Secondmind

Benchmarking Bayesian optimisation

Ablation study, global performance assessment and improvements based on trust regions

Victor Picheny

joint work with: **Y. Diouane, R. Le Riche, A. Scotto Di Perrotolo**

Introduction

Countless algorithms to solve optimisation problems

(Gaussian-process based) Bayesian optimisation (BO): specialises on **expensive black-box** problems

General claim: BO is state-of-the-art for multimodal functions, low budgets, low dimension.

Evidence is weak because BO is mostly compared to itself

...and it is also mostly tested on problems for which the objective function is well-modelled by a GP.

Objective 1:

What happens outside this comfort zone? On which function class and dimension BO is relevant?

Introduction

Large number of BO toolboxes

Consensus on some choices (e.g. Matern 5/2 kernel as default)

Other fundamental choices (e.g. initialisation budget) left open

Objective 2:

Answer some practical and ever-present questions in BO design

Once benchmarking is done...

Objective 3:

Identify weaknesses and improve BO performance

Experimental method

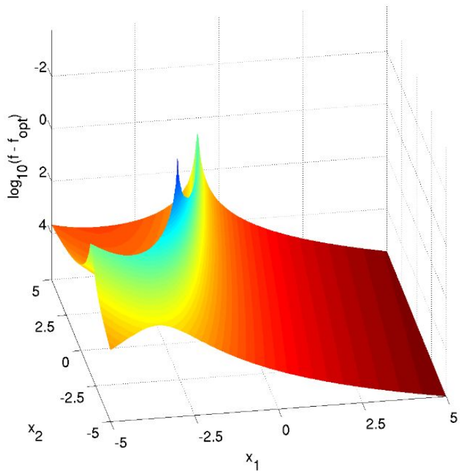
Comparing continuous optimizers (COCO) benchmark

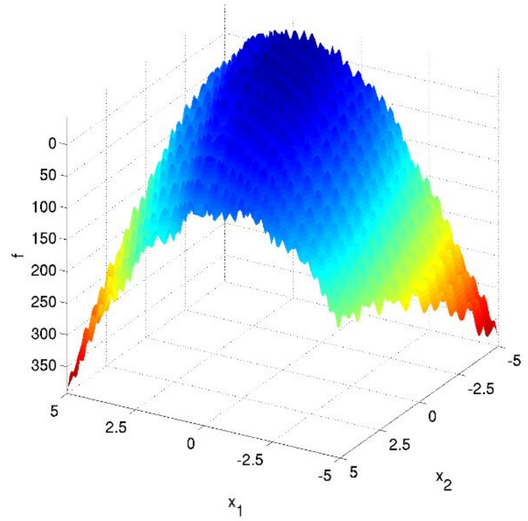
https://coco.gforge.inria.fr/

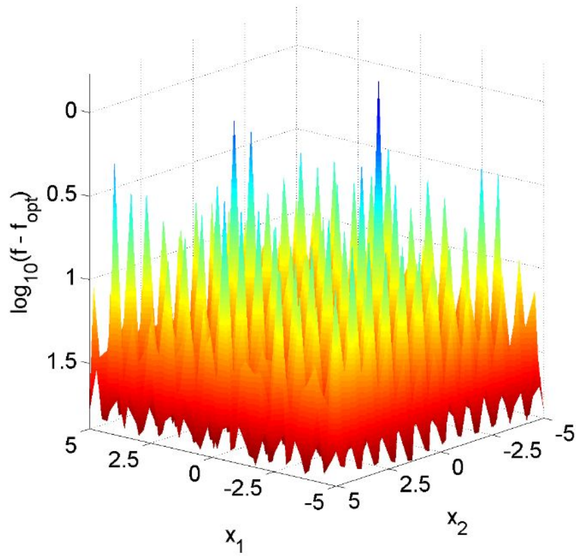
Here: focus on **single objective, noiseless problems**.

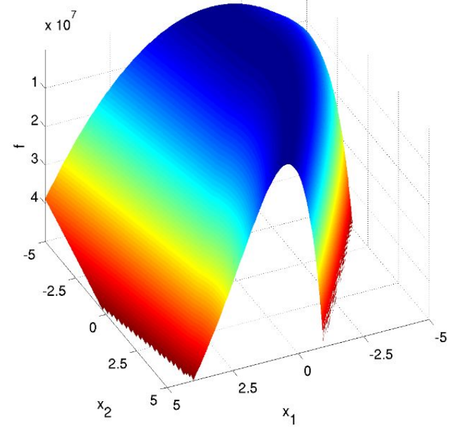
24 challenging problems, divided in groups (e.g. “unimodal with moderate conditioning”).

Each problem has many instances (random rotations, optimum position) to avoid chance effects, and is defined for different numbers of dimensions > 1









Outline

1. Overview of Bayesian Optimisation and research questions
2. Benchmark results
3. Improving performance using trust regions

Secondmind

Overview of BO and design questions

BO relies on Gaussian process (GP) regression

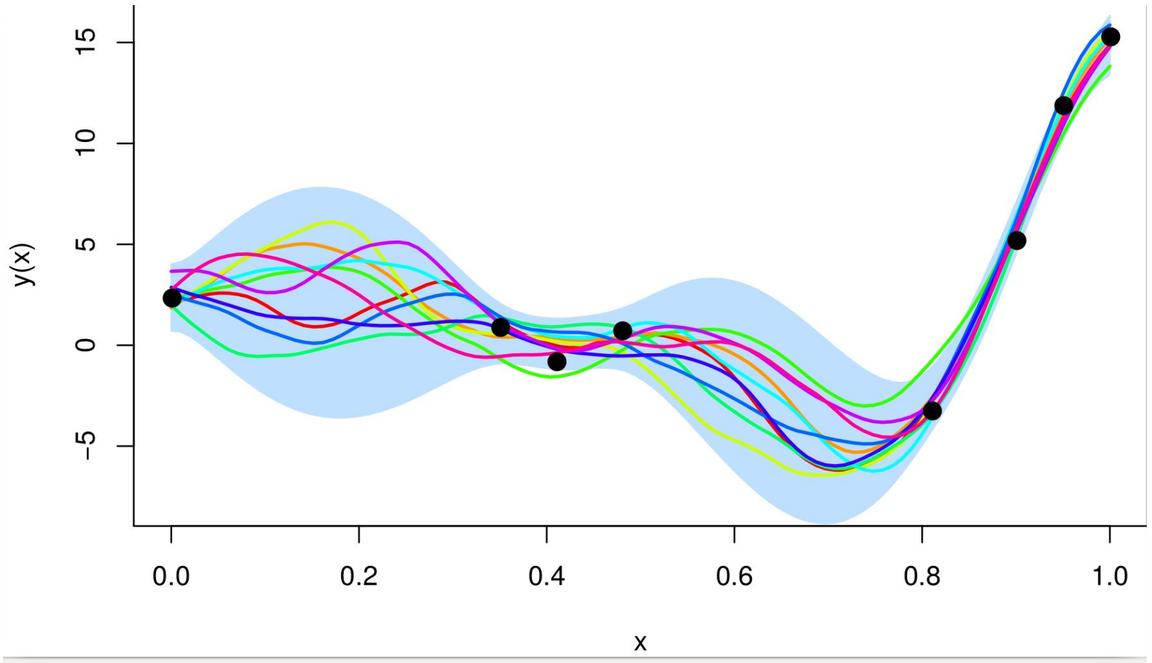
GP prior: characterized by parametric mean and covariance

Question 1: which covariance should we use?

