SIAM CSE21

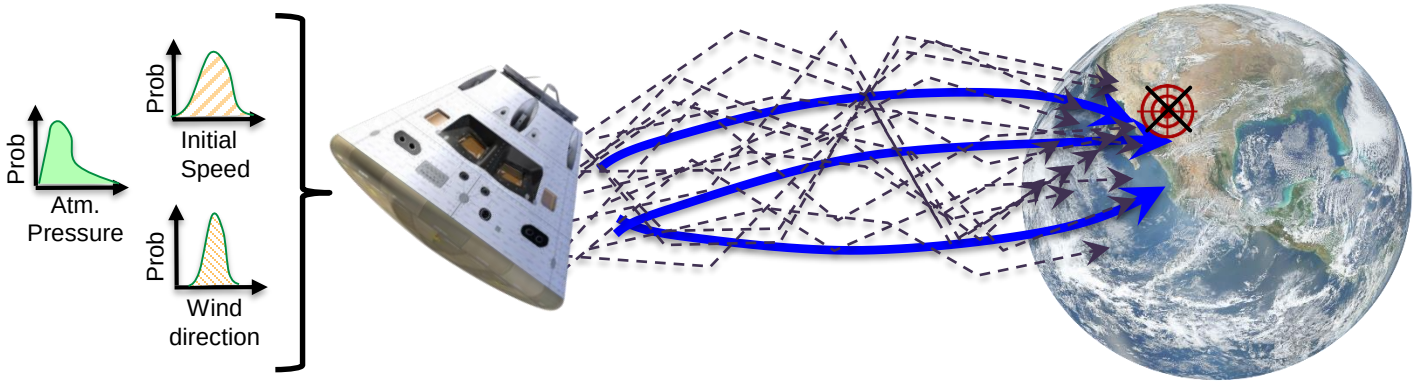


Parametrically-defined Approximate Control Variates

GF Bomarito, PE Leser, JE Warner, WP Leser, L Morrill

NASA Langley Research Center

Motivating Application



Multifidelity Trajectory Simulation

- Uncertain initial conditions → uncertain landing location
- Used Multilevel Monte Carlo (MLMC)
- …then Multifidelity Monte Carlo (MFMC)
- …then Approximate Control Variates (ACVs)

We can't be the only ones

Outline

- Problem statement
- ACVs and Sample Allocation Optimization
- Parametrically-defined ACVs
- Results

Problem Statement

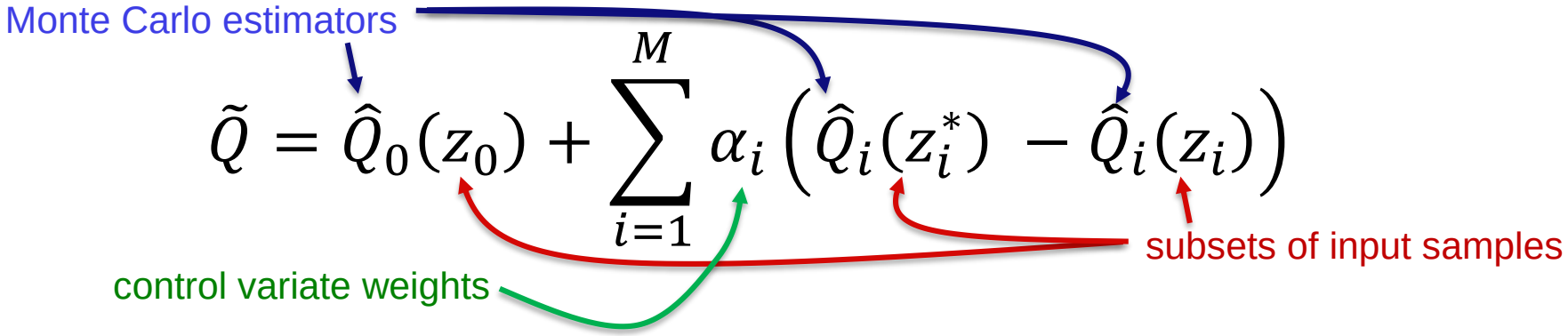
Given:

- A high fidelity model Q_0 with computational cost w_0
- M lower-fidelity models Q_i with costs w_i
- The covariance among models
- A time budget $\hat{w}$

Find:

- The best estimate of the mean of the high fidelity model $E[Q_0]$

Approximate Control Variates[1]



- MLMC[2] and MFMC[3] are instances of this estimator
- New ACV estimators[1] ACVIS, ACVMF, and ACVKL
- Estimator is unbiased (wrt $E[Q_0]$)
- Variance (error) controlled by size and intersections of z_0, z_i^*, z_i
- Total estimator cost also controlled by size and intersections of z_0, z_i^*, z_i

