Xopt: Flexible Black Box Optimization of Simulations and Experiments

Ryan Roussel, Christopher Mayes

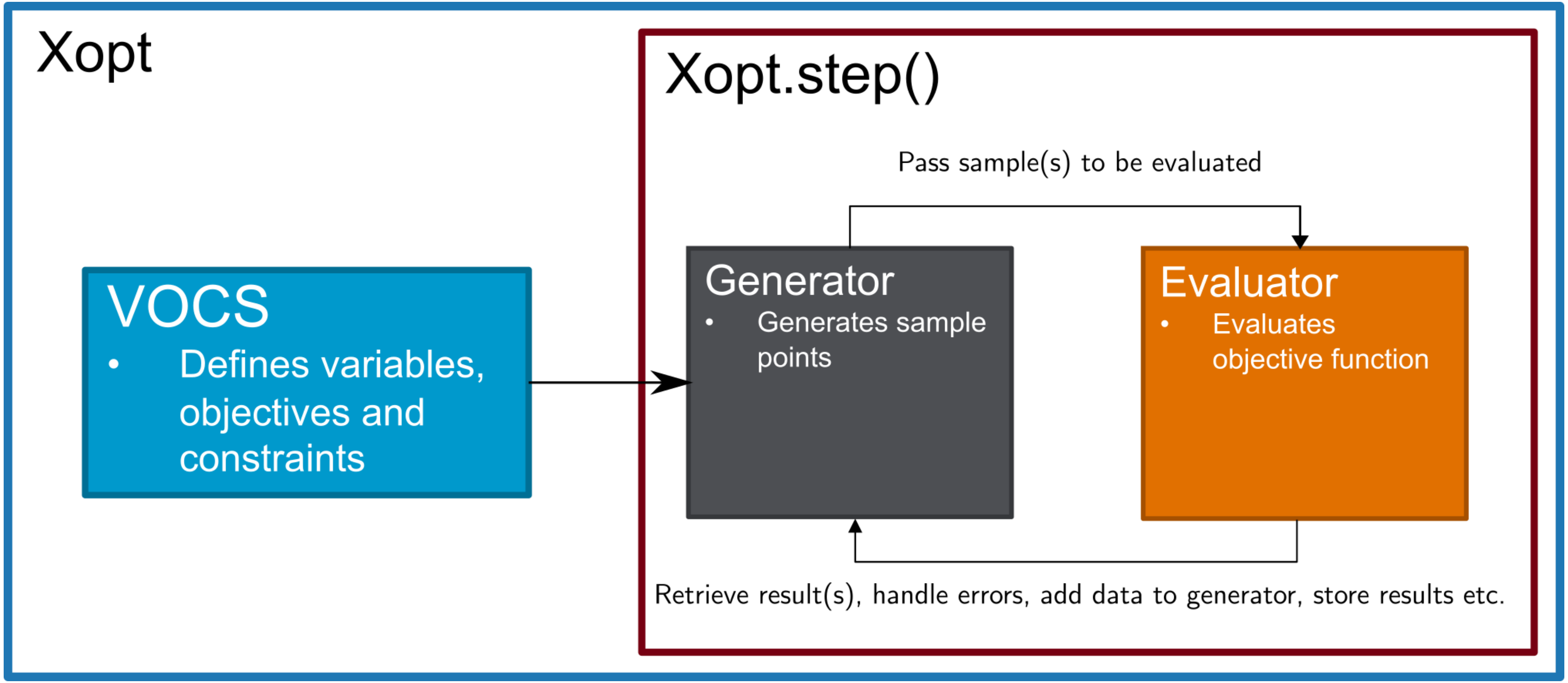
rroussel@slac.stanford.edu, cmayes@slac.stanford.edu

What is Xopt?

- Flexible **framework** for optimization of arbitrary problems using python
- **Independent** of problem type (simulation or experiment)
- **Independent** of optimization algorithm + easy to incorporate custom algorithms
- **Easy to use** text interface and/or advanced customized use for professionals