Question 2: which mean should we use?

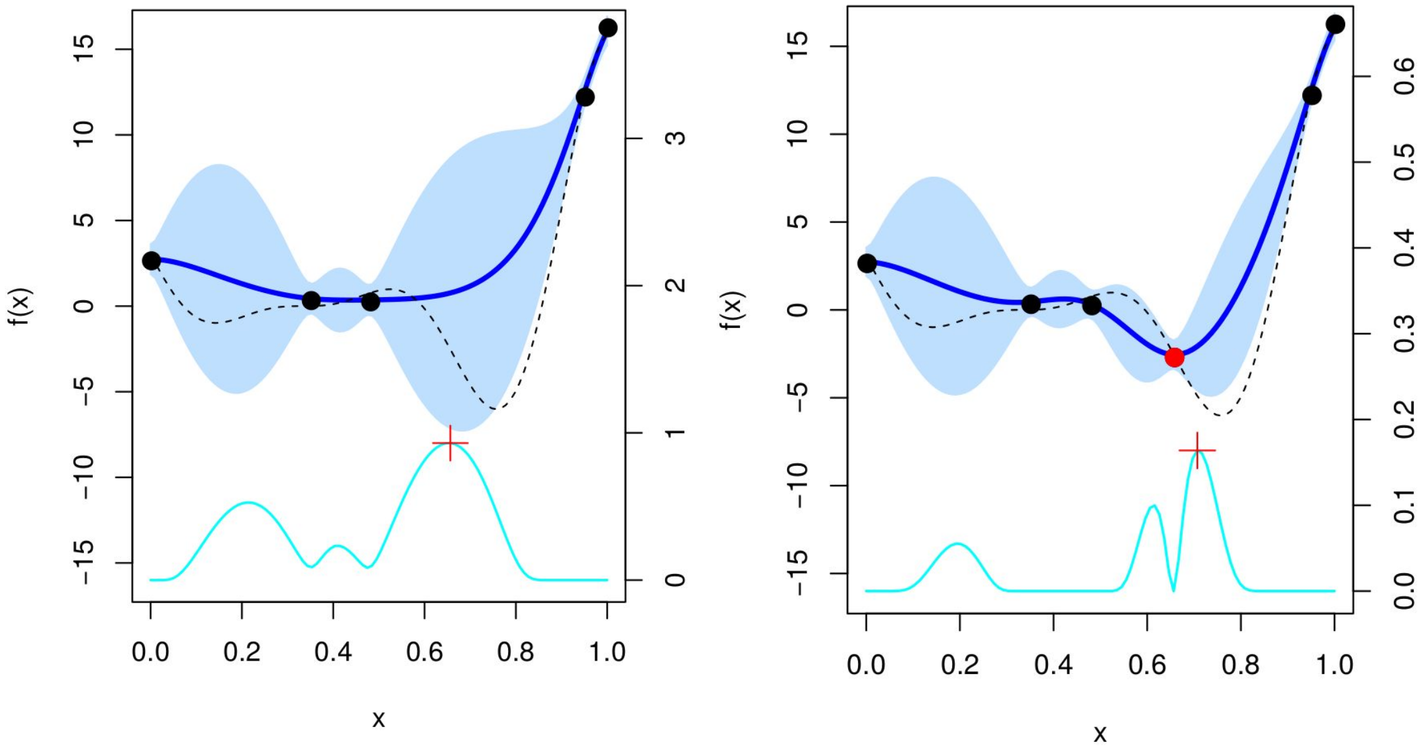
BO starts with a “warm-up” phase, with initial dataset chosen by space filling.

Question 3: how large should this initial set be?



BO iterates by maximising an acquisition function

Here: acquisition function = Expected improvement



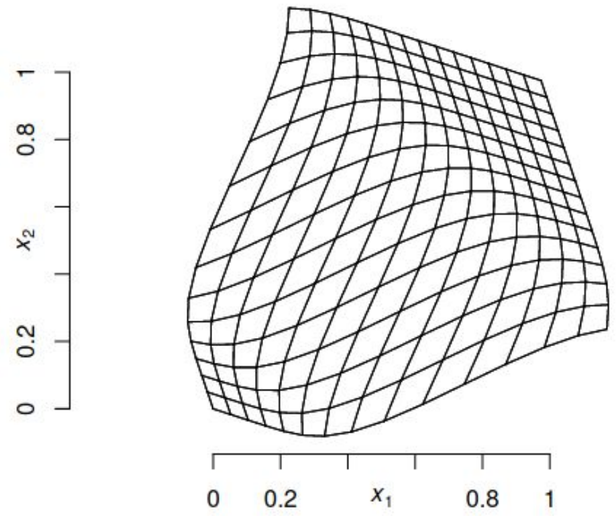
This step is a computational bottleneck...

Question 4: how important is this optimisation?

Other questions of interest

GPs are sensitive to model misspecification (e.g. non Gaussian, non-stationary)

Question 5: can warping (input or output) help?



Input warping,
source: S. Marmin

Secondmind

The COCO experiments

Experimental setup

Implementation: **DiceOptim**

Lots of changes to improve speed and reliability

+ Tricks e.g. clipping lengthscales, avoid poor conditioning, recover from “bad” model, etc.

COCO: 24 functions x 15 instances

Dimensions tested: 2, 3, 5, 10

Budget: 30*dimension

Over 40 BO variants tested (weeks of calculations on a cluster)

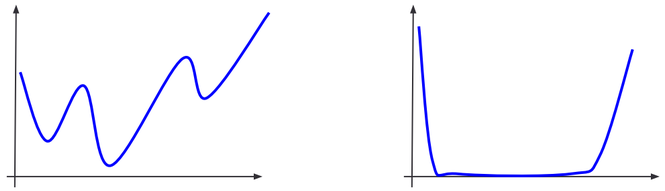
Factors:

- Size of initial DoE: 2d, 7d, 20d
- Kernel: Matern52 or Exp
- Trend: constant, linear, quadratic
- Input warping
- Output warping
- EI optimisation: random search + multistart BFGS, single BFGS, random search

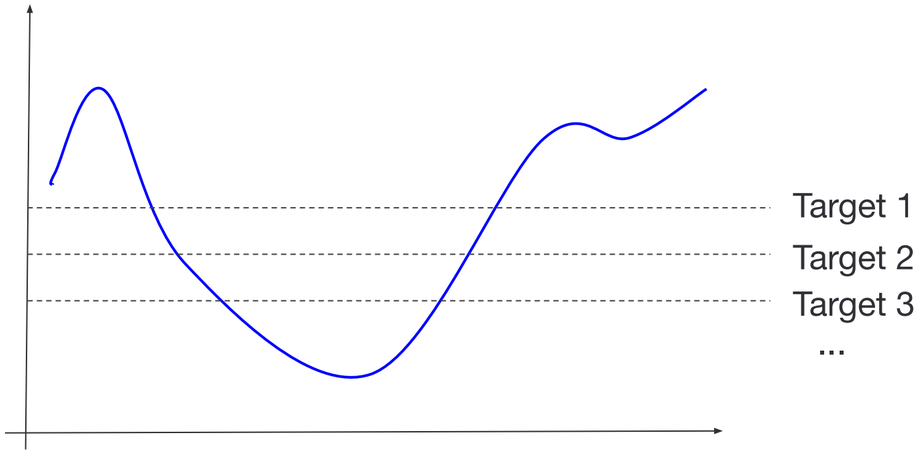
+ some combinations but not all...