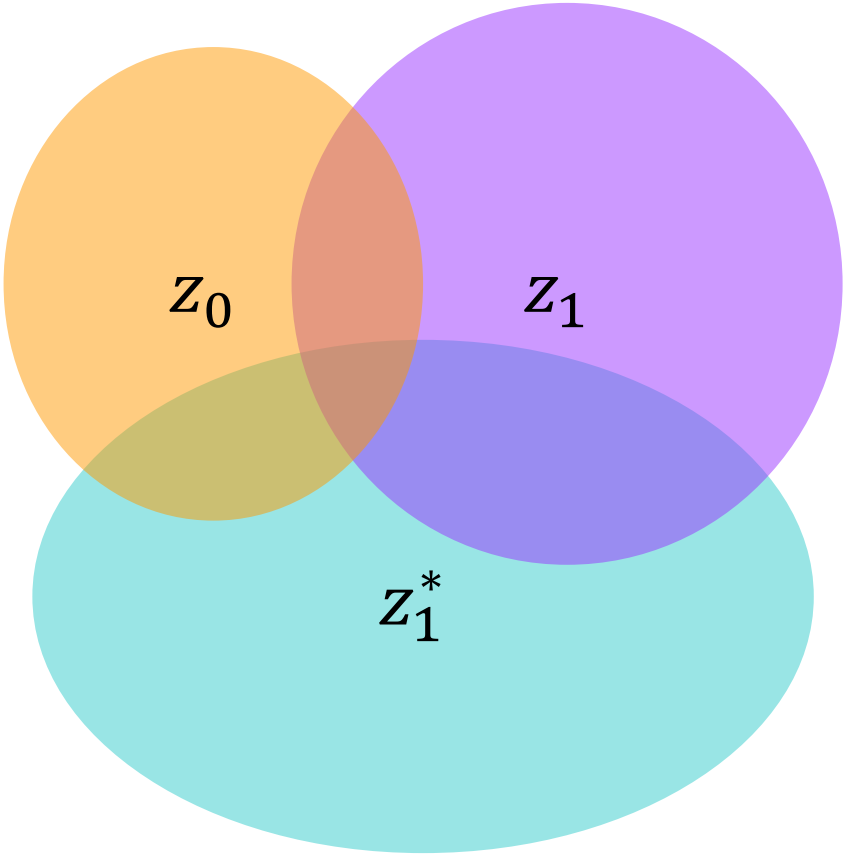
Goal: Sample allocation optimization

Find z_0, z_i^*, z_i such that variance is minimal and cost is under budget

[1] Gorodetsky, A A., et al. Journal of Computational Physics (2020)
[2] Giles, M B. Operations Research (2008)
[3] Peherstorfer, B, et al. SIAM Journal on Scientific Computing (2016)

