ParMOO: A Python library for parallel multiobjective simulation optimization

Tyler Chang[a] and Stefan Wild[a,b]

[a]Mathematics and Computer Science Division,
Argonne National Laboratory

[b]Applied Mathematics and Computational Research Division,
Lawrence Berkeley National Laboratory

SIAM CSE 23

Outlines

Introduction to MOSO + my experience

3 challenges + solutions

ParMOO software design + release

Example Problems

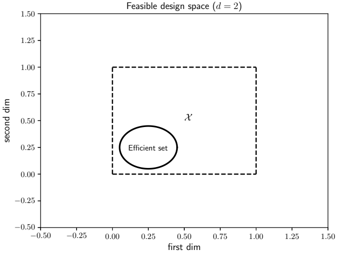
Conclusions + some closing thoughts

Multiobjective Optimization Problems

$$\min_{x \in \mathcal{X}} F(x)$$

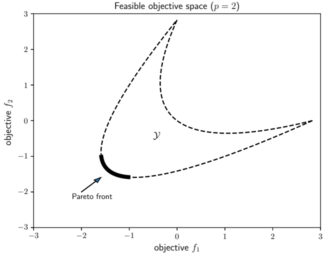
Multiobjective Optimization Problems

$$\min_{x \in \mathcal{X}} F(x)$$



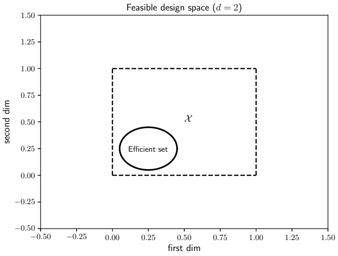
$$\longrightarrow$$

$$F : \mathcal{X} \to \mathcal{Y}$$



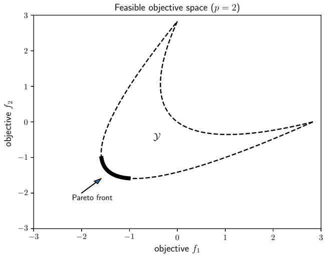
Multiobjective Optimization Problems

$$\min_{x \in \mathcal{X}} F(x)$$



$$F : \mathcal{X} \to \mathcal{Y}$$

expensive
blackbox process

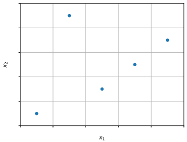


Multiobjective Response Surface Methodology

or Model-Based Optimization or Active Learning

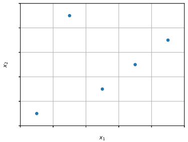
Multiobjective Response Surface Methodology

or Model-Based Optimization or Active Learning



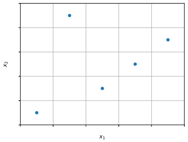
Multiobjective Response Surface Methodology

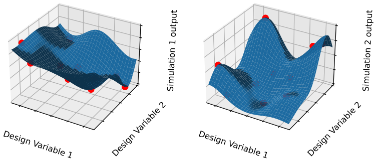
or Model-Based Optimization or Active Learning



Multiobjective Response Surface Methodology

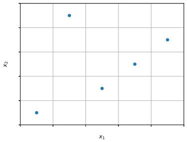
or Model-Based Optimization or Active Learning

