

ParMOO: A Python library for parallel multiobjective simulation optimization

Tyler Chang^a and Stefan Wild^{a→b}

^aMathematics and Computer Science Division,
Argonne National Laboratory

^bApplied 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

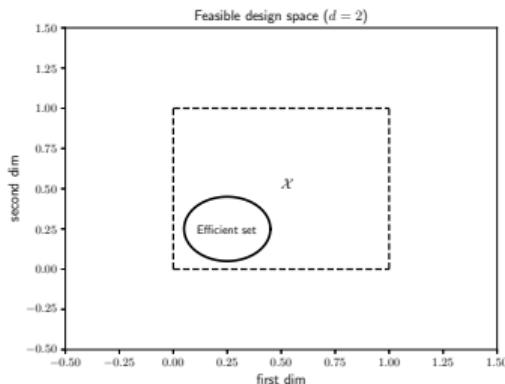
Grand challenges + some closing thoughts

Multiobjective Optimization Problems

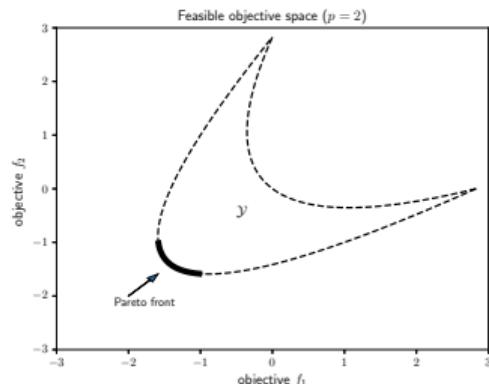
$$\min_{x \in \mathcal{X}} F(x)$$

Multiobjective Optimization Problems

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

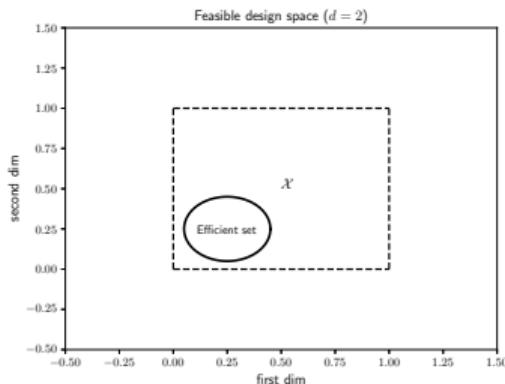


$$F : \mathcal{X} \rightarrow \mathcal{Y}$$



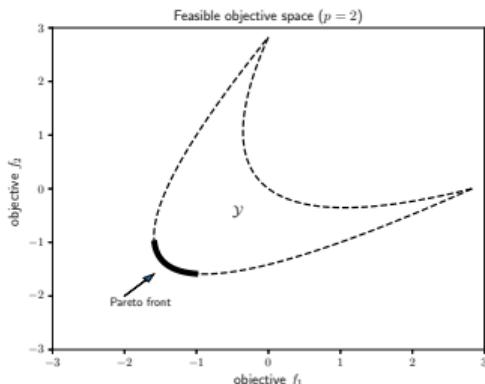
Multiobjective Optimization Problems

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



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

expensive
blackbox process

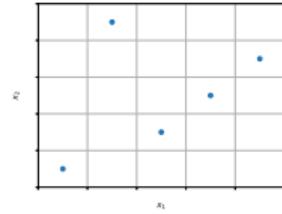


Multiobjective Response Surface Methodology

or Model-Based Optimization or Active Learning

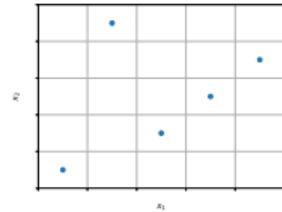
Multiobjective Response Surface Methodology

or Model-Based Optimization or Active Learning



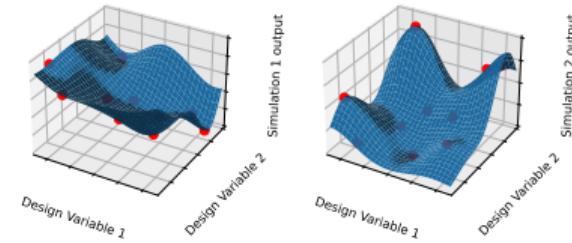
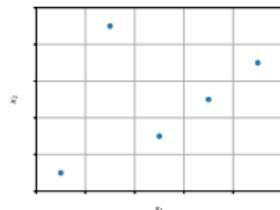
Multiobjective Response Surface Methodology