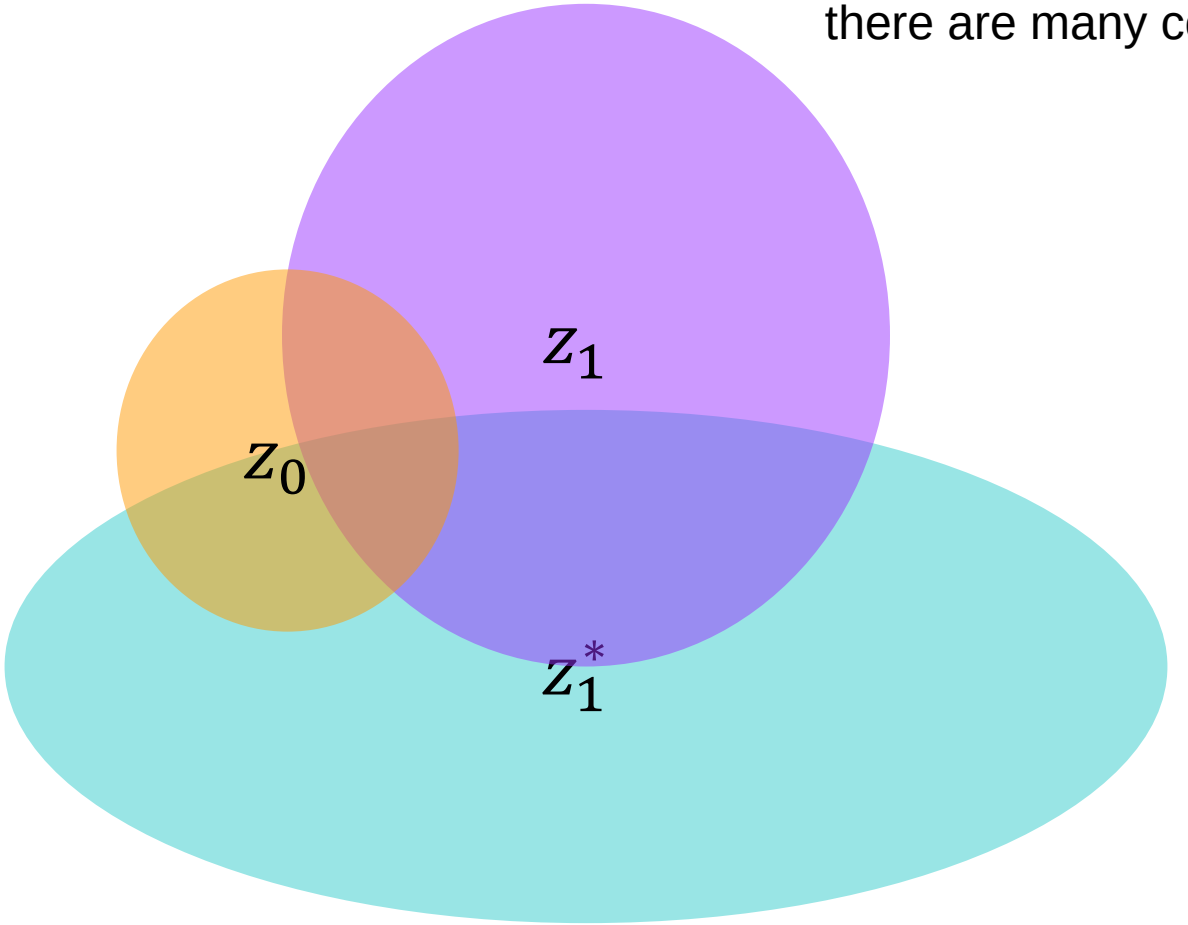
Sample Allocation Optimization

must optimize size and overlap



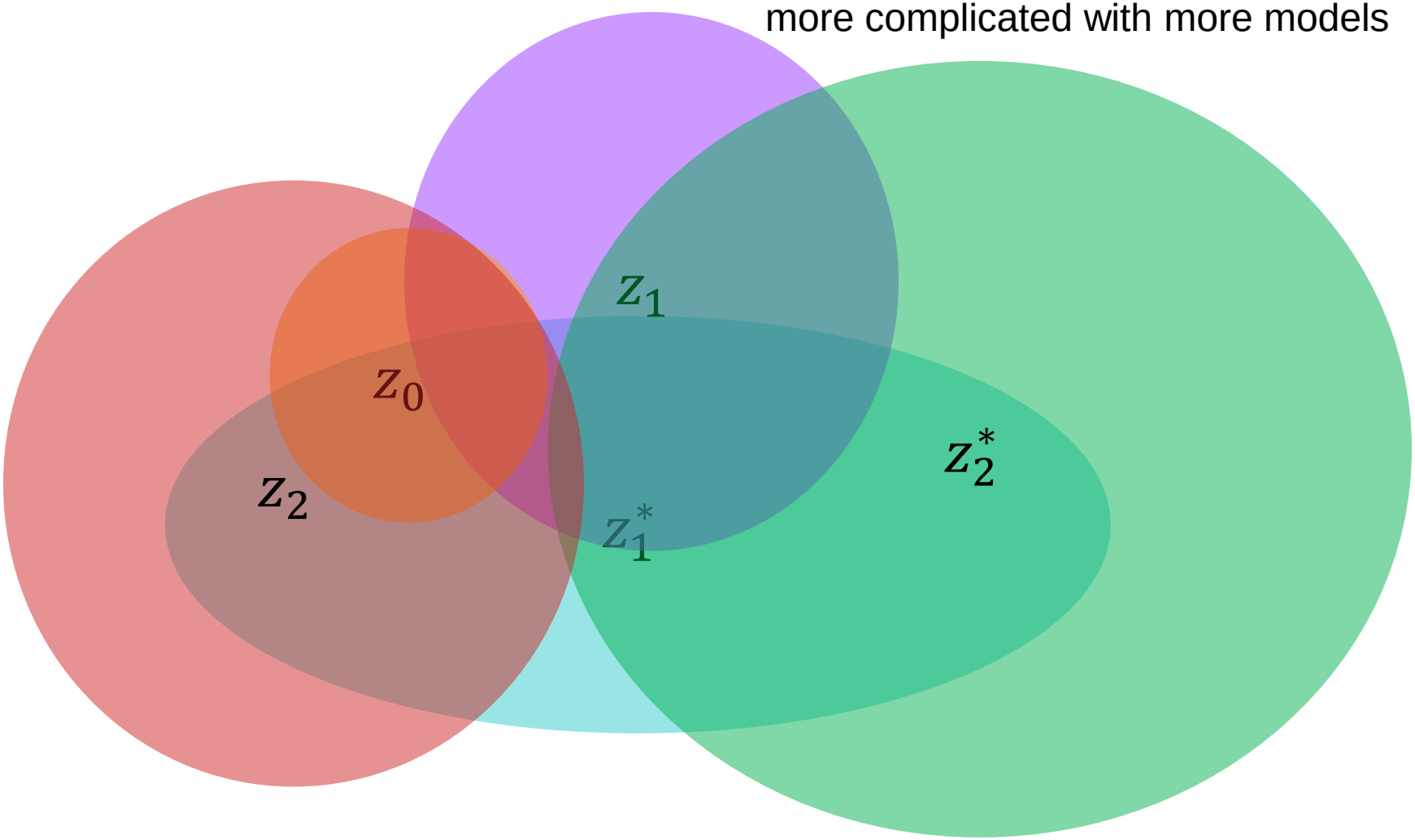
Sample Allocation Optimization

there are many configurations



Sample Allocation Optimization

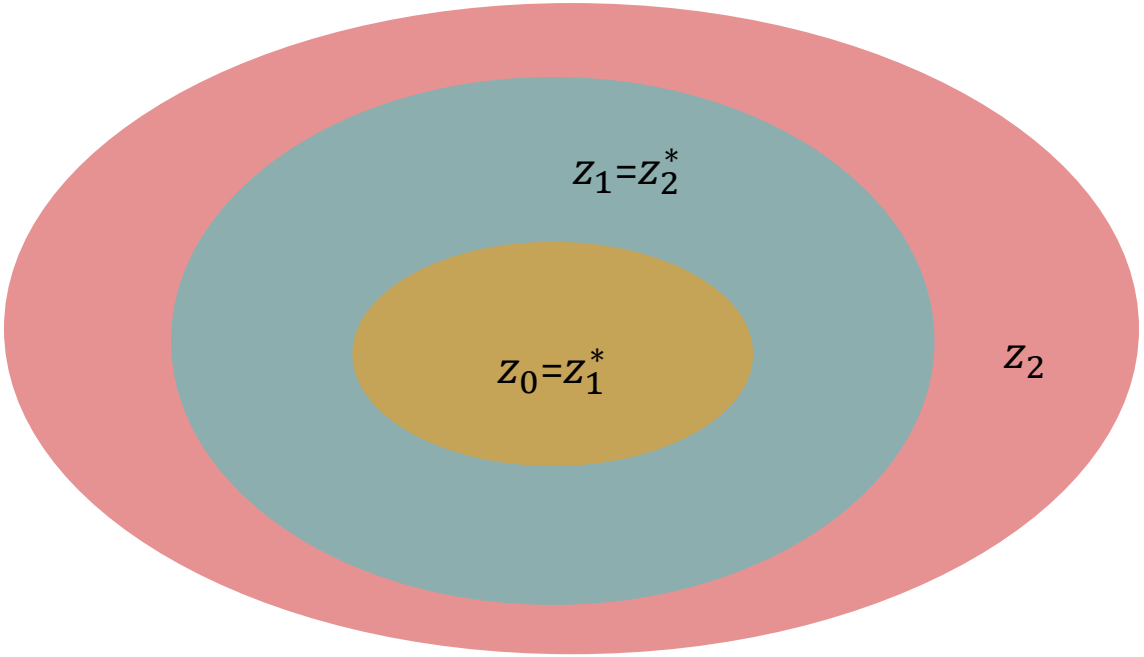
more complicated with more models



The domain of optimization is huge!