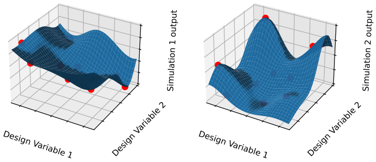


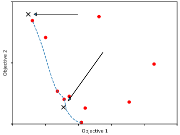


Multiobjective Response Surface Methodology

or Model-Based Optimization or Active Learning

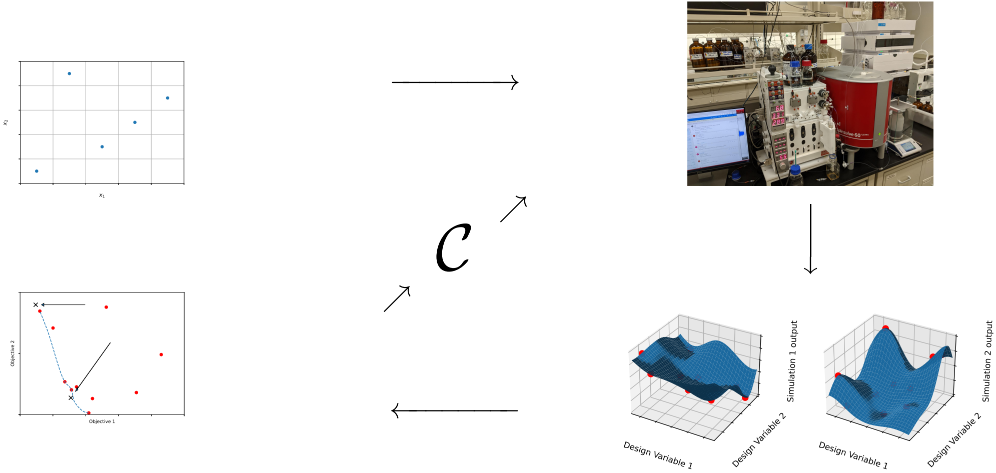




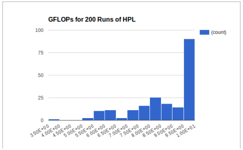


Multiobjective Response Surface Methodology

or Model-Based Optimization or Active Learning



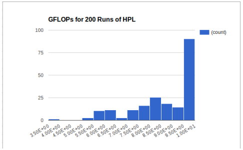
Example: HPC Performance Tuning



VT VarSys Project – 40 runs of HPL

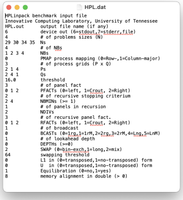
ANL – LCRC Computing Resources: Bebop

Example: HPC Performance Tuning



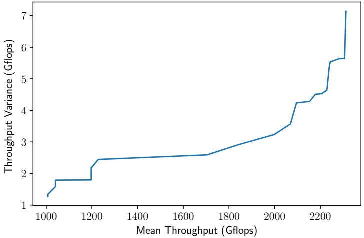
VT VarSys Project – 40 runs of HPL

ANL – LCRC Computing Resources: Bebop



```
HPL.dat
HPLinpack benchmark input file
Innovative Computing Laboratory, University of Tennessee
HPL.out      output file name (if any)
6            device out (6=stdout,7=stderr,file)
4            # of problems sizes (N)
29 30 34 35  Ns
4            # of NBs
1 2 3 4      NBs
0            PMAP process mapping (0=Row-,1=Column-major)
3            # of process grids (P x Q)
2 1 4        Ps
2 4 1        Qs
16.0         threshold
3            # of panel fact
0 1 2        PFACTs (0=left, 1=Crout, 2=Right)
2            # of recursive stopping criterium
2 4          NBMINs (>= 1)
1            # of panels in recursion
2            NDIVs
3            # of recursive panel fact.
0 1 2        RFACTs (0=left, 1=Crout, 2=Right)
1            # of broadcast
0            BCASTs (0=1rg,1=1rM,2=2rg,3=2rM,4=Lng,5=LnM)
1            # of lookahead depth
0            DEPTHs (>=0)
2            SWAP (0=bin-exch,1=long,2=mix)
64           swapping threshold
0            L1 in (0=transposed,1=no-transposed) form
0            U  in (0=transposed,1=no-transposed) form
1            Equilibration (0=no,1=yes)
8            memory alignment in double (> 0)
```

VTMOP solver $\longrightarrow$

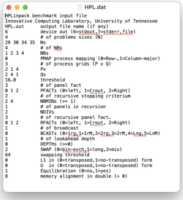


[1] Chang, Larson, and Watson. Multiobjective optimization of the variability of the high-performance LINPACK solver. In Proc. WSC 2020.