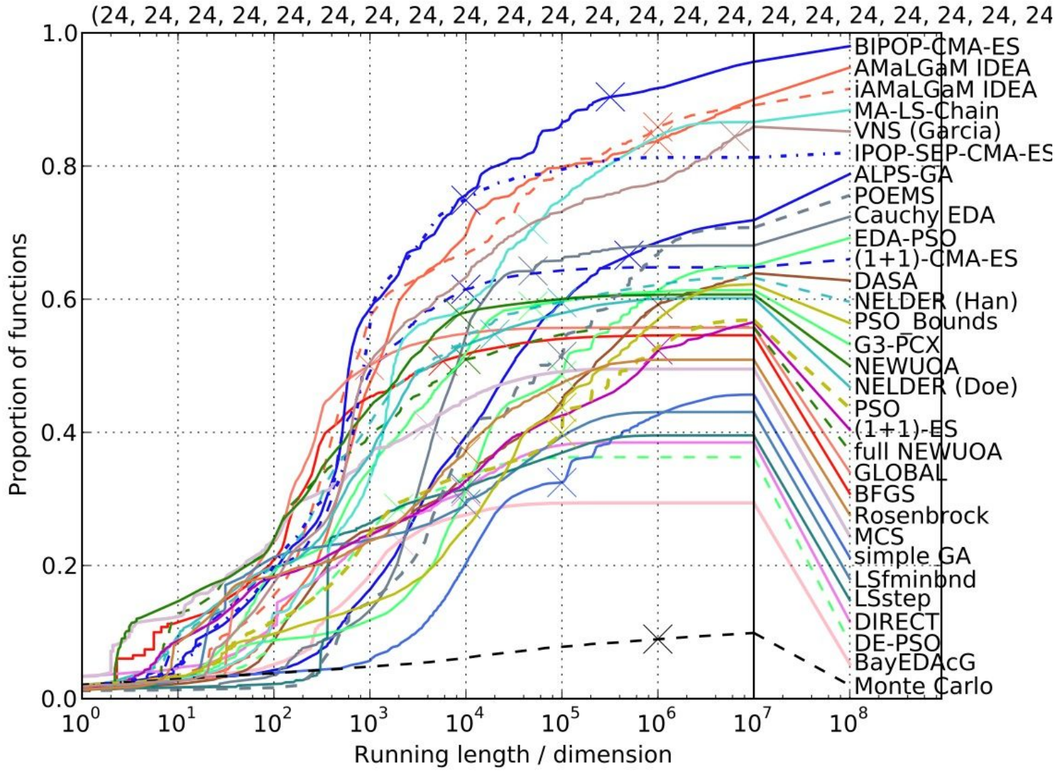
Performance measure: Empirical run time distribution (ERDT)

Difficult to average regret over many problems:



Instead, we use targets to reach. ERDT = proportion of targets reached by an algorithm at a given time.





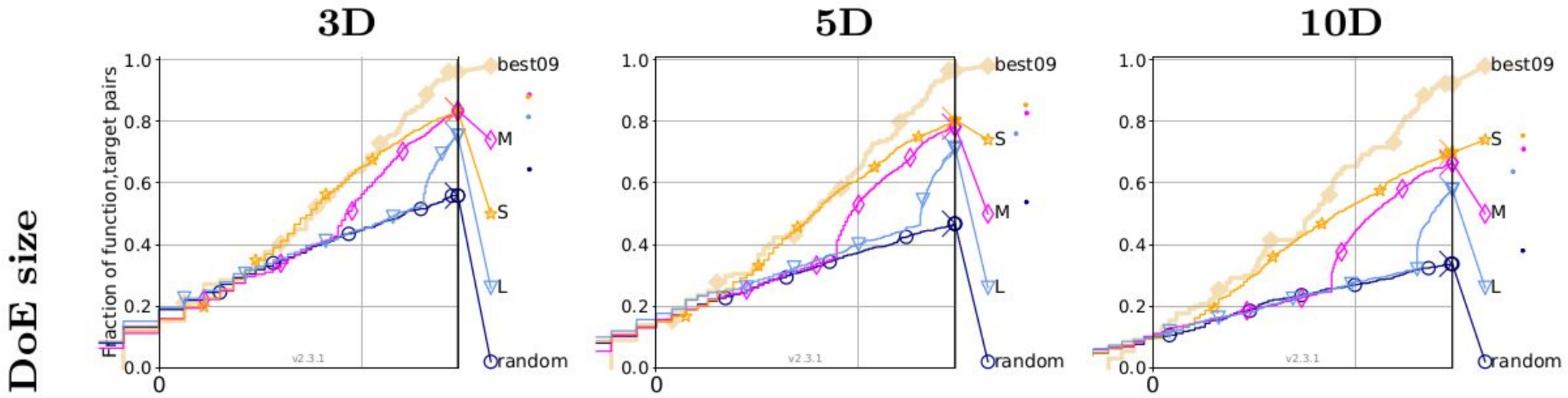
Results (excerpt): initial DoE size

Over all functions, by dimension

best09: utopic algorithm made of the best (for each cost and dimension)of the 32 algorithms competing at BBOB 2009