or Model-Based Optimization or Active Learning



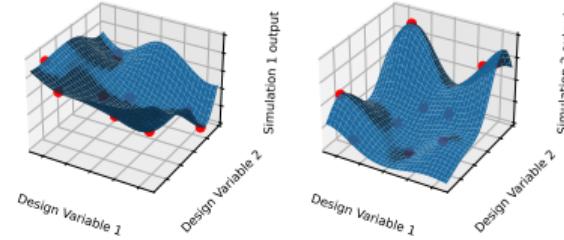
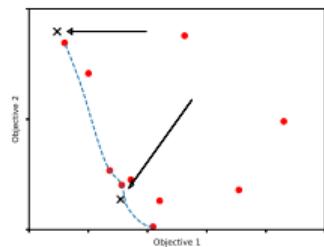
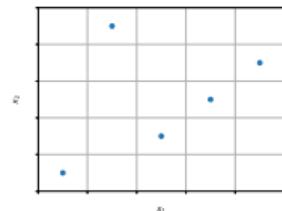
Multiobjective Response Surface Methodology

or Model-Based Optimization or Active Learning



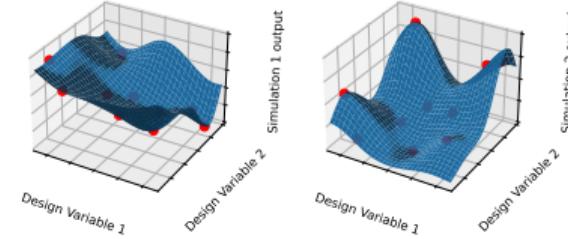
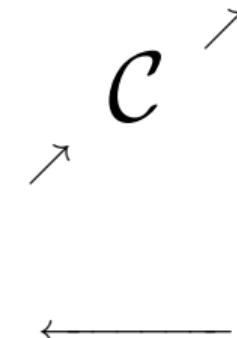
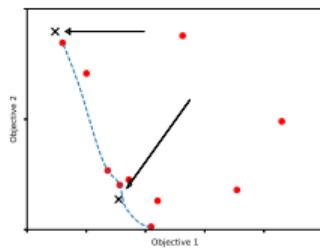
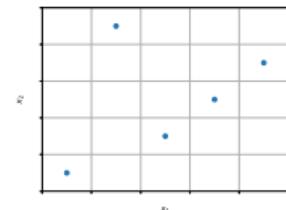
Multiobjective Response Surface Methodology