Challenge 1:
mixed vars + problem types

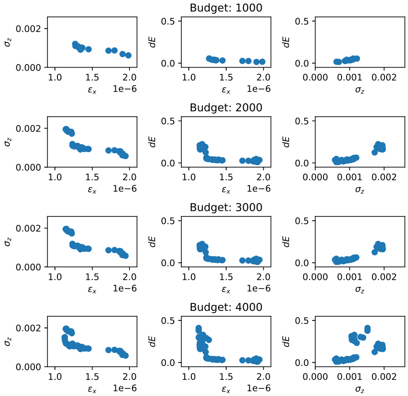
Example: Particle Accelerator Design

OPAL-t
(Object Oriented Parallel Accelerator Library for beam-lines + linacs)



```
HPL.dat
HPLinpack benchmark input file
Innovative Computing Laboratory, University of Tennessee
HPL.out      output file name (if any)
6            device out (6=stdout,7=stderr,file)
4            # of problems sizes (N)
29 30 34 35  Ns
4            # of NBs
1 2 3 4      NBs
0            PMAP process mapping (0=Row-,1=Column-major)
3            # of process grids (P x Q)
2 1 4        Ps
2 4 1        Qs
16.0         threshold
3            # of panel fact
0 1 2        PFACTs (0=left, 1=Crout, 2=Right)
2            # of recursive stopping criterium
2 4          NBMINs (>= 1)
1            # of panels in recursion
2            NDIVs
3            # of recursive panel fact.
0 1 2        RFACTs (0=left, 1=Crout, 2=Right)
1            # of broadcast
0            BCASTs (0=1rg,1=1rM,2=2rg,3=2rM,4=Lng,5=LnM)
1            # of lookahead depth
0            DEPTHs (>=0)
2            SWAP (0=bin-exch,1=long,2=mix)
64           swapping threshold
0            L1 in (0=transposed,1=no-transposed) form
0            U  in (0=transposed,1=no-transposed) form
1            Equilibration (0=no,1=yes)
8            memory alignment in double (> 0)
```

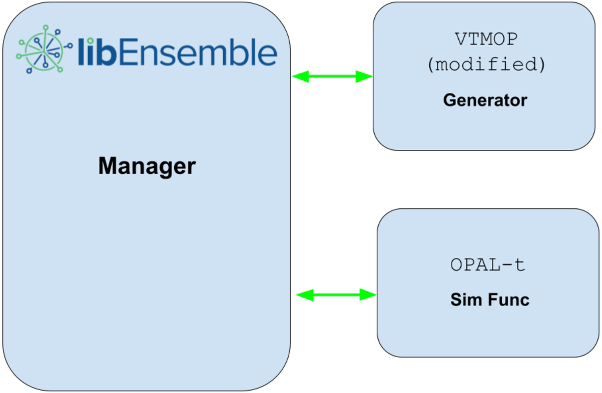
VTMOP solver



[2] Chang et al. Algorithm 1028: VTMOP: Solver for blackbox multiobjective optimization problems. ACM TOMS 48(3):36 (2022).

[3] Neveu et al. Comparison of multiobjective optimization methods for the LCLS-II photoinjector. CPC 283:108566 (2023).

Handling parallel evals



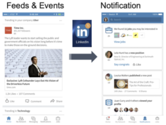
[4] *Chang et al. Managing computationally expensive blackbox multiobjective optimization problems with libEnsemble. In Proc. SpringSim 2020.*

Challenge 2:

parallel evals + computing environments

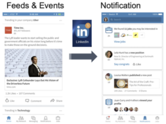
Commercial solutions

Commercial solutions

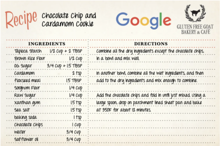


"Using Bayesian optimization for balancing metrics in recommendation systems" by Yunbao Ouyang et al. on LinkedIn Engineering Blog.