Random: random search

S / M / L: BO with resp. 2*d, 7.5*d and 20*d initial points

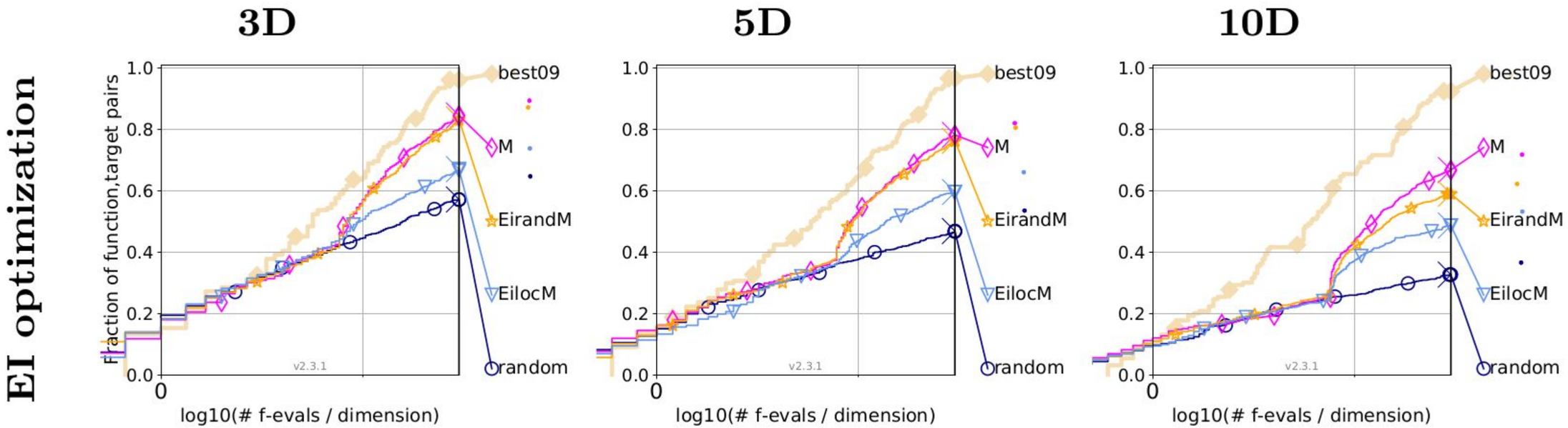


Starting small is **much** better - in line with new practice, not old recommendations (30% of the budget)

Results (excerpt): acquisition function optimisation

Over all functions, by dimension

M: multistart BFGS **EilocM**: single BFGS **EirandM**: random search



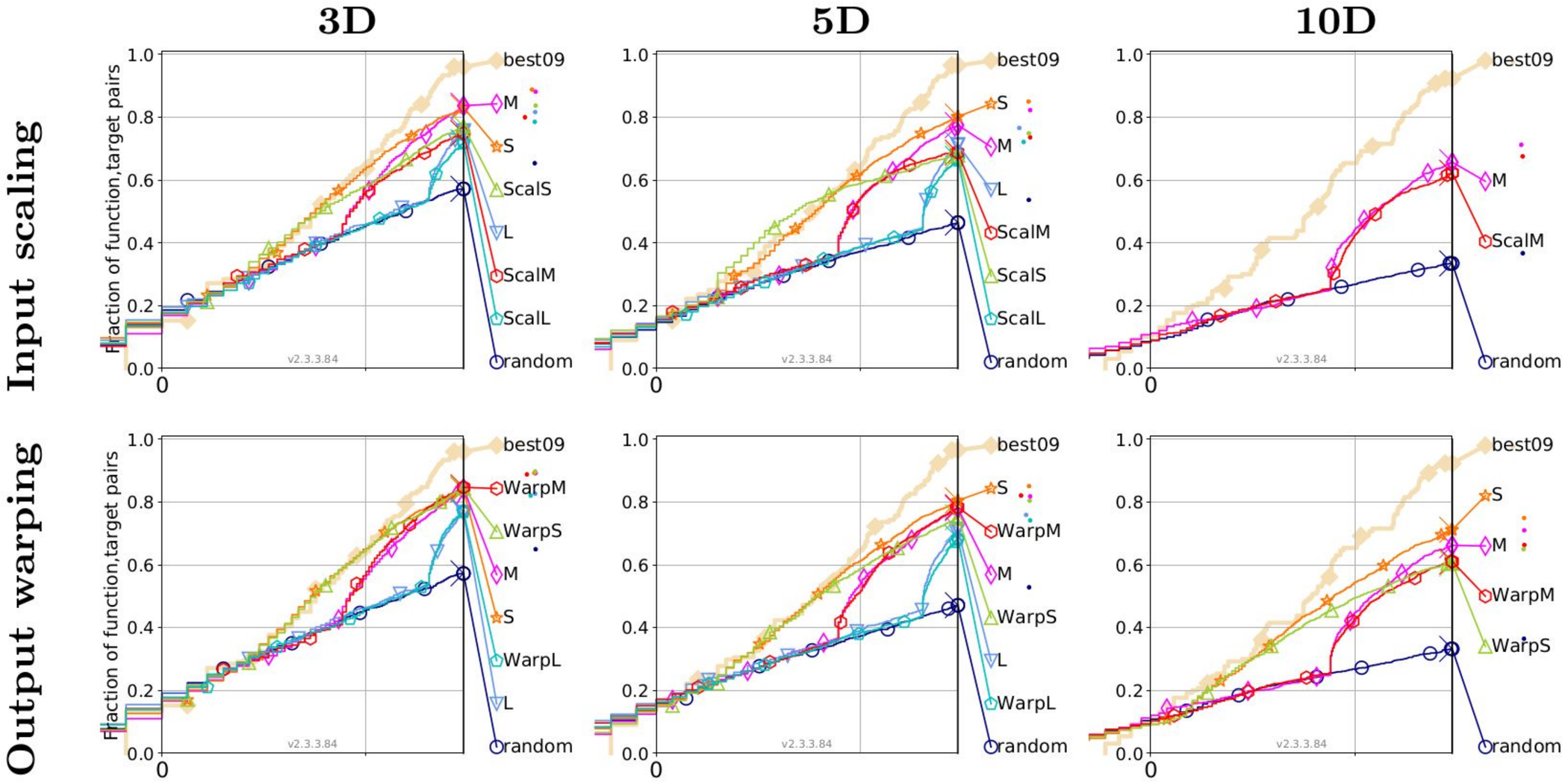
Single BFGS run is very bad

In small dimension: local search isn't even necessary!

In higher dimension: random search scales poorly, gradients help

Results (excerpt): ablation study: warping

Both are generally bad, with a couple of exceptions (on functions, not groups).

