Exploiting structures in multiobjective simulation optimization problems

Tyler Chang[a] and Stefan Wild$^{a \to b}$

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

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

SIAM OP 23

Outlines

Intro, ParMOO, and Problem Types

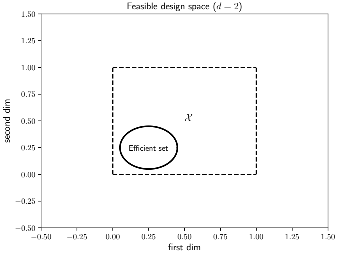
Initial Results + Optimizer Stalling

Deep-dive into RBFs and Connection to Bayesian optimization

Comparison with Bayesian optimization

Multiobjective Optimization Problems

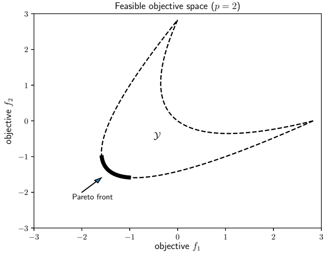
$$\min_{x \in \mathcal{X}} F(x)$$



$$\longrightarrow$$

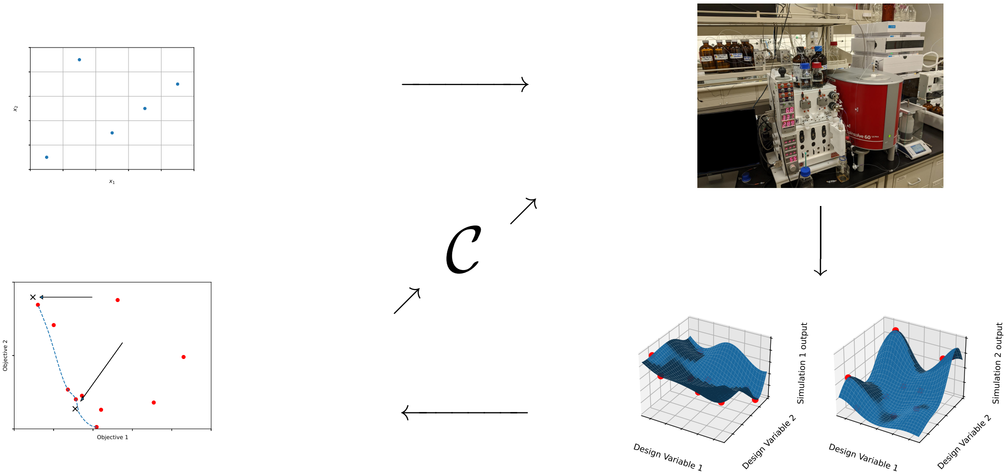
$$F : \mathcal{X} \to \mathcal{Y}$$

expensive
blackbox process



Multiobjective Response Surface Methodology

or Model-Based Optimization or Active Learning



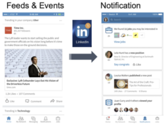
Challenge 1:

Mixed vars & problem types

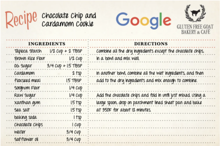
+

Unusual computing environments

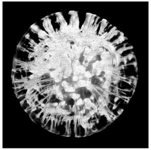
Commercial solutions



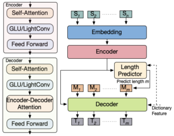
"Using Bayesian optimization for balancing metrics in recommendation systems" by Yunbao Ouyang et al. on LinkedIn Engineering Blog.



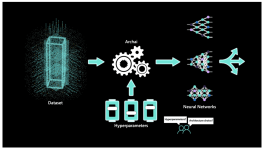
"The makings of a smart cookie" by Daniel Golovin on Google Research Blog.



"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.



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

Challenge 2:

SOA blackbox optimization

$+$

Exploiting problem structure

SOA in blackbox optimization

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


siam news
Optimization and Learning with Zeroth-order Stochastic Oracles

Stochastic dimension reduction explained in this context by Stefan.