Commercial solutions

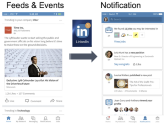


"Using Bayesian optimization for balancing metrics in recommendation systems" by Yunbao Ouyang et al. on LinkedIn Engineering Blog.

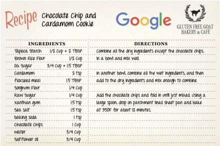


"The makings of a smart cookie" by Daniel Golovin on Google Research Blog.

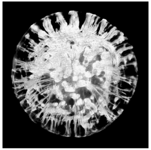
Commercial solutions



"Using Bayesian optimization for balancing metrics in recommendation systems" by Yunbao Ouyang et al. on LinkedIn Engineering Blog.

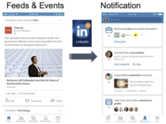


"The makings of a smart cookie" by Daniel Golovin on Google Research Blog.

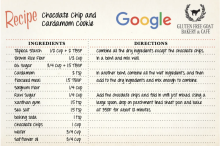


"Accelerating molecular optimization with AI" by Payel Das et al. on IBM Research Blog.

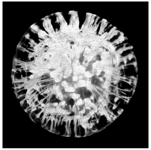
Commercial solutions



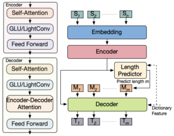
“Using Bayesian optimization for balancing metrics in recommendation systems” by Yunbao Ouyang et al. on LinkedIn Engineering Blog.



“The makings of a smart cookie” by Daniel Golovin on Google Research Blog.

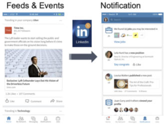


“Accelerating molecular optimization with AI” by Payel Das et al. on IBM Research Blog.

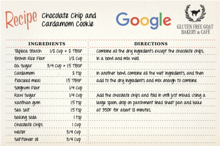


“Optimizing model accuracy and latency using Bayesian multi-objective NAS” by David Eriksson et al. on Meta Research Blog.

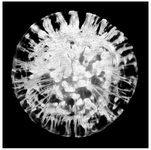
Commercial solutions



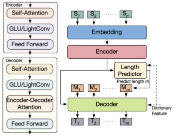
“Using Bayesian optimization for balancing metrics in recommendation systems” by Yunbao Ouyang et al. on LinkedIn Engineering Blog.



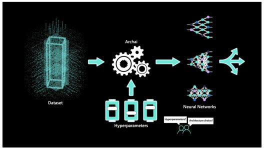
“The makings of a smart cookie” by Daniel Golovin on Google Research Blog.



“Accelerating molecular optimization with AI” by Payel Das et al. on IBM Research Blog.



“Optimizing model accuracy and latency using Bayesian multi-objective NAS” by David Eriksson et al. on Meta Research Blog.



“Archai can design your neural network with state-of-the-art NAS” by Shital Shah et al. on Microsoft Research Blog.

Commercial solvers

Commercial solvers

General purpose: (solver + backend)

Google – OSS Vizier + Pythia backend

[5] Song et al. OSS Vizier: distributed infrastructure and API for reliable and flexible black-box optimization. In Proc. 2022 AutoML-Conf.

Meta – BoTorch + Ax backend

[6] Balandat et al. BoTorch: a framework for efficient monte-carlo Bayesian optimization. In NeurIPS 2020.

Commercial solvers

General purpose: (solver + backend)

Google – OSS Vizier + Pythia backend

[5] Song et al. OSS Vizier: distributed infrastructure and API for reliable and flexible black-box optimization. In Proc. 2022 AutoML-Conf.

Meta – BoTorch + Ax backend

[6] Balandat et al. BoTorch: a framework for efficient monte-carlo Bayesian optimization. In NeurIPS 2020.

Special purpose: (solver + special purpose deployment)

IBM – Querry-based Molecular Optimization (QMO)

[7] Hoffman et al. Optimizing molecules using efficient queries from property evaluations. Nature Machine Intelligence 4:21–31 (2022).

Microsoft – Archai for NAS

[8] Shah et al. Archai: platform for neural architecture search. Microsoft Research (Jul, 2022).