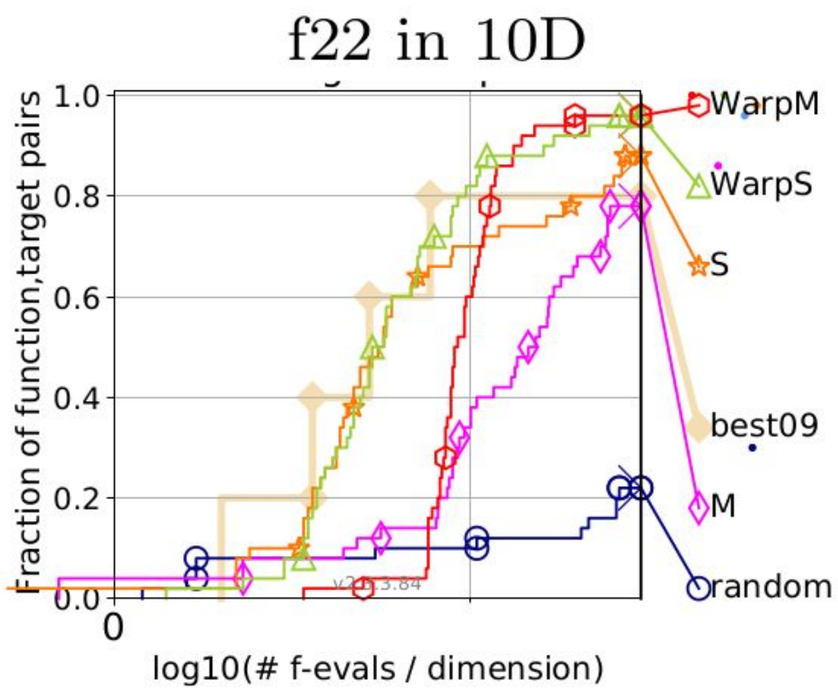


Some findings

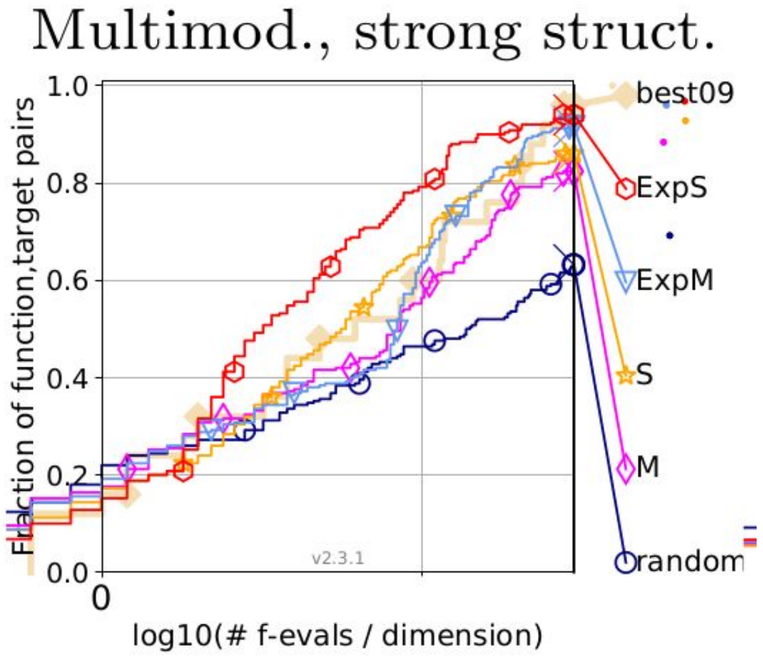
Initial DoE size	Small is always better! (except on super tricky problems, high dim)
Kernel	Matern 5/2 is a good default choice, but Exp quite good on multimodal functions
Trend	(Surprisingly) quadratic trend helps! Mostly on convex functions, but it’s never detrimental
Warping	Not good in general (with exceptions on some functions, not groups)
EI optimisation	Global search >> local search

No free lunch applies...

E.g. warping is surprisingly good on one function



E.g. Exponential kernel outperforms Matern52 on multimodal functions… in 5D only



Comparison with non-BO algorithms

Competitors

Random search, BFGS (for reference)

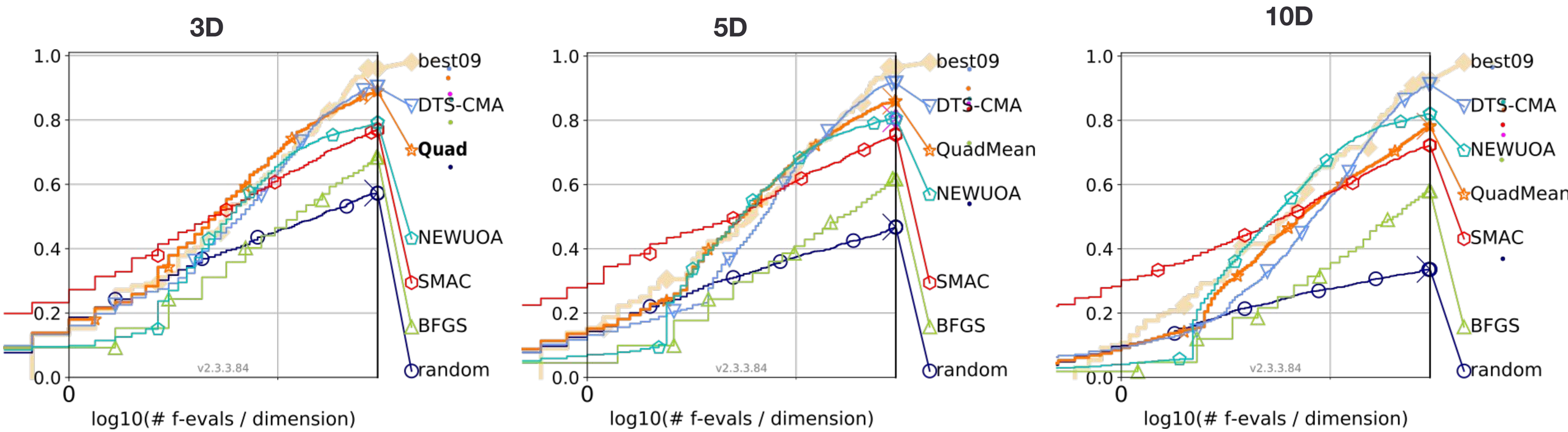
NewUOA: quadratic response surfaces + trust regions

SMAC: BO alternative (based on isotropic covariance, starts from a single point)

DTS-CMA: GP-enhanced evolutionary strategy

Comparison with non-BO algorithms

Over all functions, by dimension



In 3D and 5D:

- our BO (Quad) works best on intermediate budgets (10*d)
- SMAC (BO alternative) has an impressive head start
- DTS-CMA exploits better large budgets

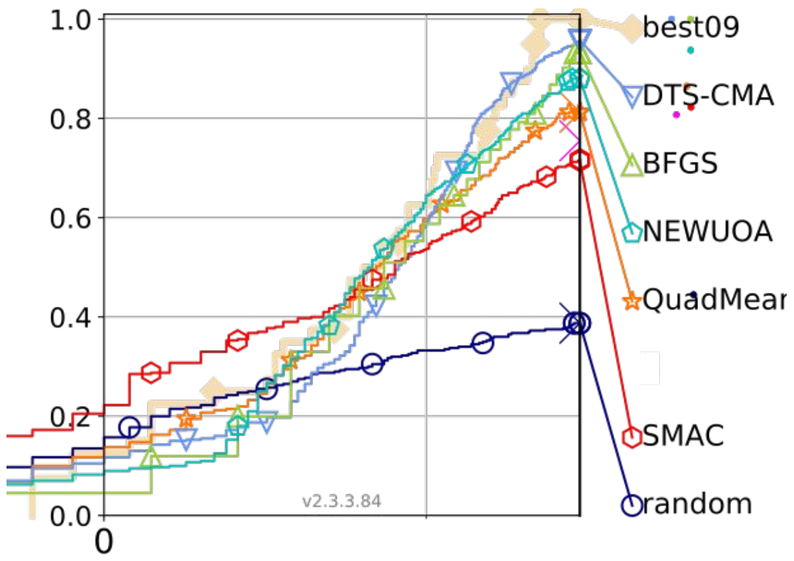
In 10D: BO is not competitive!