or Model-Based Optimization or Active Learning

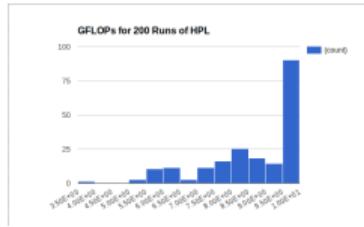


Multiobjective Response Surface Methodology

or Model-Based Optimization or Active Learning



Example: HPC Performance Tuning

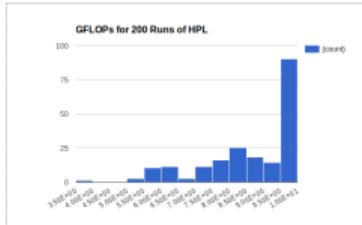


VT VarSys Project – 40 runs of HPL



ANL – LCRC Computing Resources: Bebop

Example: HPC Performance Tuning



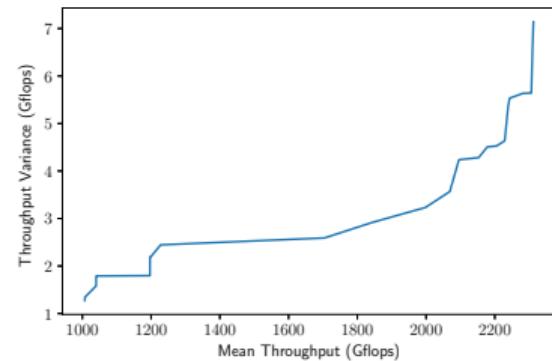
VT VarSys Project – 40 runs of HPL



ANL – LCRC Computing Resources: Bebop

```
HPL.dat
HPL (High Performance Linpack) benchmark input file
Copyright (c) 2002, University of Tennessee
HPL.out    output file name (if any)
0          generate output (1=yes,0=no),0,1
0          number of problems solved (N)
29 30 31 35  Ns
0          # of RHS
1 2 3 4  NBs
0          # of process mapping (NbRows,1=column-major)
3          # of process grids (P > 1)
2 4 4  Ps
0          threshold
3          # of nested fact
0 1 2  PRFACTs (0:left, 1=Crout, 2=right)
2          # of recursive stopping criterion
NMIN
1          # of panels in recursion
NMAX
2          # of recursive panel fact
NFACTS
3          PRFACTs (0:left, 1=Crout, 2=right)
1 2  E of broadcast
0 1  E of transpose
0 1  E of column fact
0 1  E of row fact
0 1  E of column transpose
0 1  E of row transpose
0 1  E of broadcast transpose
0 1  E of transpose broadcast transpose
0 1  Equilibrate (1=none,2=sym,3=col)
0 1  memory alignment (in double C > 0)
```

VTMOP solver



[1] Chang et al. Multiobjective optimization of the variability of the high-performance LINPACK solver. In Proc. WSC 2020.

Challenge 1:

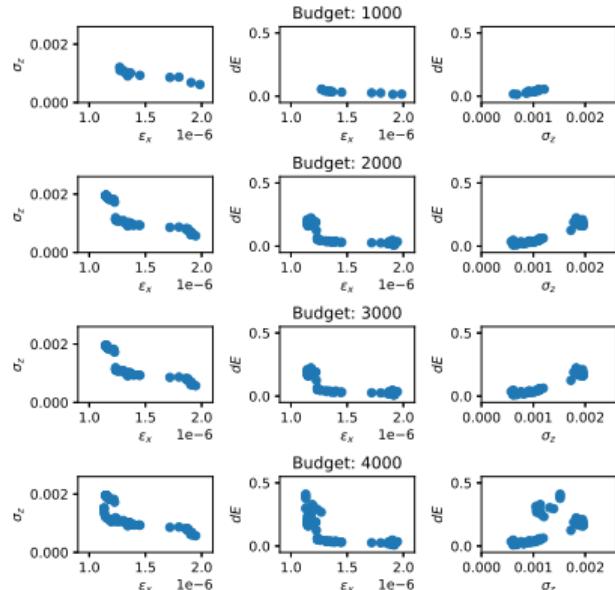
mixed vars + problem types

Example: Particle Accelerator Design

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

```
OPAL.dat
MULinac benchmark input file
Developed by the Accelerator Laboratory, University of Tennessee
10.1.0.0          default float type (double)
0               device set (bsl0000,bsl0001,filg)
4               # of problems sizes (M)
20 30 35      Ns
2               # of Ns
1 2 3 4        Ns
0               # of MPI
1               # of threads
2 1 4          Ps
2               # of process grids P = 3!
1               Ts
10.0           threshold
2               # of parallel fact
0 3 2          PMFACT (0=left, 3=right, 2=right)
2 4          # of recursive stopping criterium
0               NOFACT
1               # of panels in recursion
NOFACT
3 2          # of recursive panel fact.
PMFACT (0=left, 3=right, 2=right)
1               # of parallel fact
BCMATH (0=long,1=short,2=reg,3=2M,4=long,5=4M)
1               # of parallel fact
DEPTHS (0=0)
0               DDFP (0=none,1=long,2=mid)
0               Swapping threshold (0=0)
L1 in (0=transposed,1=non-transposed) Term
0               L2 in (0=transposed,1=non-transposed) Term
Equilibrium (0=none,1=yes)
memory alignment in double (> 0)
```