SOA and structure-exploiting blackbox optimization

SOA and structure-exploiting blackbox optimization

"Optimization and root finding (scipy.optimize)" in SciPy v1.10.0 [9].

[9] Virtanen et al. SciPy 1.0: fundamental algorithms for scientific computing in Python. Nature Methods 17:261–272 (2020).

SOA and structure-exploiting blackbox optimization

“Optimization and root finding (scipy.optimize)” in SciPy v1.10.0 [9].


siam news

Optimization and Learning with Zeroth-order Stochastic Oracles

Stochastic dimension reduction explained in this context by Stefan [10].

[9] Virtanen et al. SciPy 1.0: fundamental algorithms for scientific computing in Python. Nature Methods 17:261–272 (2020).

[10] Wild. Optimization and learning with zeroth-order stochastic oracles. SIAM News 56(1):1,3 (2023).

SOA and structure-exploiting blackbox optimization

“Optimization and root finding (scipy.optimize)” in SciPy v1.10.0 [9].


siam news

Optimization and Learning with Zeroth-order Stochastic Oracles

Stochastic dimension reduction explained in this context by Stefan [10].

SOS structure can be exploited by DFO solver POUNDERS in TAO [11].

[9] Virtanen et al. SciPy 1.0: fundamental algorithms for scientific computing in Python. Nature Methods 17:261–272 (2020).

[10] Wild. Optimization and learning with zeroth-order stochastic oracles. SIAM News 56(1):1,3 (2023).

[11] Wild. Solving derivative-free nonlinear least squares problems with POUNDERS. Adv. and Trends in Optimization with Engineering Applications, Ch. 40, pp. 529–540 (2017).

Challenge 3:

SOA optimization + exploiting problem structure

ParMOO Design Criteria

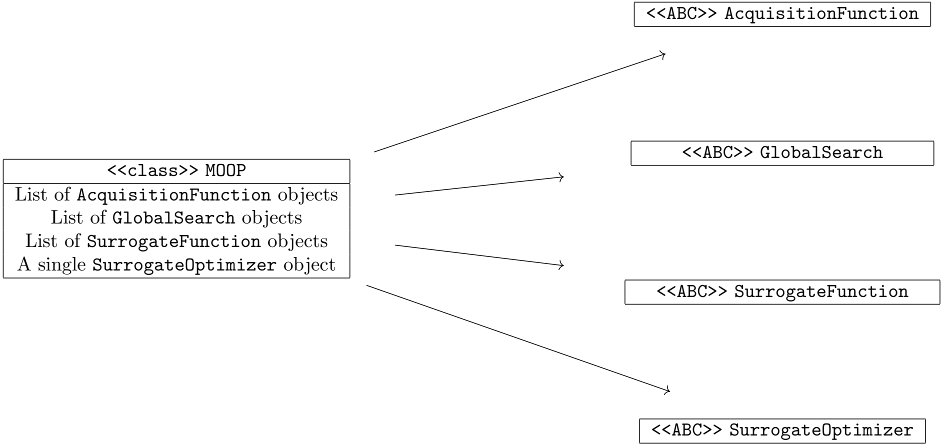
Design goals:

1. Highly customizable framework for multiobjective RSM
2. Flexible problem types (mixed-variables, constraints, etc.)
3. Easy to use, deploy, and extend (unforeseen use-cases and environments)
4. Solve large-scale problems + exploit structure and domain knowledge