SMAC -> NewUOA -> DTS-CMA

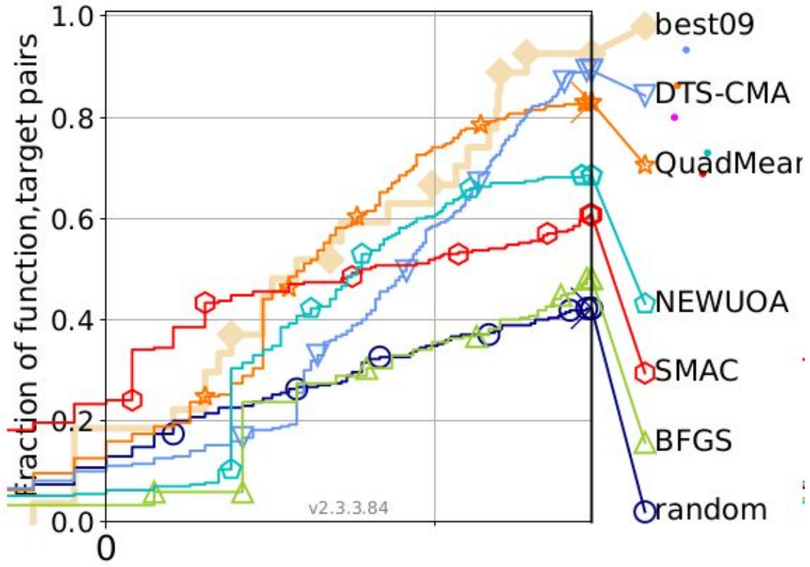
Comparison with non-BO algorithms

By function groups

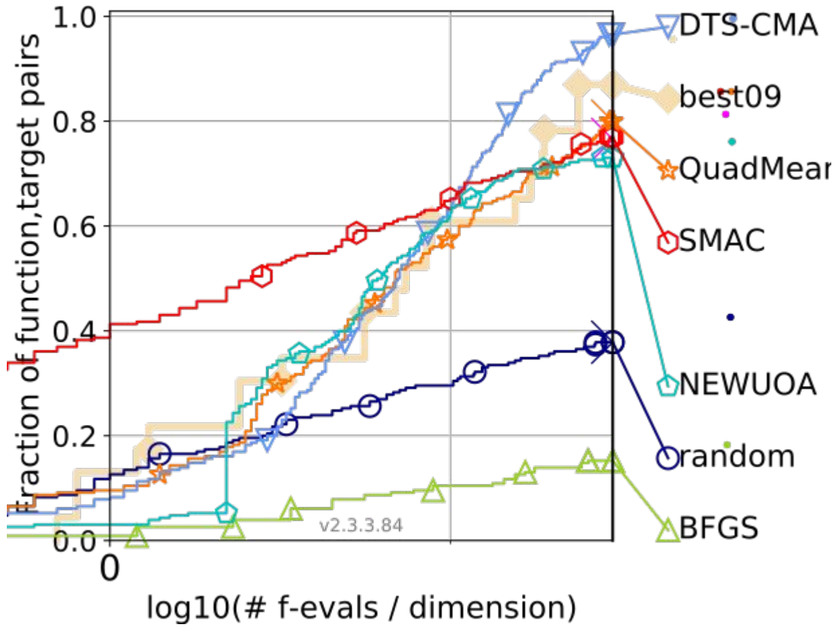
Unimodal, high cond, 5D



Separable, 5D



Multimodal, 10D



Key take-aways

BO strong where expected: low dim, separable, multi-modal functions

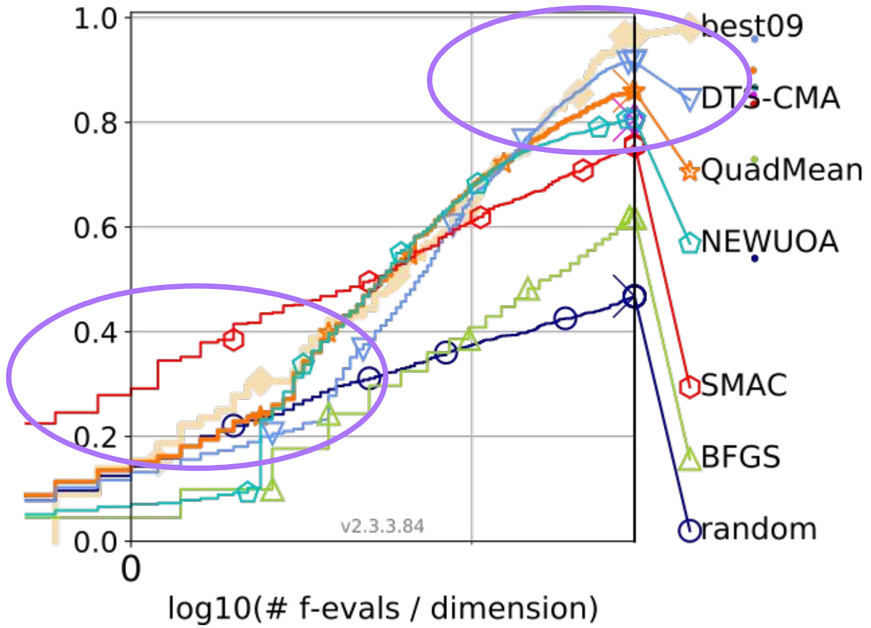
Surprisingly good on some easy functions, eg spherical

Not great on unimodal functions with high conditioning

Not competitive in dimension 10

2 annoying results:

- SMAC starts from 1 point
- DTS-CMA is much better at exploiting GPs with larger budgets

