

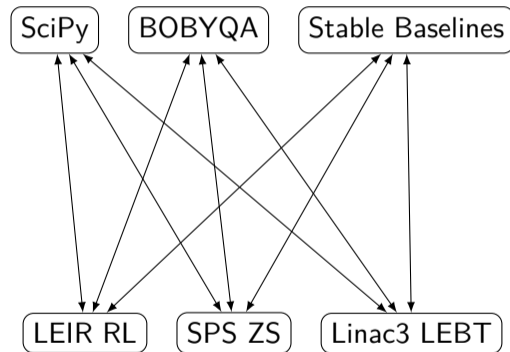
# Generic Optimization and ML at CERN

N. Madysa

December 16, 2021

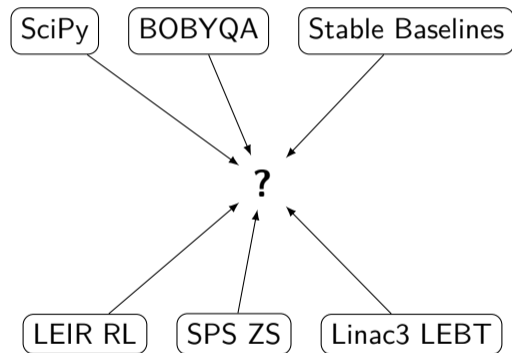
# Motivation

- many different optimization problems at CERN
- many different optimizers with different APIs
- each optimization problem involves complex machine communication
- operators don't want to juggle Python scripts!



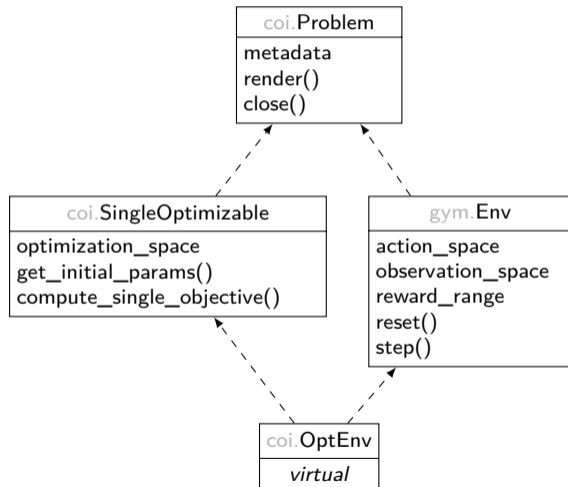
# Motivation

- many different optimization problems at CERN
- many different optimizers with different APIs
- each optimization problem involves complex machine communication
- operators don't want to juggle Python scripts!



# Common Optimization Interfaces (COI)

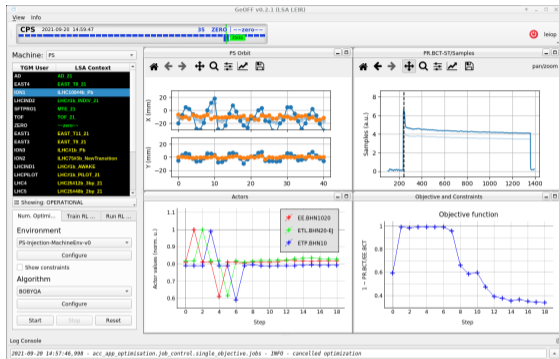
- based on **OpenAI Gym** for RL
- extends the system to numeric optimization
- extend Gym's metadata system with CERN-specific info
  - ▶ which accelerator?
  - ▶ communicates with machines?
  - ▶ wants to plot additional data?



- separate package for faster versioning
- encapsulate many common tasks
- modular thanks to `extras-require`: only provides what is needed

The screenshot shows the documentation page for 'cermi-coi-utils 0.2.3.dev0+ge857768.d20210721'. The page title is 'Utilities for the Common Optimization Interfaces'. It contains a brief introduction, a paragraph about the package's utility functions, and a list of features categorized into 'User Guide', 'API Reference', and 'Changelog'. The 'User Guide' section includes links for 'Installation', 'Managing PyJapc Subscriptions', 'Communicating with the LSA Database', 'Normalizing Parameters', 'Receiving Figures from render()', and 'Keeping Rendering Logic Concise'. The 'API Reference' section includes links for 'PyJapc Utilities', 'PILSA Utilities', 'Gym Utilities', and 'Matplotlib Utilities'. The 'Changelog' section includes links for 'Unreleased', 'v0.2.2', 'v0.2.1', 'v0.2.0', and 'v0.1.0'. The page also features navigation links for 'next', 'modules', and 'index', and a search bar.

# Generic Optimization Frontend & Framework (GeOFF)



- PyQt5-based GUI
- lists, configures and runs optimization problems
- built-in list of optimizers
- very extensible: optimization problems are loaded as plugins (also at runtime!)

- 1 implement class that inherits from COI (~ 300 lines of code)
  - ▶ declare metadata
  - ▶ implement machine communication
  - ▶ add plotting (if any)
- 2 dynamically load package into GeOFF

Optionally:

- 3 turn code into package (provided by Acc-Py)
- 4 add to CERN package index via Gitlab CI (provided by Acc-Py)
- 5 notify us
- 6 get package integrated into GeOFF

- PS: steering from PS to n-TOF
- Linac3: LEBT steering
- LEIR:
  - ▶ transfer line from Linac3
  - ▶ cooler bumps
  - ▶ multi-turn injection
  - ▶ transfer line to PS
- SPS:
  - ▶ tune adjustments
  - ▶ ZS alignment
  - ▶ longitudinal blowup
  - ▶ crystal shadowing
- ISOLDE: generic transfer line optimizer under investigation



# Next steps

- Bayesian optimization
- turn GUI into modular components
  - ▶ provide each part *and* the whole GUI as a library
  - ▶ experiments & machines can package GUI with their own plugins
  - ▶ gives control back to the users, avoids bottleneck
- data recording, automatic model training
- integration with Machine Learning Platform