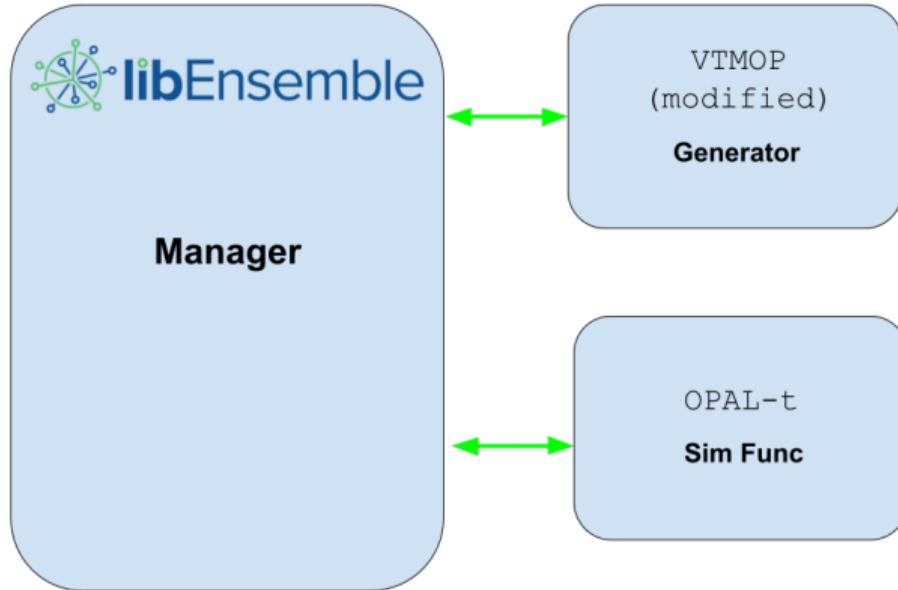
VTMOP solver



[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

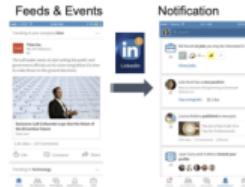


[4] Chang et al. Managing computationally expensive blackbox multiobjective optimization problems with libEnsemble. In Proc. SpringSim 2020.

Challenge 2: **parallel evals + computing environments**

Commercial solutions

Commercial solutions

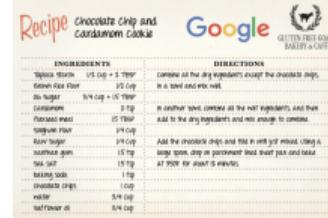


"Using Bayesian optimization for balancing metrics in recommendation systems" by Yunbao Ouyang et al. on LinkedIn Engineering Blog.

Commercial solutions



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

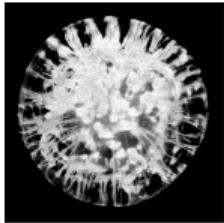


"The makings of a smart cookie" by Daniel Golovin on Google Research Blog.

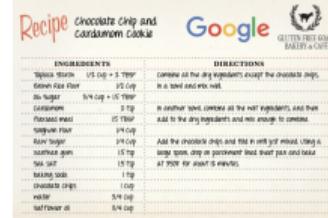
Commercial solutions



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



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



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

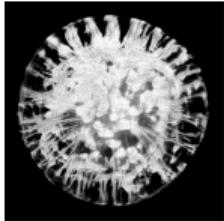
Commercial solutions



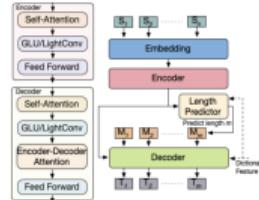
"Using Bayesian optimization for balancing metrics in recommendation systems" by Yunbao Ouyang et al. on LinkedIn Engineering Blog.



"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.

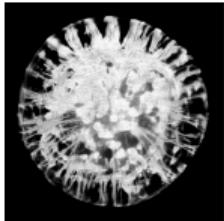


"Optimizing model accuracy and latency using Bayesian multi-objective NAS" by David Eriksson et al. on Meta AI Research Blog.

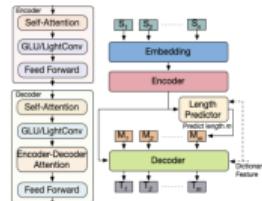
Commercial solutions



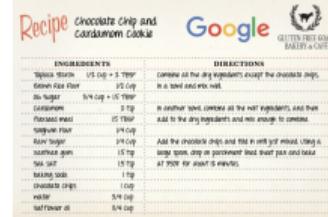
"Using Bayesian optimization for balancing metrics in recommendation systems" by Yunbao Ouyang et al. on LinkedIn Engineering Blog.