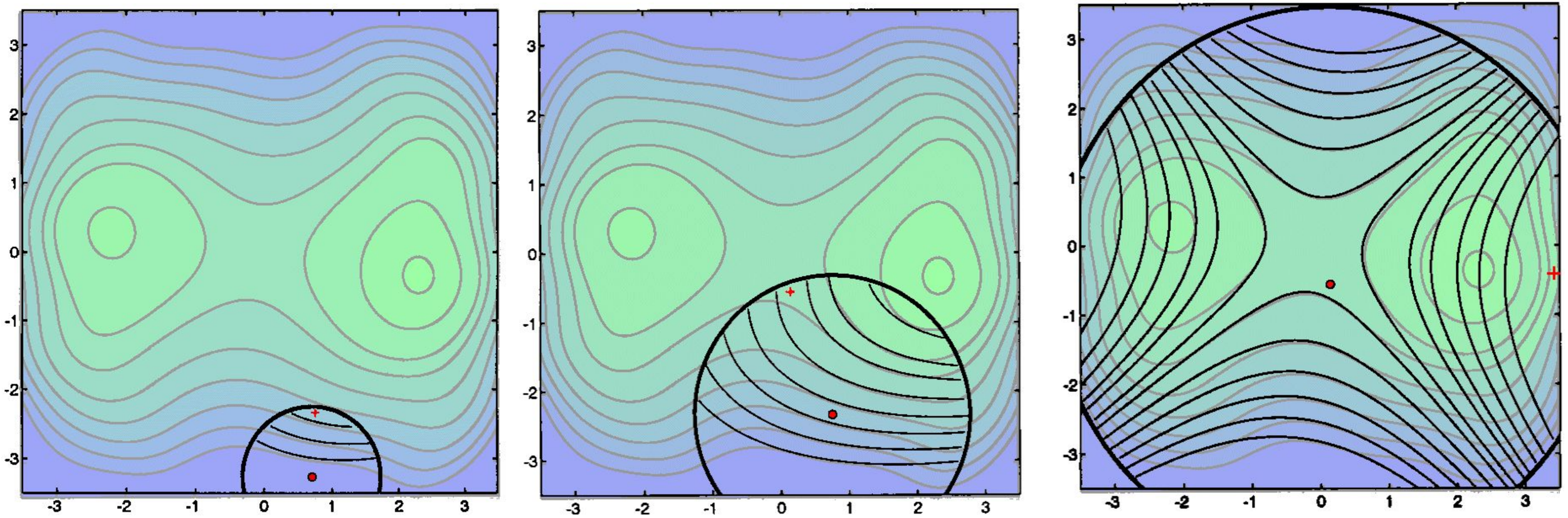


Secondmind

TREGO: a Trust-Region Framework for Efficient Global Optimization

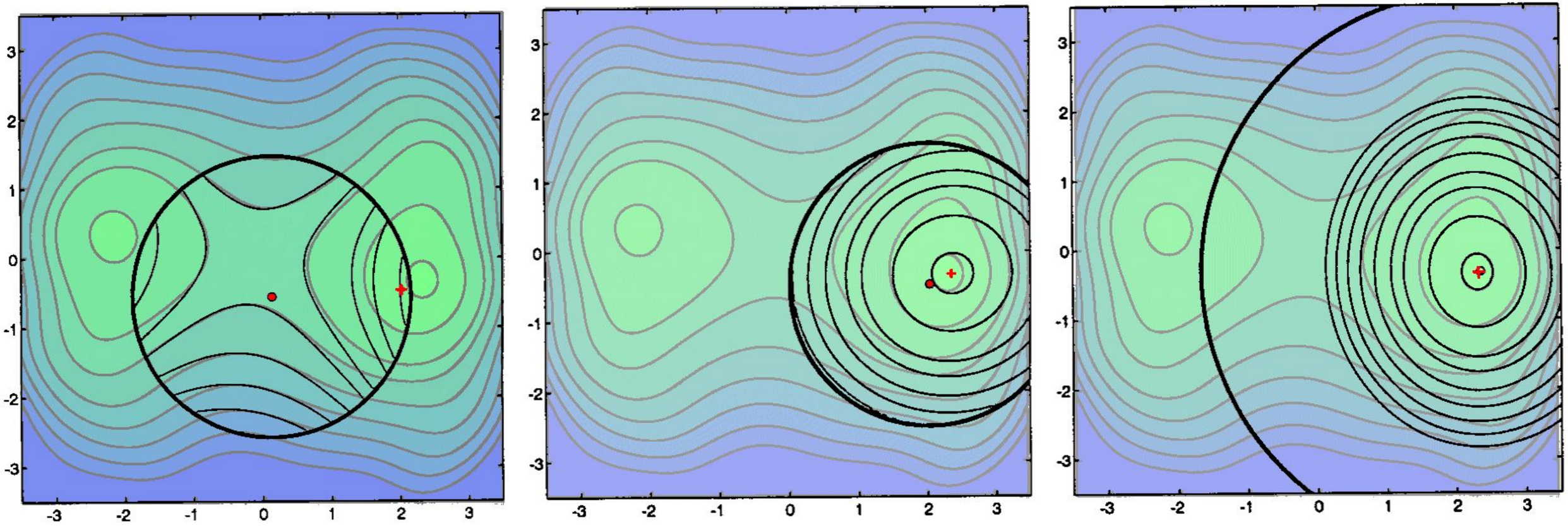
Trust regions = local models + limited search space

When successful: increase trust region



Trust regions

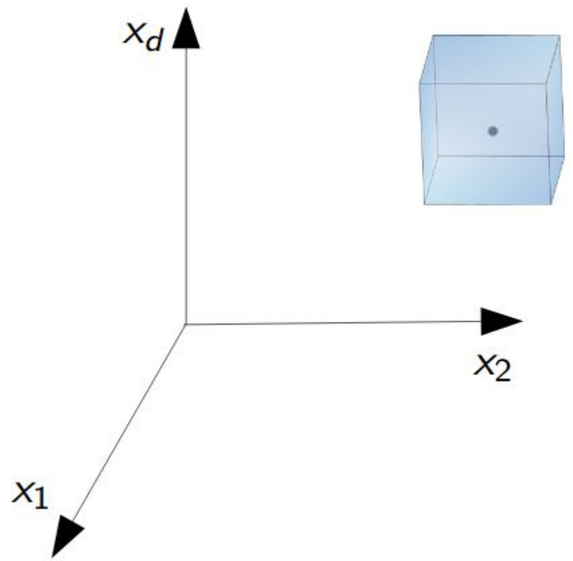
When unsuccessful: decrease trust region



BO and trust regions

Principle: counteract the effect of increasing dimension (volume) by restricting the search to a smaller (controlled) trust region.