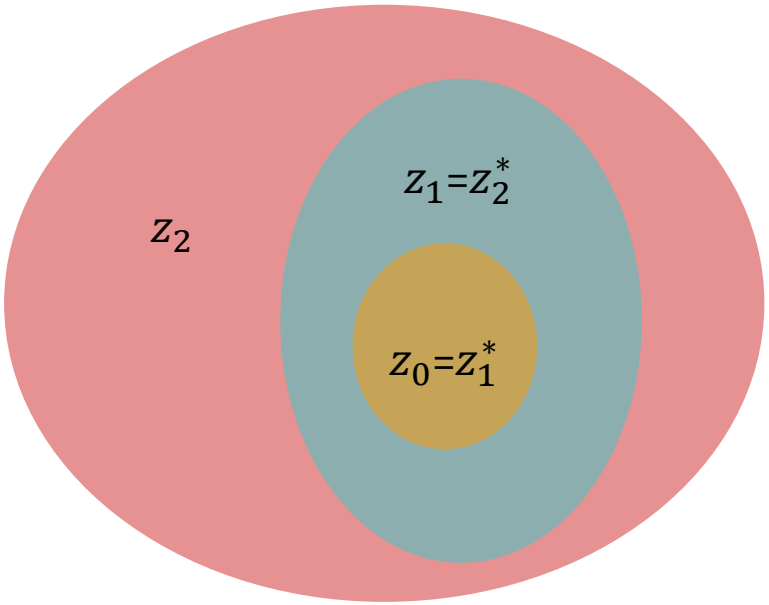
Sub-domain Technique

MFMC

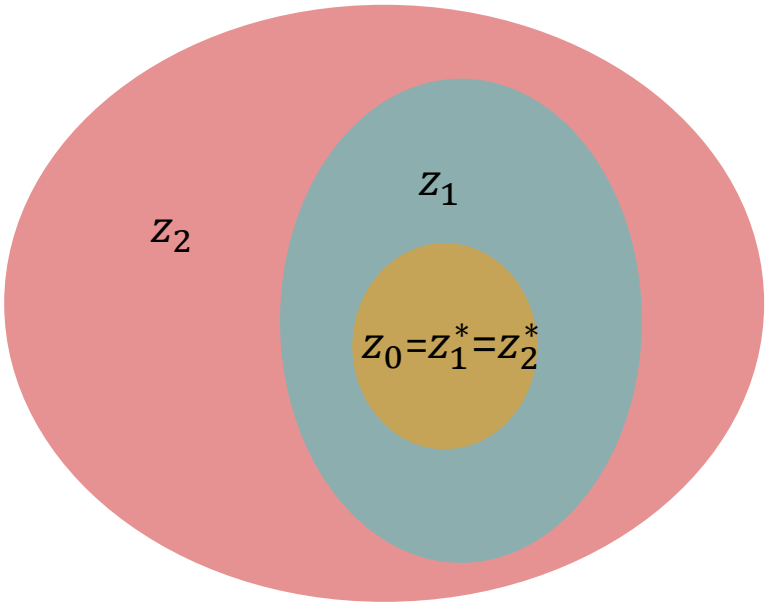


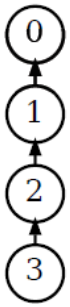
Sub-domain Technique

MFMC

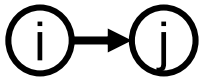


ACVMF

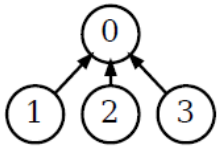




$z_i^* = z_j$

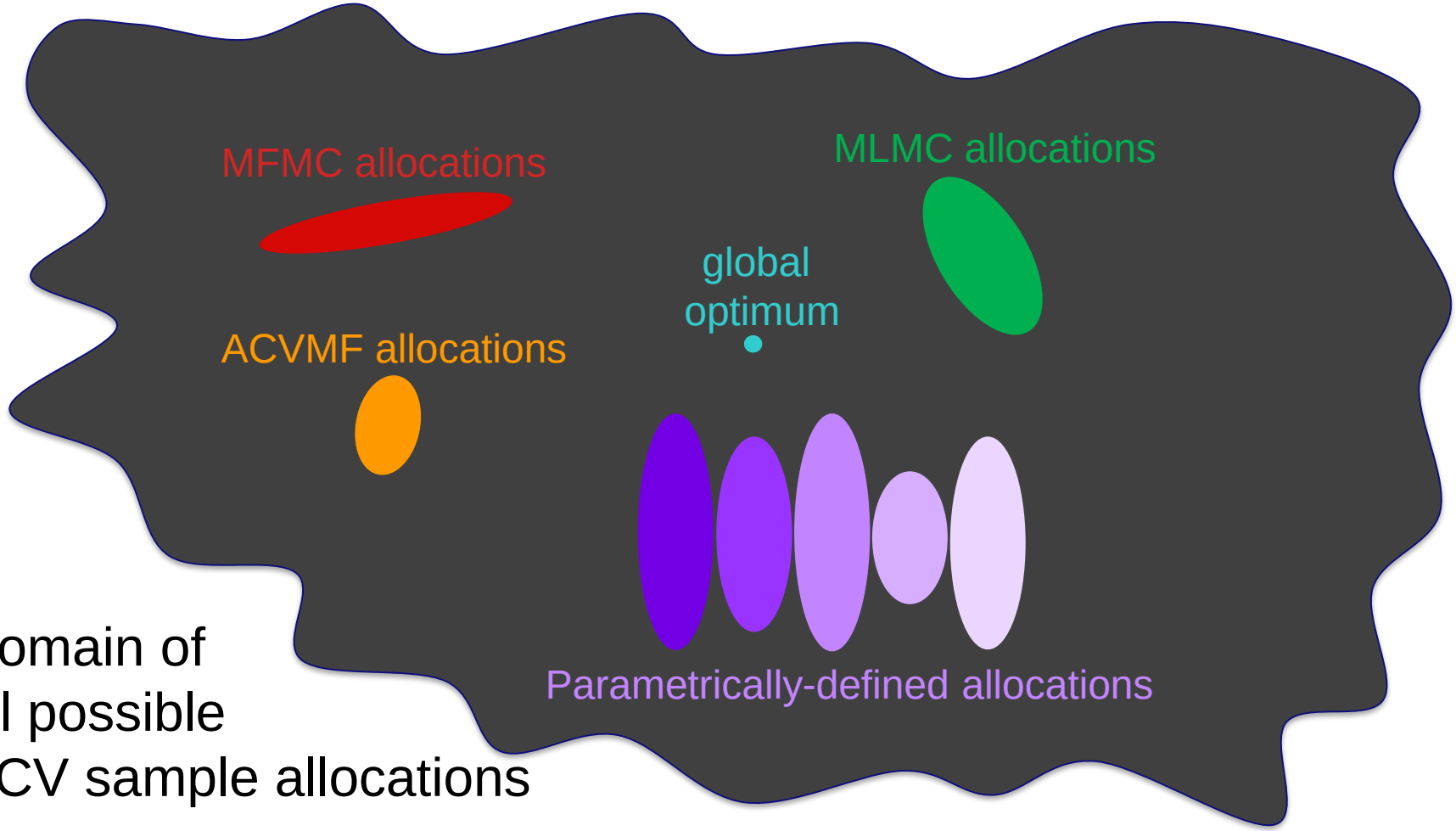


defines a tree structure