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

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

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

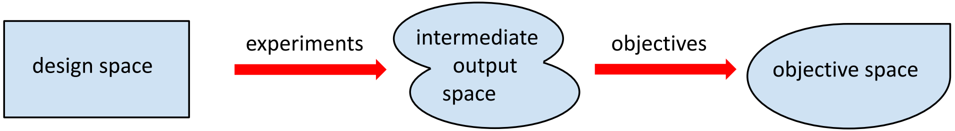
Wild. Solving derivative-free nonlinear least squares problems with POUNDERS. In Advances and Trends in Optimization with Engineering Applications (2017).

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

Problem structures



Least-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

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

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

`https://parmoo.readthedocs.io`

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

Example 1: 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
- Limited budget

Design vars and bound constraints

Parameter	LB	UB
Temp (deg C)	40	150
Reaction time (secs)	60	300
Equivalence ratio (N/A)	0.9	2
Solvent (categorical)	2	lvls
Base (categorical)	2	lvls

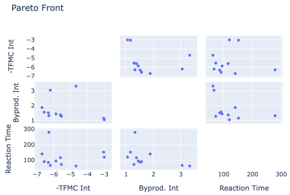
CFR Optimization with ParMOO

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

Used embeddings to represent categorical variables

Modeled Product/Byprod as sims and reaction time using ID mapping from input

Ran to convergence and used data to create a surrogate problem for future study

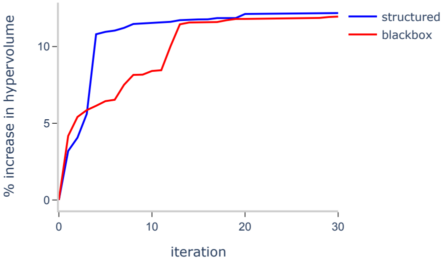


Chang et al. A framework for fully autonomous design of materials via multiobjective optimization and active learning: challenges and next steps. In ICLR 2023, Workshop on ML4Materials.

CFR Optimization Results

- 5 mixed-type design variables (including reaction time), 3 heterogeneous objs
- 50-pt Latin hypercube design-of-experiments
- Fit **global** Gaussian RBF surrogate on 2 simulation outputs
- 3rd obj is identity mapping of reaction time
- 3 ε-constraint scalarization functions
- Multi-start L-BFGS-B global optimization of scalarized objectives
- Compared against blackbox implementation: all 3 objectives modeled with Gaussian RBFs

Ran 30 iterations (batch size 3) after 50-pt DOE (avg over 5 seeds)



Notice: structure-exploiting is better early on, then **stalls out** at the end...

Example 2: 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

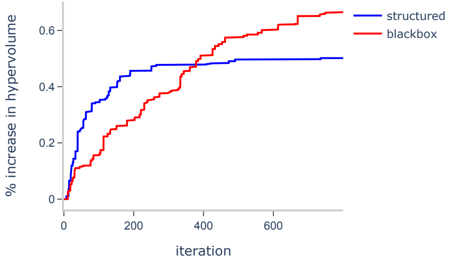
$$F_t = \sum_{i=1}^{m_t} \left(S_{t,i}(x)\right)^2$$

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

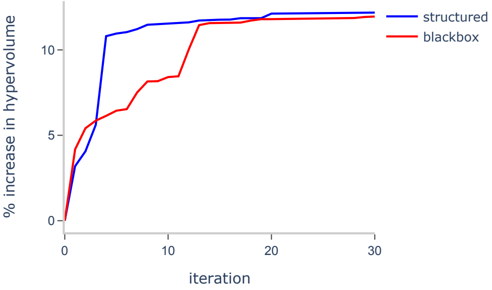
- Approximated Fayans residuals with MLP trained on existing dataset
- Implemented batch-parallel structure-exploiting solver in ParMOO
  - 13 continuous design variables
  - 2000-pt LH design-of-experiments
  - Use **local** Gaussian RBF surrogates to model sim outs
  - SOS structure applied on top of Gauss RBF surrogate outputs
  - Solve 10 randomized ε-constraint scalarizations per batch
  - Solve each with a trust-region multi-start L-BFGS-B
- Compared against same solver w/o exploiting least-squares structure (Gaussian RBFs fit directly to three objective values)