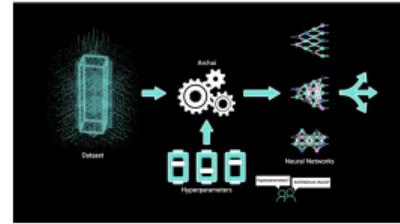
"Accelerating molecular optimization with AI" by Payel Das et al. on IBM Research Blog.



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



"The makings of a smart cookie" by Daniel Golovin on Google 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 in structure-exploiting blackbox optimization

SOA in 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 in structure-exploiting blackbox optimization



SciPy

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



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 in structure-exploiting blackbox optimization



SciPy

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



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. In *Advances and Trends in Optimization with Engineering Applications* (2017).

Challenge 3:

SOA optimization + exploiting problem structure

ParMOO Design Criteria

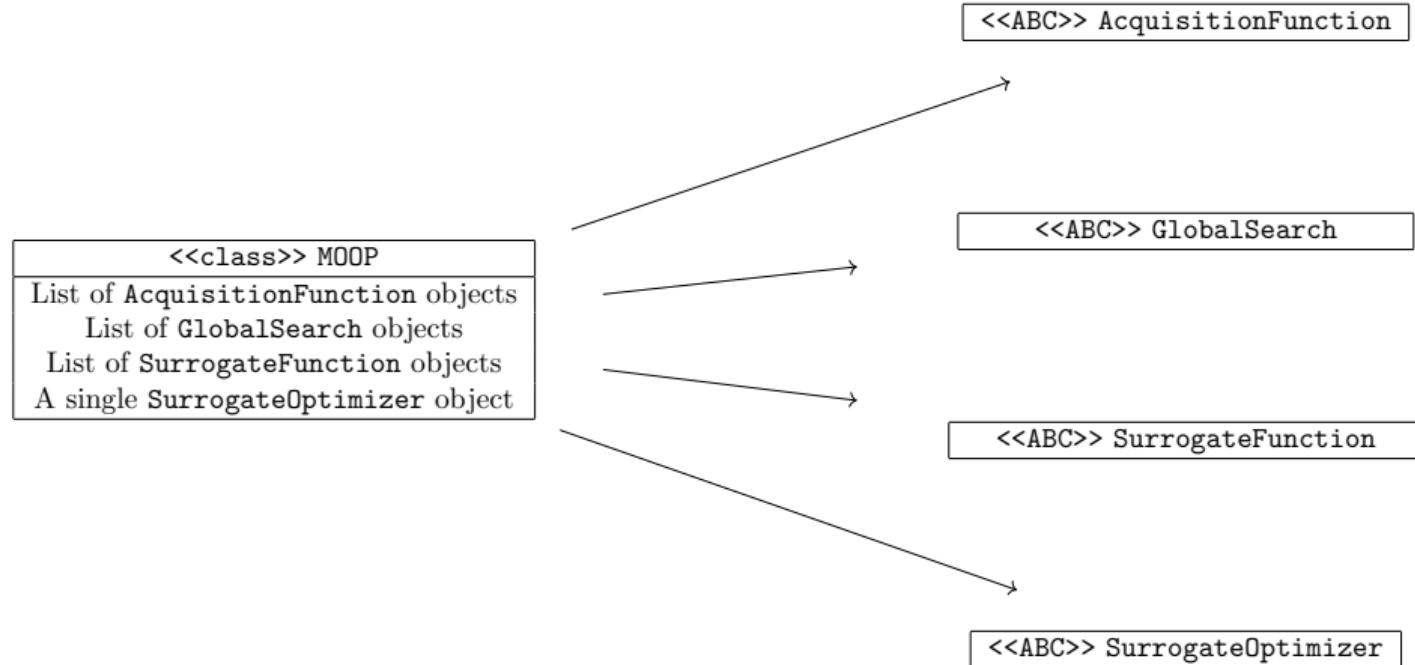
Design goals:

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

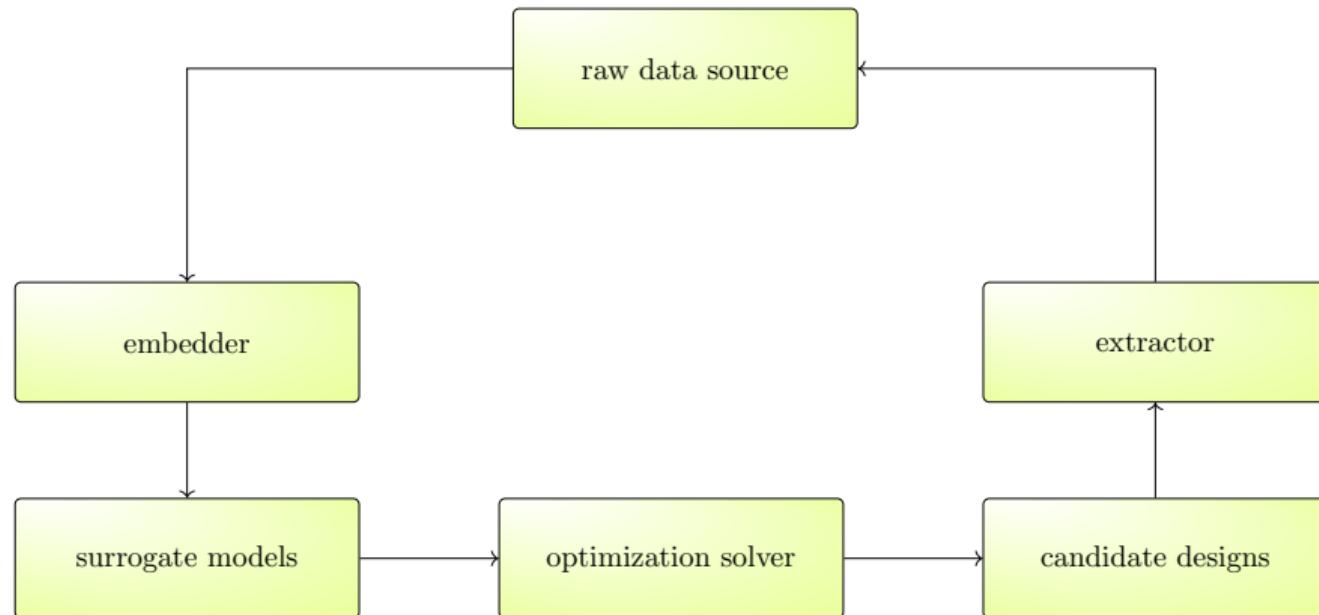
[12] Chang and Wild. *Designing a framework for solving multiobjective simulation optimization problems*. In prep.

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.

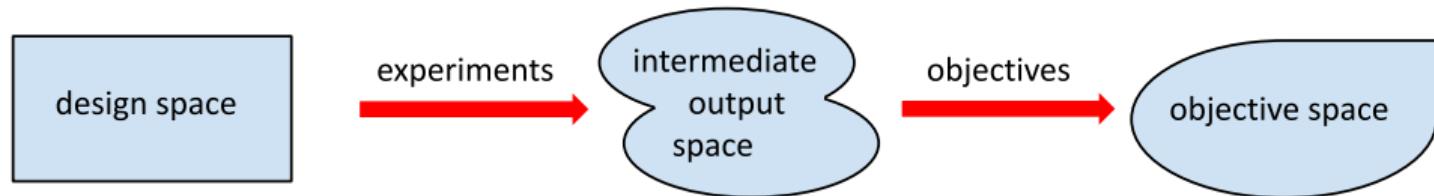
Examples:

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

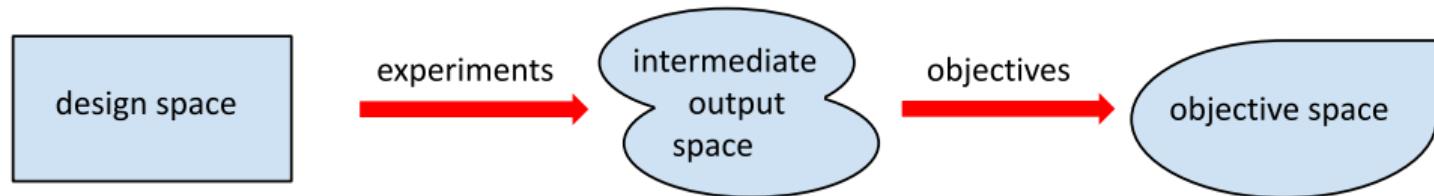
[13] Hudson et al. *libEnsemble: a library to coordinate the concurrent evaluation of dynamic ensembles of calculations*. IEEE TPDS 33(4):977–988 (2021).