Key: sub-domain optimization is much easier

Parametrically-defined ACV Estimators

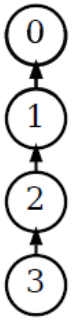


Domain of all possible ACV sample allocations

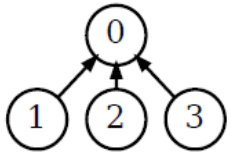
Tunable Recursion

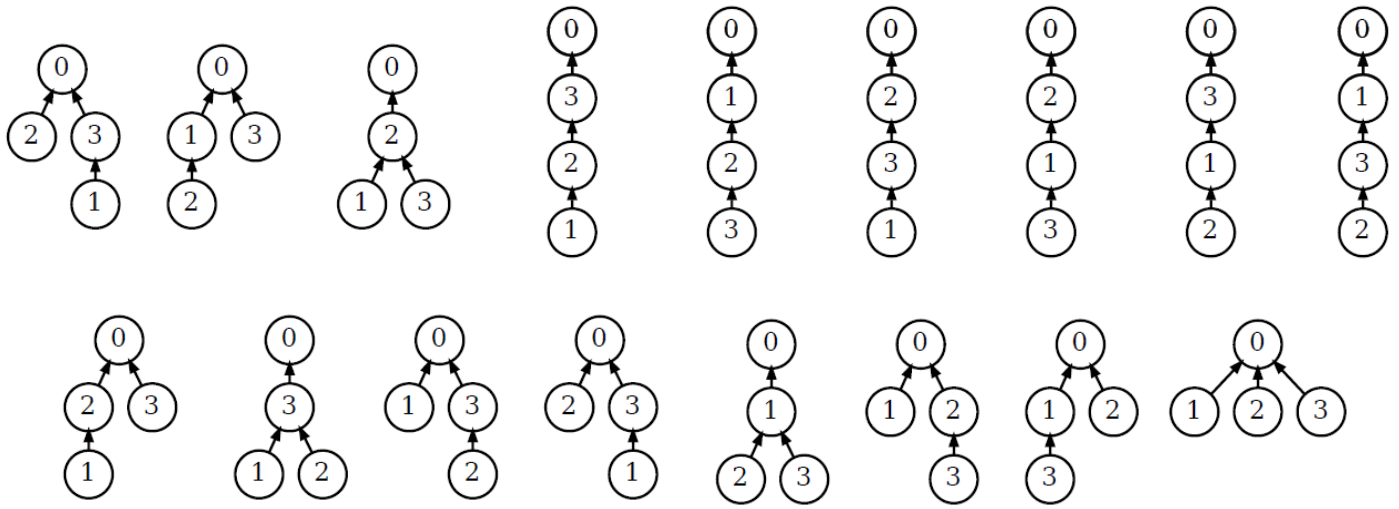
- $z_i^* = z_j$ defined trees
- Search all possible trees!