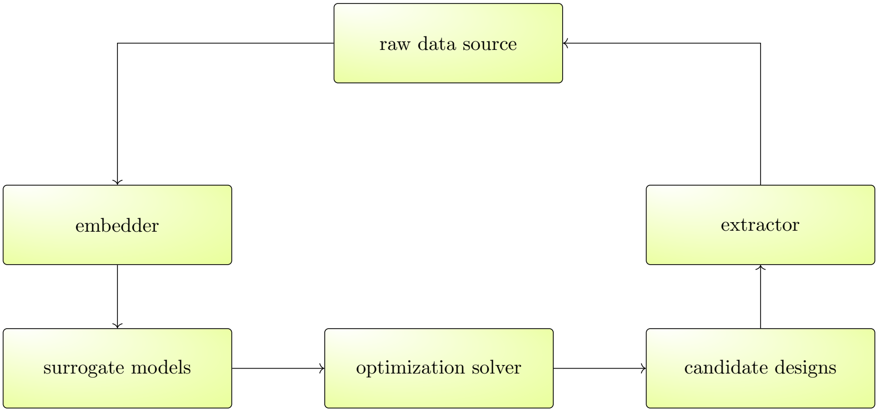
[12] Chang and Wild. Designing a framework for solving multiobjective simulation optimization problems. In prep.

Goal 1: Customizability

ParMOO UML:



Goal 2: Flexible problem types



Goal 3: Easy to deploy

Extend `MOOP` base class and overwrite `MOOP.evaluateSimulation()` evaluator backend.

Goal 3: Easy to deploy

Extend `MOOP` base class and overwrite `MOOP.evaluateSimulation()` evaluator backend.

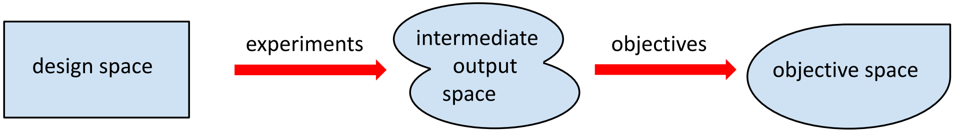
Examples:

- parallel simulation evaluations on HPC systems with libEnsemble [13]
- streaming experiment data via Kafka producer/consumer requests with the MDML [14]

[13] Hudson et al. libEnsemble: a library to coordinate the concurrent evaluation of dynamic ensembles of calculations. IEEE TPDS 33(4):977–988 (2021).

[14] Elias et al. The manufacturing data and machine learning platform: enabling real-time monitoring and control of scientific experiments via IoT. In Proc. 2020 IEEE WF-IoT.

Goal 4: problem structure



Simulation Structure

Simulation Structure

$$f_i(x) = h_i(x, S(x)) \qquad i = 1, \ldots, o$$

Simulation Structure

$$f_i(x) = h_i(x, S(x)) \qquad i = 1, \ldots, o$$

Sum-of-squares structure:

$$h_i(x, S(x)) = \sum_{j \in N_i} \left(S_j(x)\right)^2$$

where each $N_1, \ldots, N_o$ is an index set.

Increases order of approximation $\Rightarrow$
increases order of convergence

Simulation Structure

$$f_i(x) = h_i(x, S(x)) \qquad i = 1, \ldots, o$$

Sum-of-squares structure:

$$h_i(x, S(x)) = \sum_{j \in N_i} \left(S_j(x)\right)^2$$

where each $N_1, \ldots, N_o$ is an index set.

Increases order of approximation $\Rightarrow$ increases order of convergence

Heterogeneous MOOPs:

$$h_1(x, S(x)) = S_1(x)$$
$$h_2(x, S(x)) = \|x\|^2$$

Use expensive surrogate models for h_1 (i.e., S_1) but not for h_2

Sample code

```
# Create MOOP
my_moop = MOOP(LocalGPS)
# Add a continuous design variable
my_moop.addDesign({'name': "x1", 'des_type': "continuous",
                   'lb': 0.0, 'ub': 1.0})
# Add a categorical design variable
my_moop.addDesign({'name': "x2", 'des_type': "categorical", 'levels':
    3})
# Define a simulation function
def sim_func(x):
    if x["x2"] == 0:
        return np.array([(x["x1"] - 0.2) ** 2, (x["x1"] - 0.8) ** 2])
    else:
        return np.array([99.9, 99.9])
# Add the simulation with search and surrogate
my_moop.addSimulation({'name': "MySim",
                       'm': 2,
                       'sim_func': sim_func,
                       'search': LatinHypercube,
                       'surrogate': GaussRBF,
                       'hyperparams': {'search_budget': 20}})
```