[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



Goal 4: problem structure



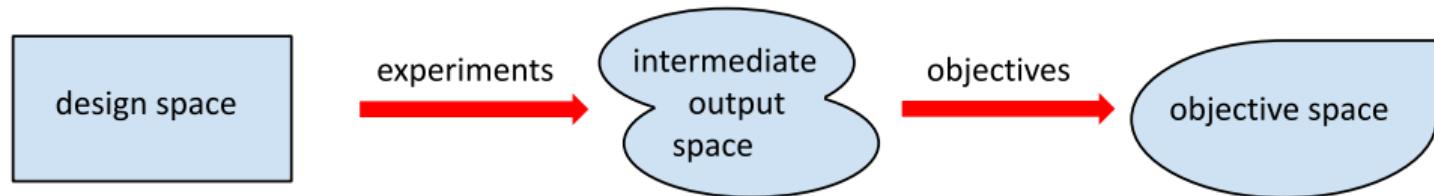
Sum-of-squares structure:

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

where each N_1, \dots, N_o is an index set.

Increases order of approximation \Rightarrow
increases order of convergence

Goal 4: problem structure



Sum-of-squares structure:

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

where each N_1, \dots, N_o is an index set.

Increases order of approximation \Rightarrow
increases order of convergence

Heterogeneous MOOPs:

$$\begin{aligned} h_1(x, S(x)) &= S_1(x) \\ h_2(x, S(x)) &= \|x\|^2 \end{aligned}$$

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

Sample code

```
from parmoo import MOOP
from parmoo.optimizers import LocalGPS as gps
from parmoo.searches import LatinHypercube as lhs
from parmoo.surrogates import GaussRBF as rbf
from parmoo.acquisitions import UniformWeights as wsum
# Create MOOP object with GPS optimizer
moop = MOOP(gps)
# Add a continuous + categorical design variable
moop.addDesign({'name': "x1", 'lb': 0.0, 'ub': 1.0})
moop.addDesign({'name': "x2", 'des_type': "cat", 'levels': 3})
# Define and add a simulation function (with surrogates and search)
def s(x): return [(x["x1"]-.2)**2, (x["x1"]-.8)**2] if x["x2"]==0 else [9,9]
moop.addSimulation({'name': "sim", 'm': 2, 'sim_func': s,
                     'search': lhs, 'surrogate': rbf})
# Add 2 objectives
moop.addObjective({'name': "f1", 'obj_func': lambda x, s: s["sim"][0]})
moop.addObjective({'name': "f2", 'obj_func': lambda x, s: s["sim"][1]})
# Add 3 weighted-sum acquisition functions
for i in range(3):
    moop.addAcquisition({'acquisition': wsum})
# Solve with 5 iterations and fetch numpy struct of solutions
moop.solve(5)
results = moop.getPF()
```

ParMOO Release



Written in Python

Version 0.2.0 is now 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 :

$$M(\xi_i; x) \approx d_i \quad i = 1, \dots, 198$$

ParMOO simulation:

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

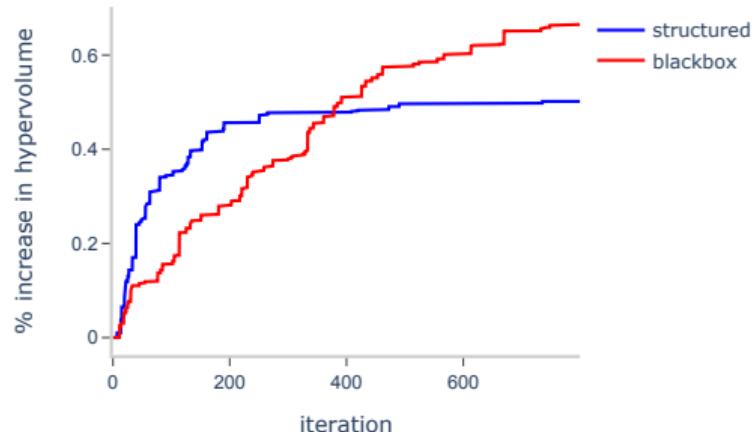
Min SOS across 3 observable classes

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

[16] Bollapragada et al. Optimization and supervised machine learning methods for fitting numerical physics models without derivatives. *Journal of Physics G* 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**
- ▶ Structure-exploiting is better at small budgets, blackbox can be better at large budgets