MFMC / MLMC



ACVMF / ACVIS

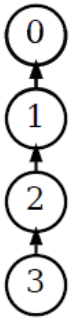




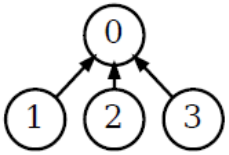
Tunable Recursion

- $z_i^* = z_j$ defined trees
- Search all possible trees!

MFMC / MLMC



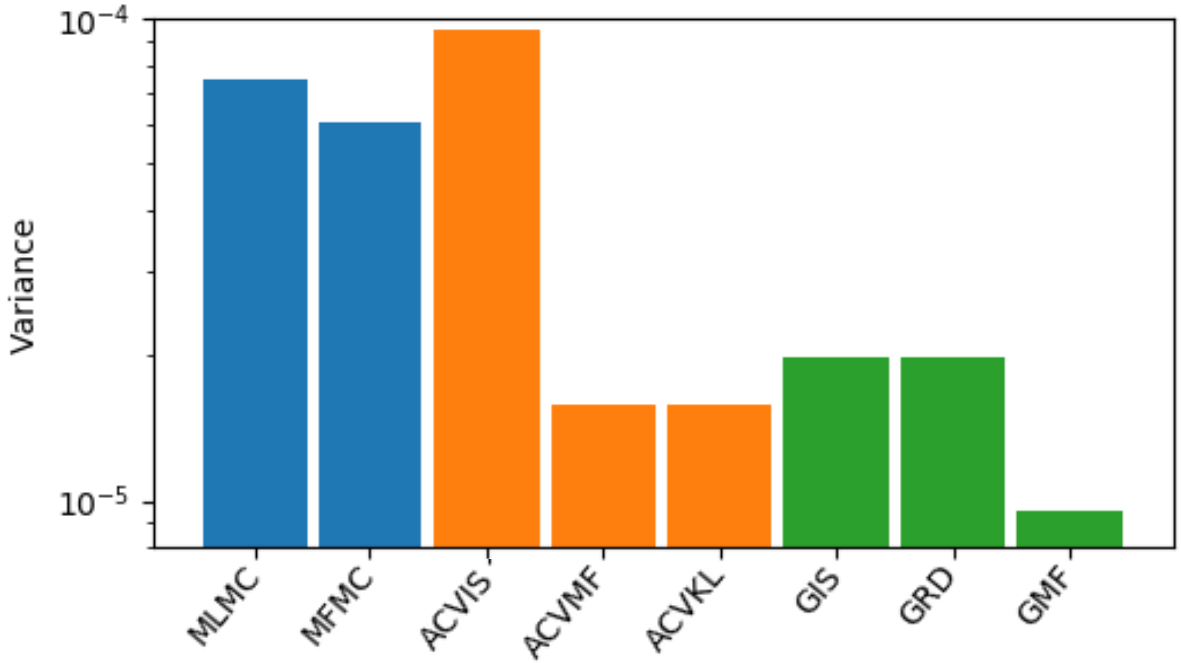
ACVMF / ACVIS



- Must decide how to structure z_i
- All tree structures + sampling strategy
  - Generalized Multifidelity (GMF)
  - Generalized Recursive Difference (GRD)
  - Generalized Independent Samples (GIS)
- Generalized methods find optimum of larger domain

Toy Problem

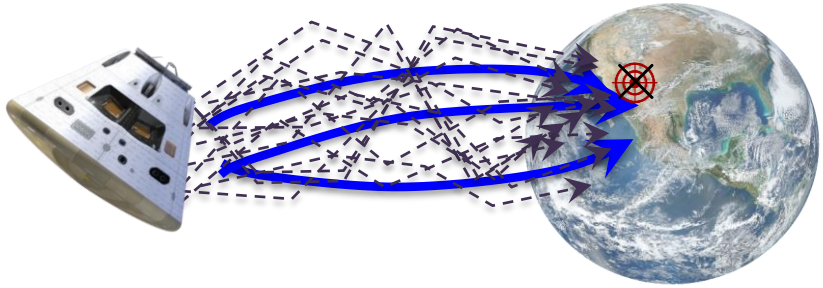
- 5 models: $Q_i(z) = z^{5-i}$
- Input distribution: $z \sim U(0,1)$
- Model costs: $w_0 = 1, w_i = 10^{-i-1}$
- Cost budget: $\hat{w} = 20$