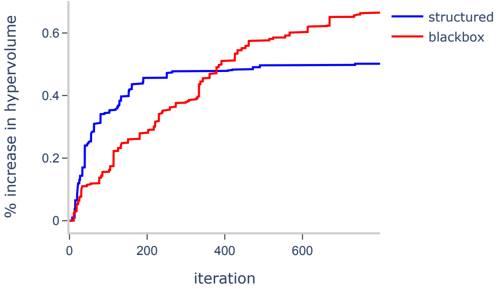
Ran for 800 iterations (10K sim evals)
Avg over 5 random seeds



Structure-exploiting is better at small budgets, but **again stalls** out...

Two different problems, same issue...





- Mixed-variable problem
- Extremely limited budget
- Heterogeneous objectives
- Global Gaussian RBF modeling

- Continuous optimization problem
- 13 vars, 3 objs, healthy budget
- Nonlinear least-squares
- Local RBF (trust-region method)

What happened?

What happened?

Many months of debugging and testing...

What happened?

Many months of debugging and testing...

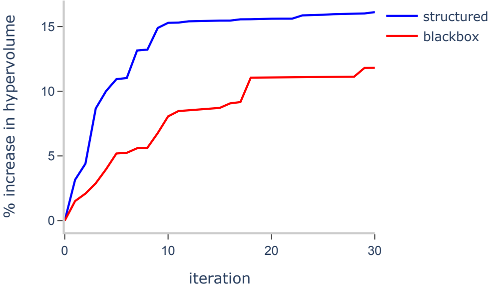
Finally tried centering response values at 0 before fitting surrogate...

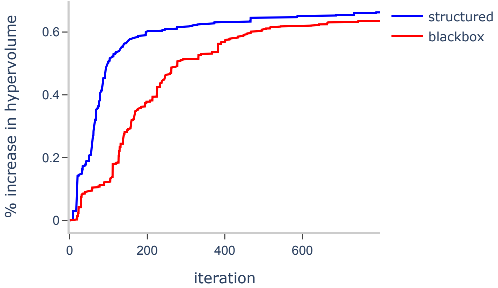
What happened?

Many months of debugging and testing...

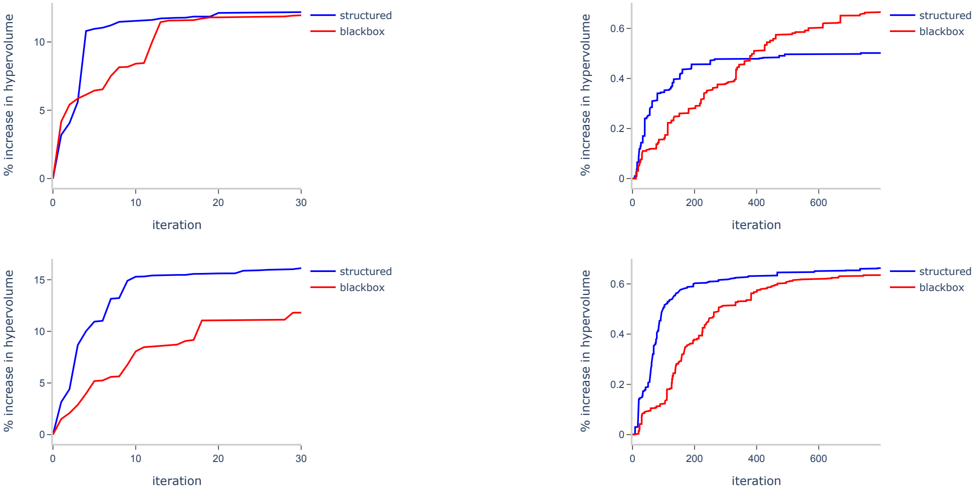
Finally tried centering response values at 0 before fitting surrogate...

Fixed!





Comparison



Chang and Wild. Designing a framework for solving multiobjective simulation optimization problems. ArXiv preprint 2304.06881 (2023).