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:

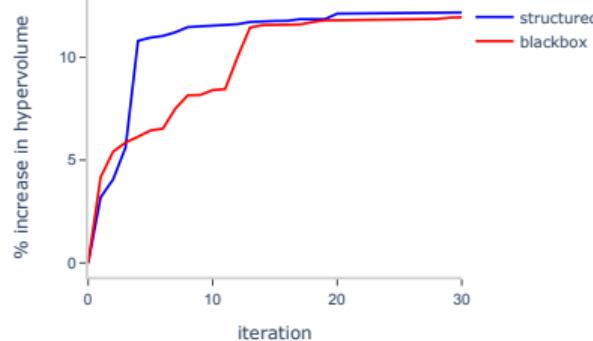
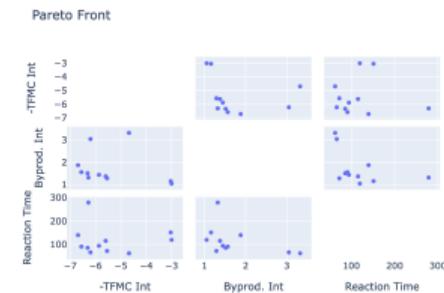
- ▶ Mixed variable types
- ▶ Heterogeneous objectives
- ▶ Must send experiments to run on CFR

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.

Grand Challenges

Grand Challenges

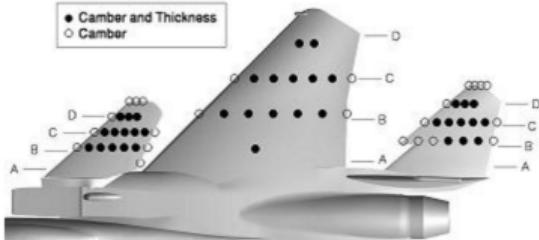


Molecular discovery via multifidelity, multiobjective optimization with simulation and experiment data.

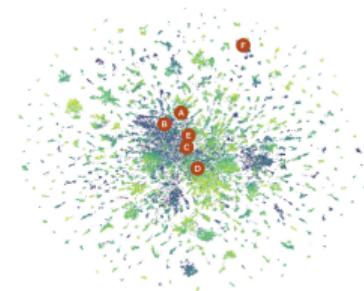
Grand Challenges



Molecular discovery via multifidelity, multiobjective optimization with simulation and experiment data.



"Application #24. Design of F-15 with simulated aeroelastic effects" by Eric Nielsen et al. NASA FUN3D Applications.



"ESM Metagenomic Atlas: the first view of the dark matter of the protein univ."
Meta AI Research Blog.

References

- [1] Chang et al. Multiobjective optimization of the variability of the high-performance LINPACK solver. In Proc. WSC 2020.
- [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).
- [4] Chang et al. Managing computationally expensive blackbox multiobjective optimization problems with libEnsemble. In Proc. SpringSim 2020.
- [5] Song et al. OSS Vizier: distributed infrastructure and API for reliable and flexible black-box optimization. In Proc. 2022 AutoML-Conf.
- [6] Balandat et al. BoTorch: a framework for efficient monte-carlo Bayesian optimization. In NeurIPS 2020.
- [7] Hoffman et al. Optimizing molecules using efficient queries from property evaluations. Nature Machine Intelligence 4:21–31 (2022).
- [8] Shah et al. Archai: platform for neural architecture search. Microsoft Research (Jul, 2022).
- [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. In Advances and Trends in Optimization with Engineering Applications (2017).
- [12] Chang and Wild. Designing a framework for solving multiobjective simulation optimization problems. In prep.
- [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.
- [15] Chang and Wild. ParMOO: A Python library for parallel multiobjective simulation optimization. JOSS 8(82):4468 (2023).
- [16] Bollapragada et al. Optimization and supervised machine learning methods for fitting numerical physics models without derivatives. Journal of Physics G 48(2):024001 (2020).
- [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.