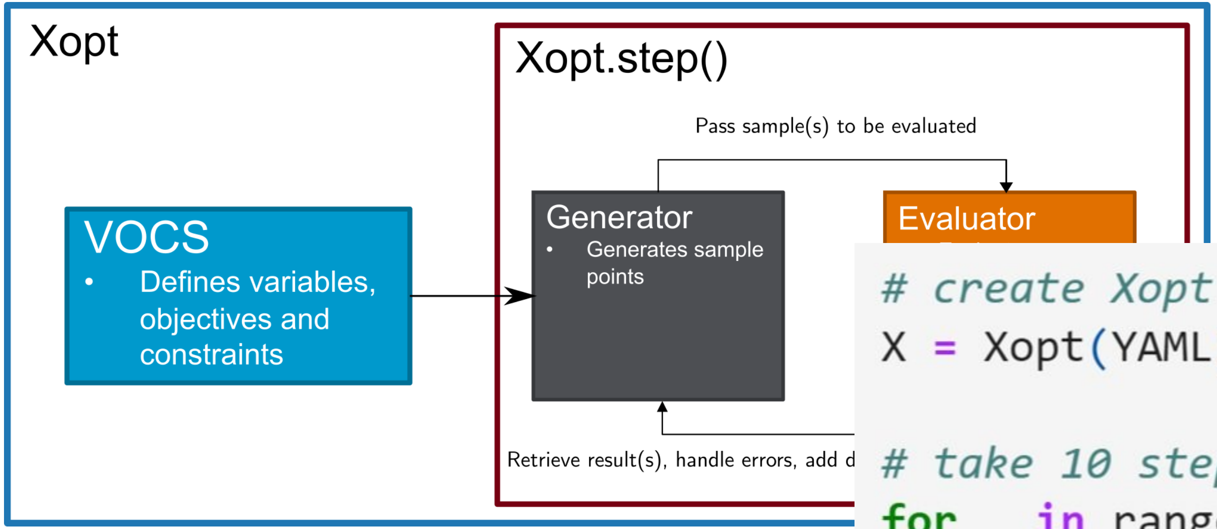
https://github.com/ChristopherMayes/Xopt

Xopt structure



Note: this process can also be done asynchronously

Xopt usage




```python
# create Xopt object.
X = Xopt(YAML)

# take 10 steps and view data
for _ in range(10):
    X.step()

X.data
```

Xopt input

Via YAML file (validated by pydantic):

```yaml
xopt:
    max_evaluations: 6400

generator:
    name: cnsga
    population_size: 64
    population_file: test.csv
    output_path: .

evaluator:
    function: xopt.resources.test_functions.tnk.evaluate_TNK
    function_kwargs:
      raise_probability: 0.1

vocs:
    variables:
        x1: [0, 3.14159]
        x2: [0, 3.14159]
    objectives: {y1: MINIMIZE, y2: MINIMIZE}
    constraints:
        c1: [GREATER_THAN, 0]
        c2: [LESS_THAN, 0.5]
    linked_variables: {x9: x1}
    constants: {a: dummy_constant}
```

Via python code:

```python
evaluator = Evaluator(…)
generator = CNSGAGenerator(…)
vocs = MyVOCS(…)

X = Xopt(
            evaluator=evaluator,
            generator=generator,
            vocs=vocs
)
```

Evaluator specification

- Python function must accept/return dicts
- Input dict must have **at least** the keys specified in vocs variables/constants (see next slide)
  - You can include extra keyword args if needed!
- Output dict must have **at least** the keys specified in objectives/constraints (see next slide)
  - The function can output extra keys to be tracked!
- Functions can be defined at the module level and passed via string if they are in PYTHONPATH, they can also be passed inside the same python file (use __main__.my_function)
- Evaluators inherit directly from python concurrent.futures so you can use this for parallel evaluation (see /xopt/docs/examples/basic/xopt_parallel)

```
evaluate(inputs: dict) -> dict
```

```yaml
xopt:
    max_evaluations: 6400

generator:
    nam
    popu
    popu
    output_path: .

evaluator:
    function: xopt.resources.test_functions.tnk.evaluate_TNK
    function_kwargs:
      raise_probability: 0.1

vocs:
    variables:
        x1: [0, 3.14159]
        x2: [0, 3.14159]
    objectives: {y1: MINIMIZE, y2: MINIMIZE}
    constraints:
        c1: [GREATER_THAN, 0]
        c2: [LESS_THAN, 0.5]
    linked_variables: {x9: x1}
    constants: {a: dummy_constant}
```

Evaluate function

- Python function must accept/return dicts
- Input dict must have **at least** the keys specified in vocs variables/constants (see next slide)
  - You can include extra keyword args if needed!
- Output dict must have **at least** the keys specified in objectives/constraints (see next slide)
  - The function can output extra keys to be tracked!

```
evaluate(inputs: dict) -> dict
```

```python
from epics import caget, caput, cainfo
import time

outputs = ["XRMS","YRMS"]
def make_epics_measurement(input_dict):
  # set inputs
  for name, val in input_dict.items():
    caput(name, val)

  # wait for inputs to settle
  time.sleep(1)

  # get output values, current time
  output_dict = caget_many(outputs)
  output_dict["time"] = time.time()

  # compute geometeric avg of beamsizes
  output_dict["RMS"] = (
    output_dict["XRMS"]*\
    output_dict["YRMS"]
  )**0.5

  return output_dict
```

VOCS Specification

- Variables: input domain limits and names
- Objectives: objective names and goals (minimize/maximize)
- Constraints: constraint names and conditions (greater than/less than)
- Constants: constant values

```yaml
xopt:
    max_evaluations: 6400

generator:
    name: cnsga
    population_size: 64
    population_file: test.csv
    output_path: .

evaluator:
    function: xopt.resources.test_functions.tnk.evaluate_TNK
    function_kwargs:
      raise_probability: 0.1

vocs:
    variables:
        x1: [0, 3.14159]
        x2: [0, 3.14159]
    objectives: {y1: MINIMIZE, y2: MINIMIZE}
    constraints:
        c1: [GREATER_THAN, 0]
        c2: [LESS_THAN, 0.5]
    linked_variables: {x9: x1}
    constants: {a: dummy_constant}
```