Equivalence between Gaussian RBFs and GPs

For both Gaussian RBFs and (traditional) Gaussian process mean-function:

$$\hat{F}_{RBF}(x) = \omega^\top \begin{bmatrix} e^{-\|x_1 - x\|^2/\sigma} \\ e^{-\|x_2 - x\|^2/\sigma} \\ \vdots \\ e^{-\|x_n - x\|^2/\sigma} \end{bmatrix}$$

Equivalence between Gaussian RBFs and GPs

For both Gaussian RBFs and (traditional) Gaussian process mean-function:

$$\hat{F}_{RBF}(x) = \omega^\top \begin{bmatrix} e^{-\|x_1 - x\|^2/\sigma} \\ e^{-\|x_2 - x\|^2/\sigma} \\ \vdots \\ e^{-\|x_n - x\|^2/\sigma} \end{bmatrix}$$

where $A\omega = y$ and

$$A = \begin{bmatrix} 1 & e^{-\|x_1 - x_2\|^2/\sigma} & \dots & e^{-\|x_1 - x_n\|^2/\sigma} \\ e^{-\|x_2 - x_1\|^2/\sigma} & 1 & \dots & e^{-\|x_2 - x_n\|^2/\sigma} \\ \vdots & \vdots & & \vdots \\ e^{-\|x_n - x_1\|^2/\sigma} & e^{-\|x_n - x_2\|^2/\sigma} & \dots & 1 \end{bmatrix} \qquad y = \begin{bmatrix} F(x_1) \\ F(x_2) \\ \vdots \\ F(x_n) \end{bmatrix}$$

RBF/GP Conditioning and Accuracy

$$A = \begin{bmatrix} 1 & e^{-\|x_1-x_2\|^2/\sigma} & \dots & e^{-\|x_1-x_n\|^2/\sigma} \\ e^{-\|x_2-x_1\|^2/\sigma} & 1 & \dots & e^{-\|x_2-x_n\|^2/\sigma} \\ \vdots & \vdots & & \vdots \\ e^{-\|x_n-x_1\|^2/\sigma} & e^{-\|x_n-x_2\|^2/\sigma} & \dots & 1 \end{bmatrix}$$

RBF/GP Conditioning and Accuracy

$$A = \begin{bmatrix} 1 & e^{-\|x_1 - x_2\|^2/\sigma} & \dots & e^{-\|x_1 - x_n\|^2/\sigma} \\ e^{-\|x_2 - x_1\|^2/\sigma} & 1 & \dots & e^{-\|x_2 - x_n\|^2/\sigma} \\ \vdots & \vdots & & \vdots \\ e^{-\|x_n - x_1\|^2/\sigma} & e^{-\|x_n - x_2\|^2/\sigma} & \dots & 1 \end{bmatrix}$$

- As we optimize, data points cluster

RBF/GP Conditioning and Accuracy

$$A = \begin{bmatrix} 1 & e^{-\|x_1 - x_2\|^2/\sigma} & \dots & e^{-\|x_1 - x_n\|^2/\sigma} \\ e^{-\|x_2 - x_1\|^2/\sigma} & 1 & \dots & e^{-\|x_2 - x_n\|^2/\sigma} \\ \vdots & \vdots & & \vdots \\ e^{-\|x_n - x_1\|^2/\sigma} & e^{-\|x_n - x_2\|^2/\sigma} & \dots & 1 \end{bmatrix}$$

- As we optimize, data points cluster
- As $\|x_i - x_j\| \to 0$, $A \to$ singularity

RBF/GP Conditioning and Accuracy

$$A = \begin{bmatrix} 1 & e^{-\|x_1 - x_2\|^2/\sigma} & \dots & e^{-\|x_1 - x_n\|^2/\sigma} \\ e^{-\|x_2 - x_1\|^2/\sigma} & 1 & \dots & e^{-\|x_2 - x_n\|^2/\sigma} \\ \vdots & \vdots & & \vdots \\ e^{-\|x_n - x_1\|^2/\sigma} & e^{-\|x_n - x_2\|^2/\sigma} & \dots & 1 \end{bmatrix}$$

