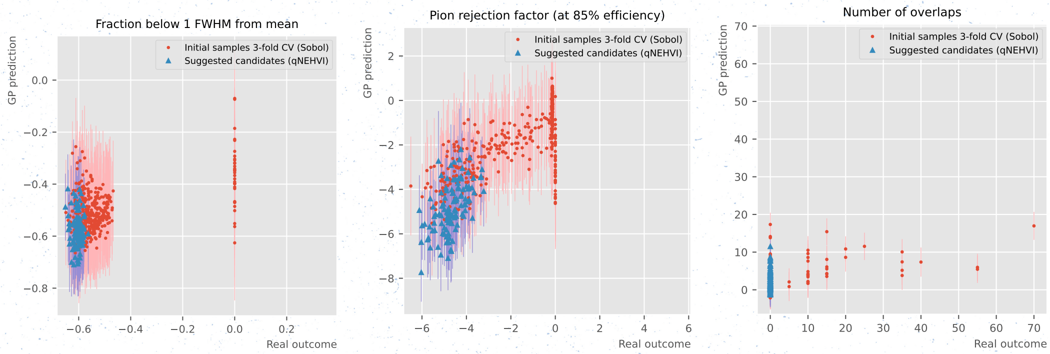
Reporting fake dominated values may not an option – could spoil the fit!

In the MOO case, Expected Improvement is commonly taken for a HyperVolume (qEHVI).



Cross Validation

Important step: cross validation for the surrogate model for a given problem



- Number of overlaps was originally split into a separate "`OutcomeConstraint`" Didn't work well – killed optimization progress.
- Objective value of `0` with an uncertainty is reported for overlapping configurations
- The GP hardcoded for SAASBOO in `Ax` gives a decent CV
- qNEHVI is effective at picking improving points

University of Kentucky

Software stack

pymoo

or

GPyTorch

BoTorch

Ax

Dask.Distributed

dask-jobqueue

Awkward Array

uproot

GEANT4
A SIMULATION TOOLKIT

DD4hep

`epic_arches` geometry

mlflow™

joblib.Memory

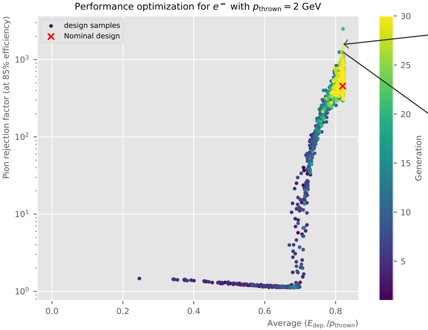
emcee

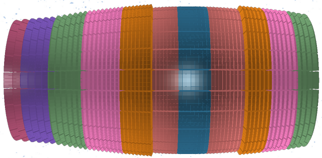
University of Kentucky

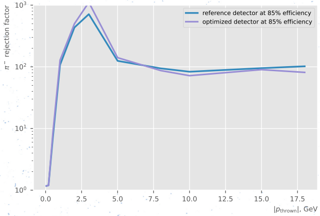
Some results (GA)

2-objective MOO (de-facto single objective)

NSGA-II (100 samples/generation)







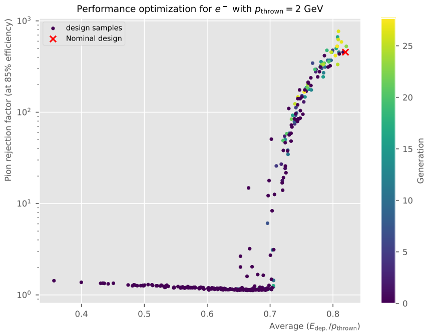
Analysis in full benchmark

University of Kentucky

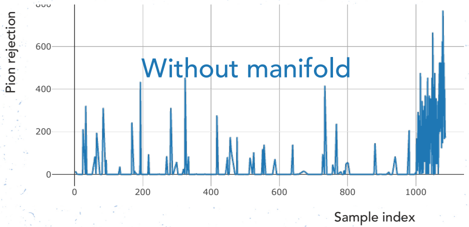
Some results (BO)

2-objective MOO (de-facto single objective)

SAASBO qNEHVI (3 samples/generation)




```
GenerationStrategy(
    steps = [
        GenerationStep(
            model=Models.SOBOL,
            num_trials=1000,
            max_parallelism=1000,
        ),
        GenerationStep(
            model=Models.FULLYBAYESIANMOO,
            num_trials=1000,
            max_parallelism=3,
            model_kwargs=dict(
                num_samples=256,
                warmup_steps=512,
                gp_kernel="rbf",
                torch_device=torch.device("cuda"),
                torch_dtype=torch.double,
                verbose=False,
            ),
        ),
    ],
)
```



Would be interesting to fit solutions from NSGA and see if those can be improved with BO.

University of Kentucky

Summary

» Several working approaches specific to optimization of particle detectors have been demonstrated

» Practical application of SciGlass (e.g. at EIC Detector II) can benefit from ML optimization

- Going beyond pure geometrical optimization
- Proper tooling to scale production and analysis of large optimization workflows
- Are there any issues with objectives involving ML? (for PID)

Acknowledgements

We would thank the University of Kentucky Center for Computational Sciences and Information Technology Services Research Computing for their support and use of the Lipscomb Compute Cluster and associated research computing resources.