Generator specification

- Use built-in generators by name

- optimization algorithms:
  - `cnsga` Continuous NSGA-II with constraints.
  - `bayesian_optimization` Single objective Bayesian optimization (w/ or w/o constraints, serial or parallel).
  - `mobo` Multi-objective Bayesian optimization (w/ or w/o constraints, serial or parallel).
  - `bayesian_exploration` Bayesian exploration.
- sampling algorithms:
  - `random sampler`

- Each generator has its own specific options
- Locate the default options in the docs or via

```python
from xopt.utils import get_generator_and_defaults
gen, options = get_generator_and_defaults("upper_confidence_bound")
print(yaml.dump(options.dict()))
```

```yaml
acq:
  beta: 2.0
  monte_carlo_samples: 512
  proximal_lengthscales: null
model:
  use_conservative_prior_lengthscale: false
  use_conservative_prior_mean: false
  use_low_noise_prior: false
n_initial: 3
optim:
  num_restarts: 5
  raw_samples: 20
  sequential: true
```

```yaml
xopt:
    max_evaluations: 6400

generator:
    name: cnsga
    population_size: 64
    population_file: test.csv
    output_path: .

evaluator:
    function: xopt.resources.test_functions.tnk.evaluate_TNK
    function_kwargs:
      raise_probability: 0.1

vocs:
    variables:
        x1: [0, 3.14159]
        x2: [0, 3.14159]
    objectives: {y1: MINIMIZE, y2: MINIMIZE}
    constraints:
        c1: [GREATER_THAN, 0]
        c2: [LESS_THAN, 0.5]
    linked_variables: {x9: x1}
    constants: {a: dummy_constant}
```

Data storage

- Data is stored by xopt in the `data` attribute
- Set dump_file in xopt options to dump data and xopt config to yaml file after every evaluation step
- Dump file can be used to restart xopt

```python
# view the data
X.data
```

	x1	x2	y1	y2	c1	c2	some_array	xopt_error	xopt_error_str	a
1	1.000000	0.750000	1.000000	0.750000	0.626888	0.312500	[1, 2, 3]	False		NaN
2	0.750000	1.000000	0.750000	1.000000	0.626888	0.312500	[1, 2, 3]	False		NaN
3	0.796389	0.807321	0.796389	0.807321	0.186596	0.182292	[1, 2, 3]	False		dummy_constant
4	0.871085	0.943368	0.871085	0.943368	0.568348	0.334279	[1, 2, 3]	False		dummy_constant
5	1.067732	0.797750	1.067732	0.797750	0.843056	0.410974	[1, 2, 3]	False		dummy_constant
6	0.995019	0.879029	0.995019	0.879029	0.707805	0.388707	[1, 2, 3]	False		dummy_constant
7	0.803822	1.022336	0.803822	1.022336	0.724145	0.365142	[1, 2, 3]	False		dummy_constant
8	0.656282	0.952071	0.656282	0.952071	0.434474	0.228792	[1, 2, 3]	False		dummy_constant
9	0.566763	0.935263	0.566763	0.935263	0.271920	0.193911	[1, 2, 3]	False		dummy_constant
10	0.547152	1.008562	0.547152	1.008562	0.326474	0.260859	[1, 2, 3]	False		dummy_constant
11	0.617813	1.081140	0.617813	1.081140	0.594283	0.351603	[1, 2, 3]	False		dummy_constant
12	0.491363	1.027666	0.491363	1.027666	0.231751	0.278506	[1, 2, 3]	False		dummy_constant

```yaml
xopt:
    dump_file: dump.yaml
```

In 3
```python
config = yaml.safe_load(open("dump.yaml"))
X2 = Xopt(config)
print(X2.options)
print(X2.generator)
print(X2.evaluator)
```

Tips and Tricks

SLAC

- **Look at the examples in docs/examples** !!!!
- Get creative with the evaluate function to track variables/outputs.
- Ask for invite to #xopt channel
- Always looking for help!

main Xopt / docs / examples /

ChristopherMayes Add 5-dim Roesnbrock example

..

basic	hand-execute parallel notebook
bayes_exp	executed notebooks
bayes_opt	executed notebooks
cnsga	add scipy docs
developer	Algorithm to generator example
scipy	Add 5-dim Roesnbrock example

Example Application: LCLS FEL Power Characterization

- **Proximal biasing** to reduce exploration step size and **constraints** to prevent charge loss.
- **Custom evaluate function** captures 80th percentile FEL power over 100 shots.
- Data stored in Pandas DataFrame objects, exported to text file with Xopt configuration
- FEL sensitivity is captured in the GP model lengthscales inside the generator object.
- Entirely executed from an **interactive Jupyter notebook.**