- As we optimize, data points cluster
- As $\|x_i - x_j\| \to 0$, $A \to$ singularity
- Decrease the shape parameter σ to keep A nonsingular

RBF/GP Conditioning and Accuracy

$$A = \begin{bmatrix} 1 & e^{-\|x_1-x_2\|^2/\sigma} & \dots & e^{-\|x_1-x_n\|^2/\sigma} \\ e^{-\|x_2-x_1\|^2/\sigma} & 1 & \dots & e^{-\|x_2-x_n\|^2/\sigma} \\ \vdots & \vdots & & \vdots \\ e^{-\|x_n-x_1\|^2/\sigma} & e^{-\|x_n-x_2\|^2/\sigma} & \dots & 1 \end{bmatrix}$$

- As we optimize, data points cluster
- As $\|x_i - x_j\| \to 0$, $A \to$ singularity
- Decrease the shape parameter σ to keep A nonsingular
- Descreasing σ restricts the support of $\hat{F}_{RBF}$

RBF/GP Conditioning and Accuracy

$$A = \begin{bmatrix} 1 & e^{-\|x_1 - x_2\|^2/\sigma} & \dots & e^{-\|x_1 - x_n\|^2/\sigma} \\ e^{-\|x_2 - x_1\|^2/\sigma} & 1 & \dots & e^{-\|x_2 - x_n\|^2/\sigma} \\ \vdots & \vdots & & \vdots \\ e^{-\|x_n - x_1\|^2/\sigma} & e^{-\|x_n - x_2\|^2/\sigma} & \dots & 1 \end{bmatrix}$$

- As we optimize, data points cluster
- As $\|x_i - x_j\| \to 0$, $A \to$ singularity
- Decrease the shape parameter σ to keep A nonsingular
- Descreasing σ restricts the support of $\hat{F}_{RBF}$
  - With 0-prior, optimizer will be driven to low-support regions
  - With GP, uncertainty will be maximal in low-support regions (will drive Bayesian optimization to behave similarly?)

RBF/GP Conditioning and Accuracy

$$A = \begin{bmatrix} 1 & e^{-\|x_1-x_2\|^2/\sigma} & \dots & e^{-\|x_1-x_n\|^2/\sigma} \\ e^{-\|x_2-x_1\|^2/\sigma} & 1 & \dots & e^{-\|x_2-x_n\|^2/\sigma} \\ \vdots & \vdots & & \vdots \\ e^{-\|x_n-x_1\|^2/\sigma} & e^{-\|x_n-x_2\|^2/\sigma} & \dots & 1 \end{bmatrix}$$

- As we optimize, data points cluster
- As $\|x_i - x_j\| \to 0$, $A \to$ singularity
- Decrease the shape parameter σ to keep A nonsingular
- Decreasing σ restricts the support of $\hat{F}_{RBF}$
  - With 0-prior, optimizer will be driven to low-support regions
  - With GP, uncertainty will be maximal in low-support regions (will drive Bayesian optimization to behave similarly?)
- "Uncertainty principle" – for imbalanced datasets, cannot have accuracy and solvability when working with RBF-like models

Big Question

Does Bayesian optimization stall out for structure-exploiting solvers?

Big Question

Does Bayesian optimization stall out for structure-exploiting solvers?

How to even check something like this?

Implemented Bayesian Optimization in ParMOO

To check, we implemented Bayesian optimization in ParMOO

Implemented Bayesian Optimization in ParMOO

To check, we implemented Bayesian optimization in ParMOO

Showing **18 changed files** with **3,394 additions** and **852 deletions**.

- When composing GPs (from simulation outputs) into nonlinear objectives, the posterior depends on the function
  - May or may not be iid
- Monte carlo integrated to evaluate expected-improvement in the ε-constraint scalarization
  - It's very expensive

Implemented Bayesian Optimization in ParMOO

To check, we implemented Bayesian optimization in ParMOO

Showing **18 changed files** with **3,394 additions** and **852 deletions**.

