Simulation-based Test Functions for Optimization Algorithms

GECCO 2017, GECH Track

Martin Zaefferer, Andreas Fischbach,
Boris Naujoks, Thomas Bartz-Beielstein

18.07.2017

SYNERGY
Horizon 2020
GA No 692286

Table of Contents

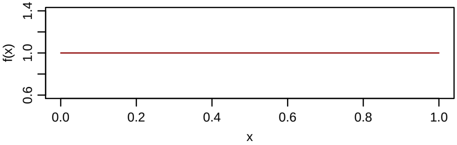
We need test functions to ...

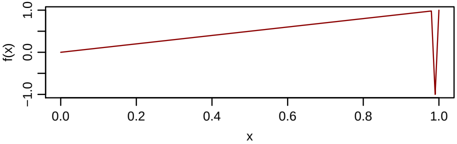
- analyze
- understand
- compare
- design / configure / tune

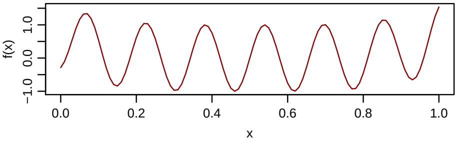
... algorithms

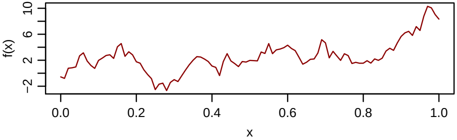
Test functions should be (sufficiently) ...

- difficult
- diverse
- flexible
- relevant
- cheap to evaluate









Existing Approach: Model-based

1. Collect data from real-world problem
2. Learn structure via model (e.g., Kriging / Gaussian processes)
3. Vary model to generate problem instances
4. Use estimation / prediction as test functions

Model-based Test Functions

Advantages

- Relevance due to real-world data
- Data more easily accessible
- Nonlinear models yield flexibility
- Variation yields diversity

Disadvantages

- Bias due to data and model
- Unknown characteristics
- Method for variation?
- and ...

Problem

- Most predictors are smoothing
- Desirable, e.g., for surrogate model-based optimization
- Undesirable for test function generation
- Too easy

$\rightarrow$ requirement: non-smoothing

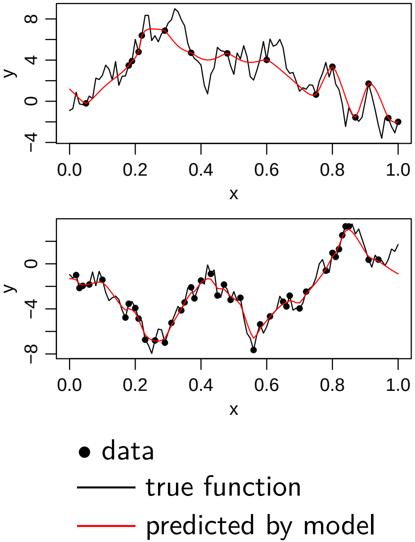
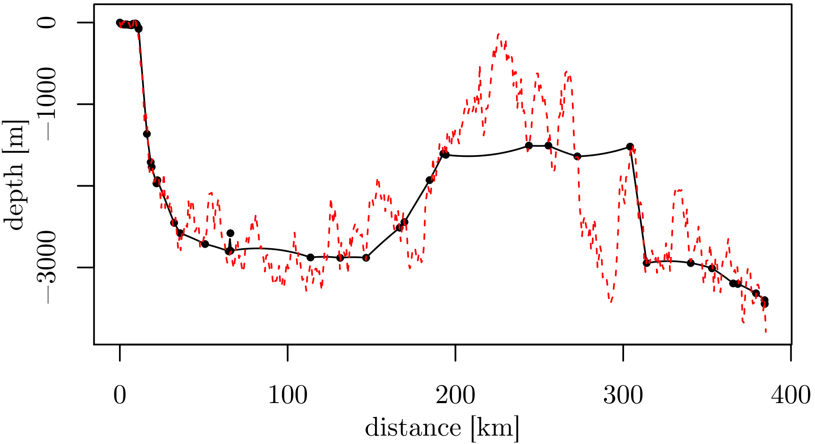


Table of Contents

Proposed Remedy

- *Simulation*
  e.g., $\hat{\mathbf{y}}_{nc} = \mathbf{1}\hat{\mu} + \boldsymbol{C}_s^{1/2}\boldsymbol{\epsilon}$
  instead of *estimation*
  e.g., $\hat{y}(x) = \hat{\mu} + \boldsymbol{k}^T\boldsymbol{K}^{-1}(\mathbf{y} - \mathbf{1}\hat{\mu})$
- Goal of *estimation*:
  - Close to the „true“ value
- Goal of *simulation*:
  - Close to the „true“ process behavior
- Potentially conflicting!

Simulation Example: Undersea Cable



- data
- estimation
- conditional simulation

Simulation-based Test Functions:

Advantages

- Avoid / reduce smoothing
- Reproduces *behavior* of the real-world process
- *Conditional* simulation can reproduce the training data
- Principled approach to generate diverse instances

Disadvantages

- Between simulated samples: interpolation (smoothing, less problematic than with estimation)
- Required number of simulated samples unknown
- Complexity in case of large number of samples

Example

- Training data: 6 samples
- Model simulated at $m = 100$ samples

Example code available at:

`https://martinzaefferer.de`

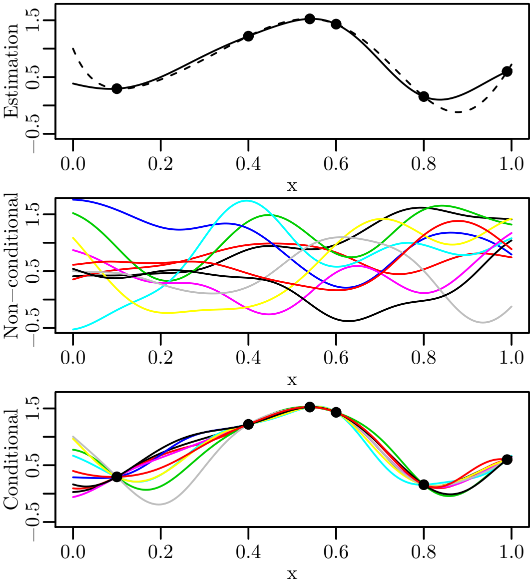


Table of Contents

Case Study: Protein Sequence Optimization

- Freely available data set
- All DNA sequences of length 10
  ACGTAACGGT, CGTAAGATTC, ...
- Objective/Fitness: maximize affinity to fluorescent protein (APC)
- Kriging model trained with 100 sequences
- Simulated for $m = 1000$ sequences

Results: Landscape Analysis

- Correlation length
    - True: 4.5
    - Model: 4.48

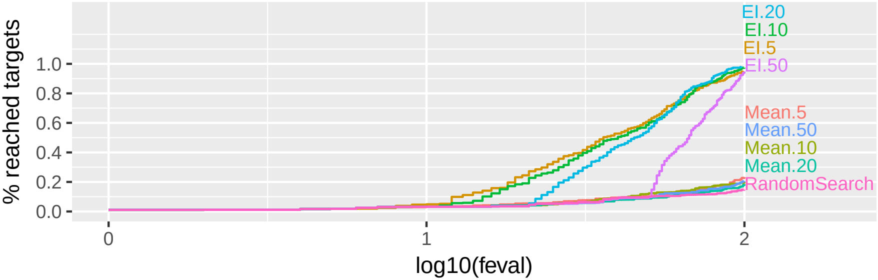
- Fitness distance correlation
    - True: -0.32
    - Model: -0.37 (+/- 0.09)

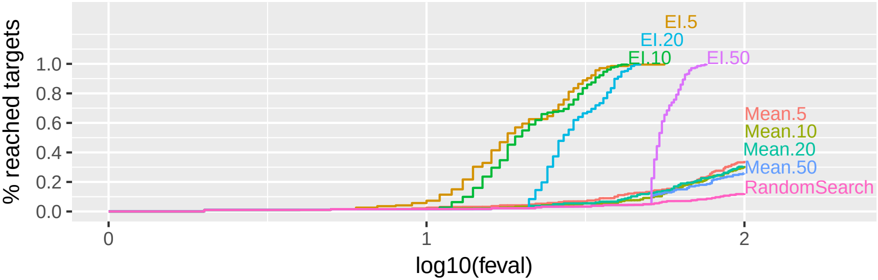
- Number of local optima
    - True: 6805
    - Model: $\leq$ 49

Results: Landscape Analysis

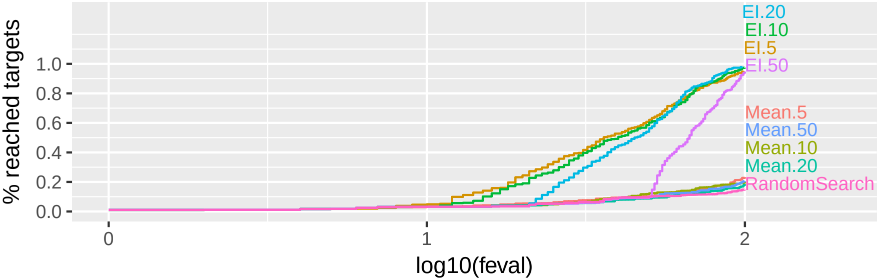
- Interpolating $m = 1000$ samples yields too much smoothness
  - but: better than interpolating the 100 training samples
- Larger m required, computational issues
- Workaround:
  - Restrict to subspace: fix end of sequences to ACGTA
  - Simulate all sequences in the subspace
    - True: 16 local optima
    - Estimation: 2 local optima
    - Simulation: 10 - 19 local optima ✓

Performance: True vs. Estimation





Performance: True vs. Simulation



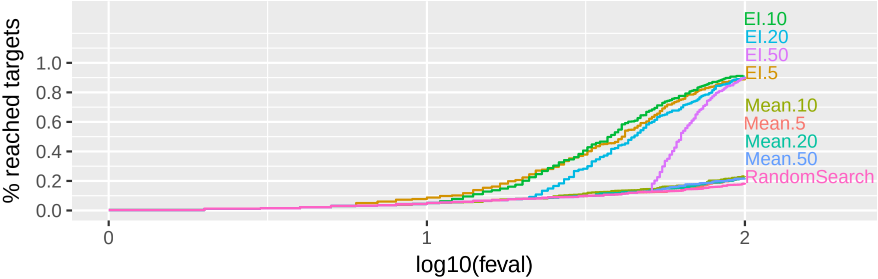


Table of Contents

Take Home Message

- Model-based test functions can produce difficult, diverse, flexible, relevant and cheap to evaluate test functions
- Use (conditional) simulation - not estimation
- If performance on **similar problems** is of interest: simulation
- If performance on **potential realizations of the same problem** is of interest: conditional simulation

Open Issues

- How many simulated samples (m) are needed for a certain problem?

- What to do if m grows too large?

Thanks for Listening

- Questions? Remarks?

PS: You can find the employed modeling tools in the package CEGO on CRAN: `https://cran.r-project.org/package=CEGO`. Check the earlier described 1-dimensional example to see how it works.