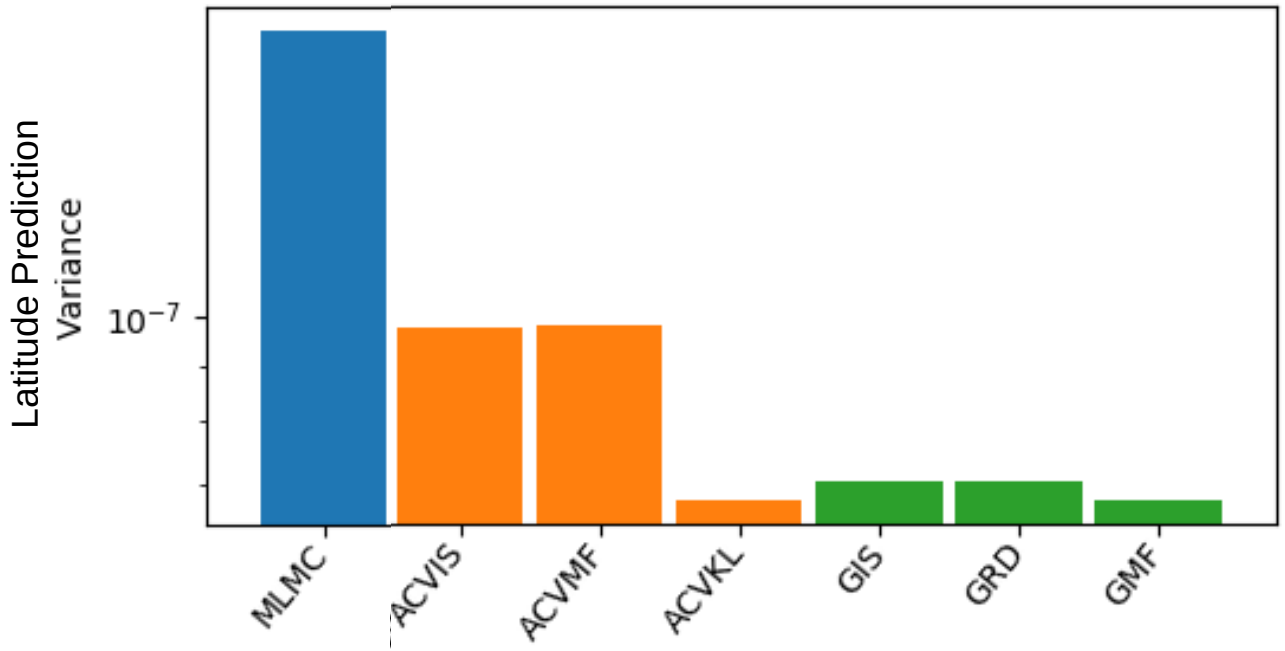


Trajectory Simulation Example

4 Models

- Trajectory Simulator
- Increased time integration
- Decreased physics
- Machine learning surrogate