- When composing GPs (from simulation outputs) into nonlinear objectives, the posterior depends on the function
  - May or may not be iid
- Monte carlo integrated to evaluate expected-improvement in the ε-constraint scalarization
  - It's very expensive
- Use BoTorch if you can

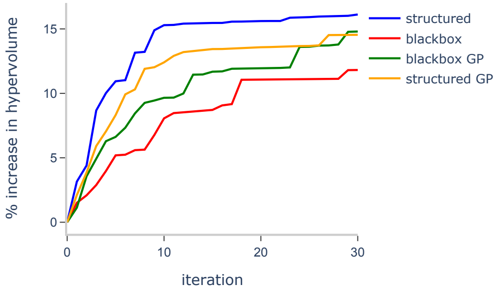
Bayesian optimization of CFR problem

- 5 mixed-type design variables (including reaction time)
- 50-pt Latin hypercube design-of-experiments
- Fit **global** Gaussian RBF surrogate on 2 simulation outputs
- 3rd obj is identity mapping of reaction time
- Compared against blackbox implementation: all 3 objectives modeled with Gaussian RBFs
- 3 EI of ε-constraint acquisition functions
- Monte carlo integration of 2 Gaussian sim outs
- Multi-start GPS for global optimization of scalarized objectives

*Green highlight = changes made from before

CFR Bayesian optimization results

Deja vu?



Bayesian optimization of Fayans problem

- 13 continuous design variables
- 2000-pt LH design-of-experiments
- Use **local** Gaussian proc. surrogates to model sims†
- SOS structure applied on top of Gauss RBF surrogate outputs
- Implemented batch-parallel structure-exploiting solver in ParMOO
- Compared against same solver w/o exploiting least-squares structure (Gaussian RBFs fit directly to three objective values)
- 10 EI of ε-constraint acquisition functions
- Monte carlo integration of 198-dim Gaussian sim outs
- Multi-start GPS in TR to solve optimization of scalarized objectives

*Green highlight = changes made from before

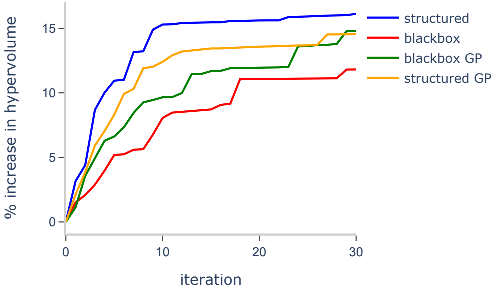
†*Eriksson et al. Scalable global optimization via local Bayesian optimization. In NeurIPS 2019.*

Fayans Bayesian optimization results

Had to drastically reduce budget (200 pt LH, 80 iterations)

MC integration with only 300 samples (enough?)

Only 2 random seeds



Conclusion & Next steps

- Solving Bayesian optimization too fast seems to stall out
  - This is actually mentioned in literature
  - Related to poor conditioning of RBF kernels?
  - Is it just the Gaussian processes? What about other models?
  - What if I optimized the shape parameter instead of setting it algebraically?
  - How do the "default" BoTorch methods do?
- Do this at a larger scale to get more thorough results
  - Parallelize surrogate problem solvers
  - Gradient-based solvers & faster MC integrators

Garnett. Bayesian optimization [Ch. 9.2] (2023).

References

Lit review/motivation refs

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

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

Wild. Solving derivative-free nonlinear least squares problems with POUNDERS. In Adv. & Trends in Optimization with Eng. Applications (2017).

Eriksson et al. Scalable global optimization via local Bayesian optimization. In NeurIPS 2019.

Garnett. Bayesian optimization [Ch. 9.2] (2023).

ParMOO software & test problems

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

Chang and Wild. Designing a framework for solving multiobjective simulation optimization problems. ArXiv preprint 2304.06881 (2023).

Datasets

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

Chang et al. A framework for fully autonomous design of materials via multiobjective optimization and active learning: challenges and next steps. In ICLR 2023 Workshop on ML4Materials.

Resources

GitHub: `github.com/parmoo/parmoo`

Pip: `pip install parmoo`

Conda: `conda install --channel=conda-forge parmoo`

Test problems: `github.com/parmoo/parmoo-solver-farm`

`tchang@anl.gov`

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.