ParMOO Release

Written in Python

Version 0.2.0 is now available on available on `pip`, `conda-forge`, and `GitHub`

`https://github.com/parmoo/parmoo`

`https://parmoo.readthedocs.io`

[15] Chang and Wild. ParMOO: A Python library for parallel multiobjective simulation optimization. *JOSS 8(82):4468 (2023).*

Example 1: Fayans EDF Model Calibration

Find params $x \in [0,1]^{13}$ to fit the Fayans model to data d_i:

$$M(\xi_i; x) \approx d_i \qquad i = 1, \ldots, 198$$

ParMOO simulation:

$$S_i(x) = M(\xi_i; x) - d_i, \qquad i = 1, \ldots, 198;$$

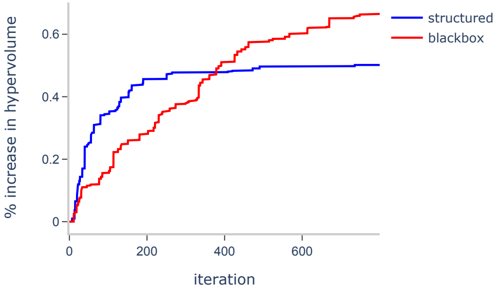
Min SOS across 3 observable classes

$$F_t = \sum_{i=1}^{m_t} \left(S_{t,i}(x) \right)^2$$

[16] Bollapragada et al. Optimization and supervised machine learning methods for fitting numerical physics models without derivatives. Journal of Physics G: Nuclear and Particle Physics 48(2):024001 (2020).

Fayans Solution with ParMOO

- Approximated Fayans model using inv dist weighting on existing dataset
- Implemented parallel solver in ParMOO using libEnsemble
- Just **14-25 lines of Python code**
- Ran for **10K** sim evals
- Compared against same solver w/o exploiting SOS structure



Example 2: Material Manufacturing with ParMOO

Choose optimal settings for material manufacturing in a continuous flow reactor (CFR)

We know how to make a desired material, need to produce at scale:

1. **Maximize the product** (battery electrolyte: TFML)
2. Can increase temperature to **reduce reaction time**
3. Too much heat activates a side reaction; need to **minimize unwanted byproduct**

Challenges:

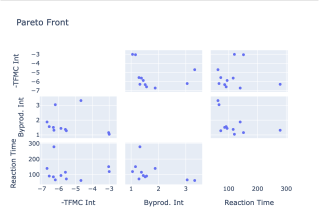
- Mixed variable types
- Heterogeneous objectives
- Must send experiments to run on CFR

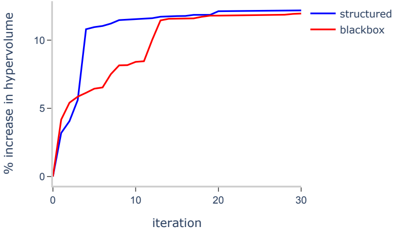
CFR Optimization with ParMOO

Extend `MOOP` class to send/receive experiment data using `MDML` library (Apache Kafka)

Used categorical variable embeddings

Modeled Product/Byproduct as simulations and reaction time using algebraic equation of input





[17] *Chang et al. A framework for fully autonomous design of materials via multiobjective optimization and active learning: challenges and next steps. Under review.*

Resources

GitHub: `github.com/parmoo/parmoo`
Docs: `parmoo.readthedocs.io`
PyPI: `pip install parmoo`
Conda: `conda install --channel=conda-forge parmoo`

E-mail: `tchang@anl.gov`
E-mail: `parmoo@mcs.anl.gov`

Chang and Wild. JOSS 8(82):4468 (2023)

This material is based upon work supported by the U.S. Department of Energy, Office of Science, Office of Advanced Scientific Computing Research, SciDAC program under contract number DE-AC02-06CH11357.