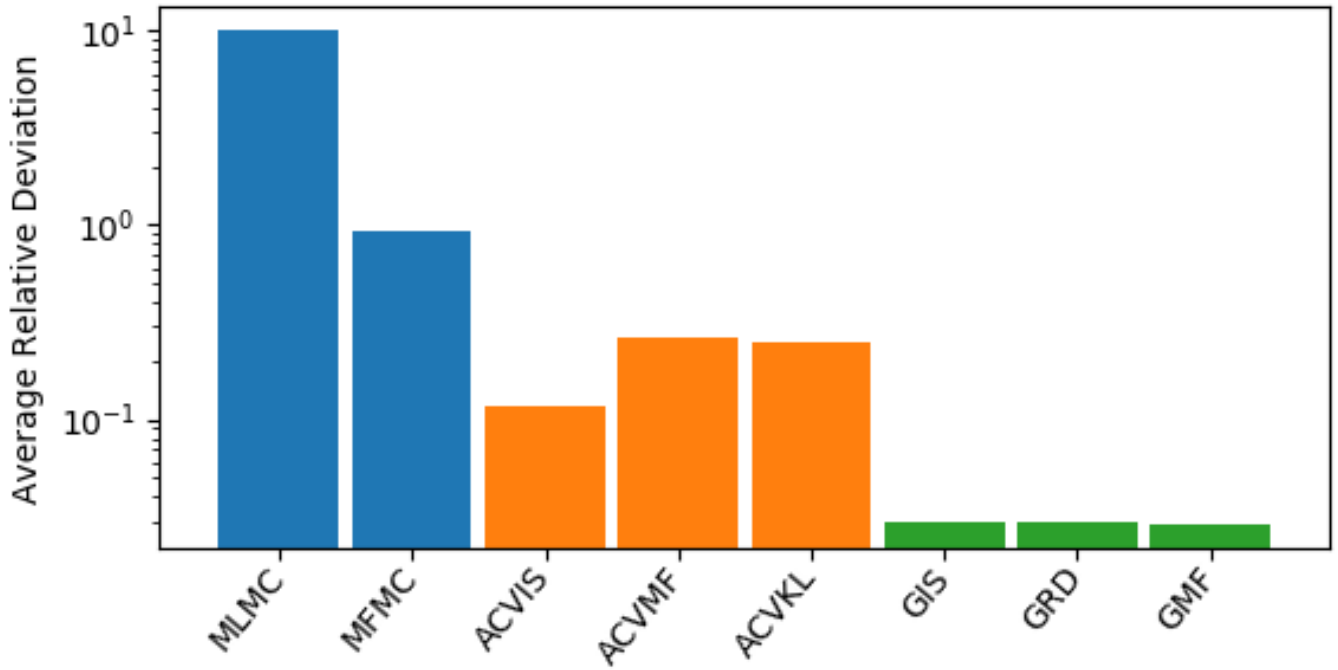




Random Model Scenarios

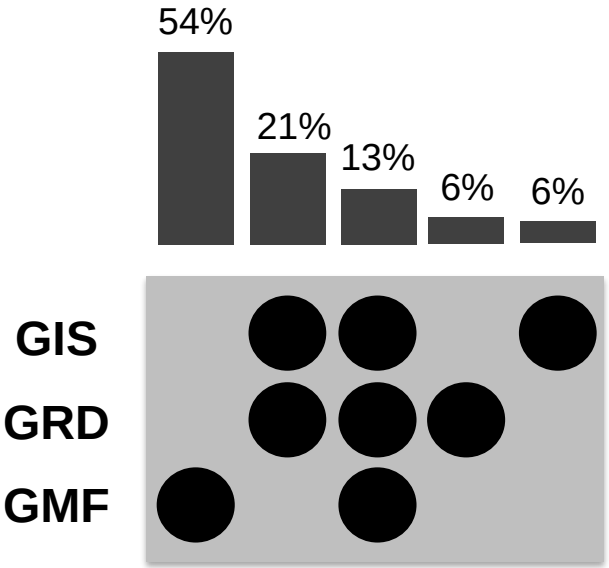
~5 Million

1. Choose number of models [2-6]
2. Sample random correlation matrix
3. Sample random variances
4. Sample random model costs (relative to hifi cost)
5. Perform allocation optimization for all algorithms (fixed cost budget)

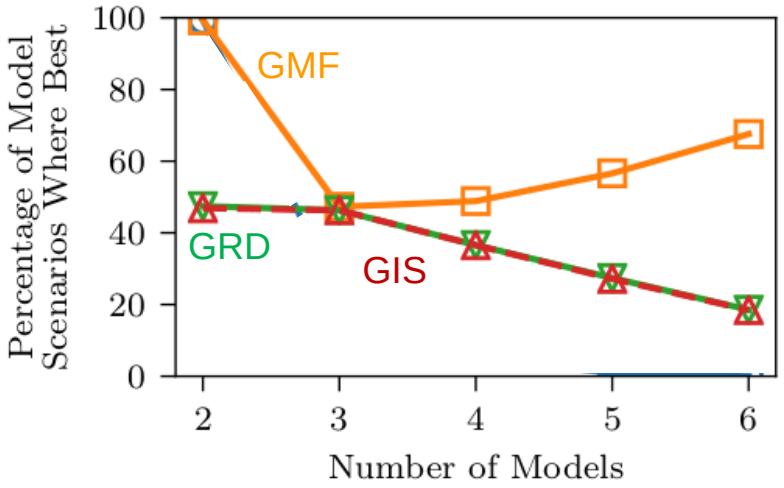


~5 Million Random Model Scenarios

Comparison of sampling strategies



Best Algorithm



GMF best 67% of model scenarios

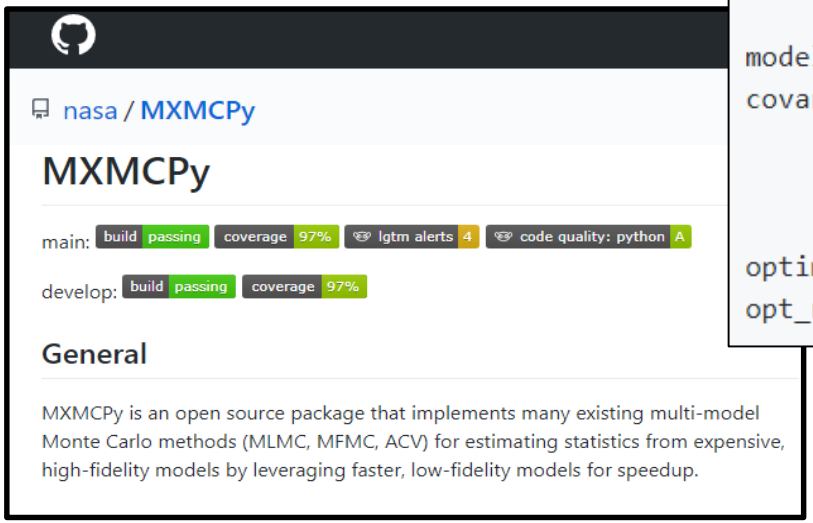
GMF scales better to more models

Summary and Conclusions

- Parametrically-defined ACVs enable optimization over an increased domain
- Parametrically-defined ACVs (GMF, GRD, GIS) give more optimal sample allocations than existing ACVs
  - at worst, the same
- GMF resulted in the best estimator for 2 test cases and 67% of all random model scenarios
- GMF becomes increasingly dominant with more models

MXMCPy

- github.com/nasa/MXMCPy
- Simple interface to
  - compare and use different algorithms
  - implement new estimators




```python
import numpy as np
from mxmc import Optimizer

model_costs = np.array([1.0, 0.05, 0.001])
covariance_matrix = np.array([[11.531, 11.523, 12.304],
                              [11.523, 11.518, 12.350],
                              [12.304, 12.350, 14.333]])

optimizer = Optimizer(model_costs, covariance_matrix)
opt_result = optimizer.optimize(algorithm="mfmc", target_cost=1000)
```



Thank You

geoffrey.f.bomarito@nasa.gov

Check out our preprint on this work:

Bomarito, Geoffrey F., et al. "On the Optimization of Approximate Control Variates with Parametrically Defined Estimators." arXiv preprint arXiv:2012.02750 (2020).