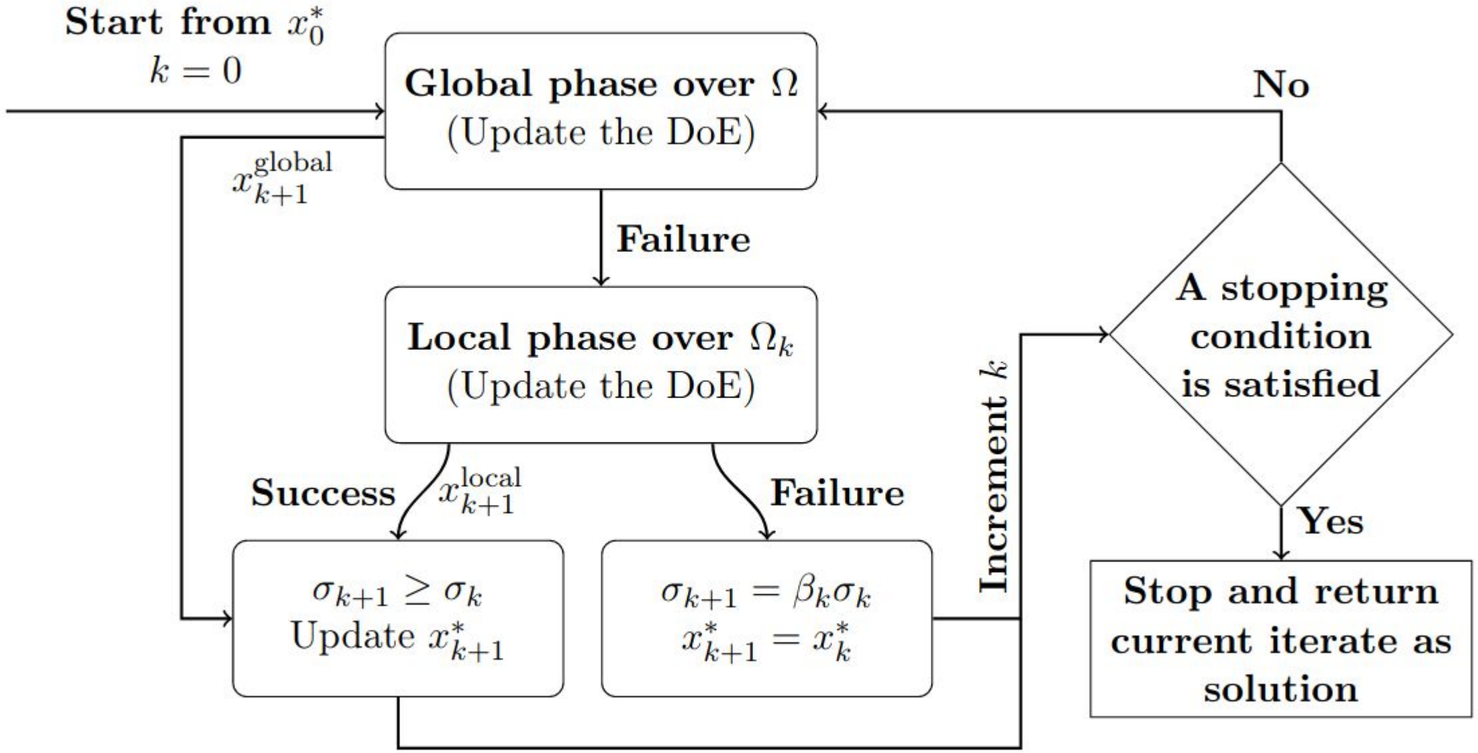
- TRIKE, Trust-Region Implementation in Kriging-based optimization with Expected Improvement, Regis, 2016
- TURBO, a TrUst-Region BO solver, Eriksson et al., 2019
- **TREGO, a Trust-Region framework for EGO, Diouane et al., 2020**



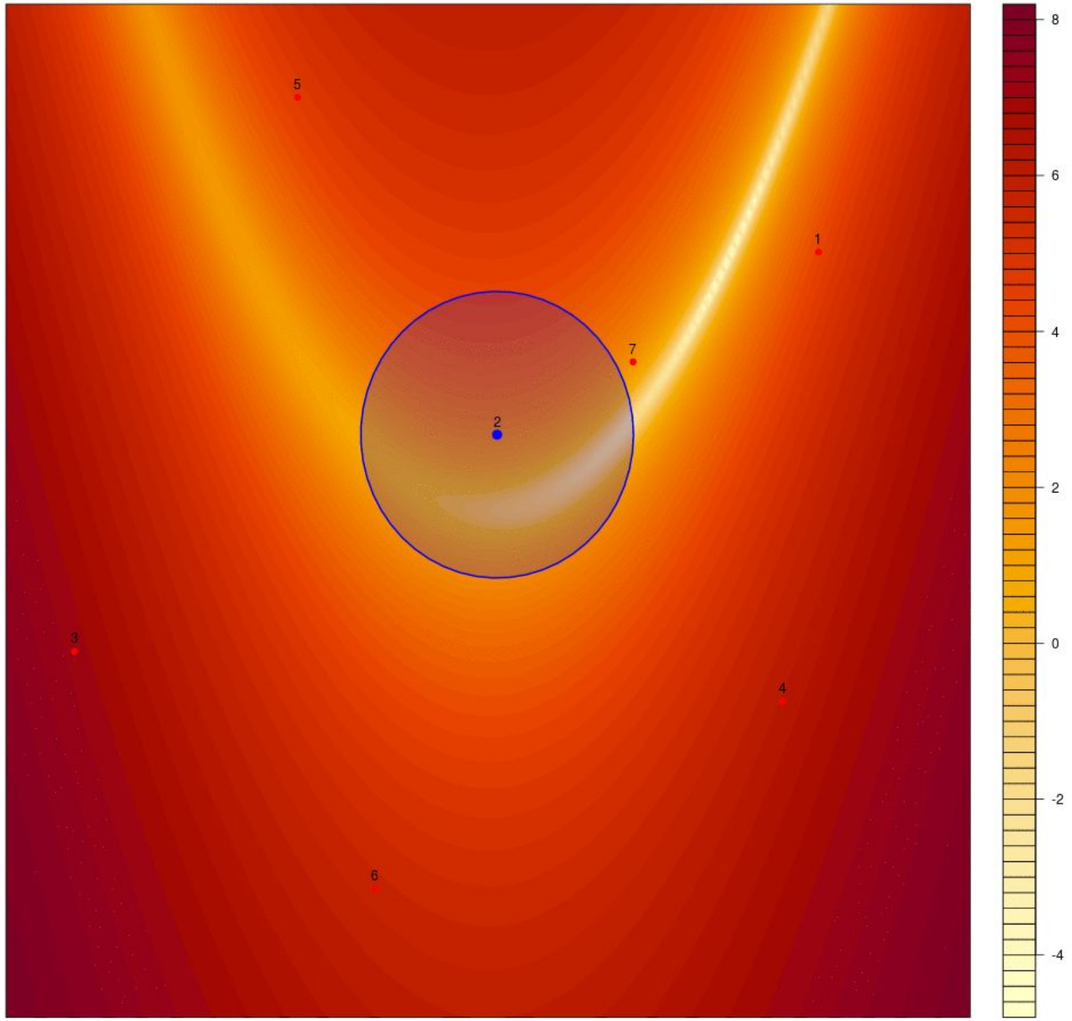
TREGO: a Trust-Region framework for EGO

We always follow a BO strategy, but we alternate between regular steps (maximising an acquisition function over the entire step) and local steps (acquisition function maximisation restricted to a trust region).

We get the best of both worlds: flexible GP fits, efficient exploration/exploitation trade-off of BO, and powerful local search + convergence guarantees of TR.



TREGO in action



Theoretical results

Key: appropriate **sufficient decrease conditions** and **trust region updates**

$$f(x_{k+1}^{\text{local}}) \leq f(x_k^*) - \rho(\sigma_k)$$

$$\rho(t)/t \to 0 \text{ when } t \downarrow 0 \text{ (for instance, } \rho(t) = t^2)$$

$$\sigma_{k+1} = \frac{\sigma_{k+1}}{\beta} \quad \text{if the iteration is successful}$$

$$\sigma_{k+1} = \sigma_{k+1}\beta \quad \text{otherwise.}$$

If the objective function is bounded from below and Lipschitz continuous near appropriate limit points, then

TREGO always generates a sequence of points that converges to a stationary point.

Under much more strict conditions (f belongs to the appropriate RKHS), TREGO also converges to the global minimum of the problem.

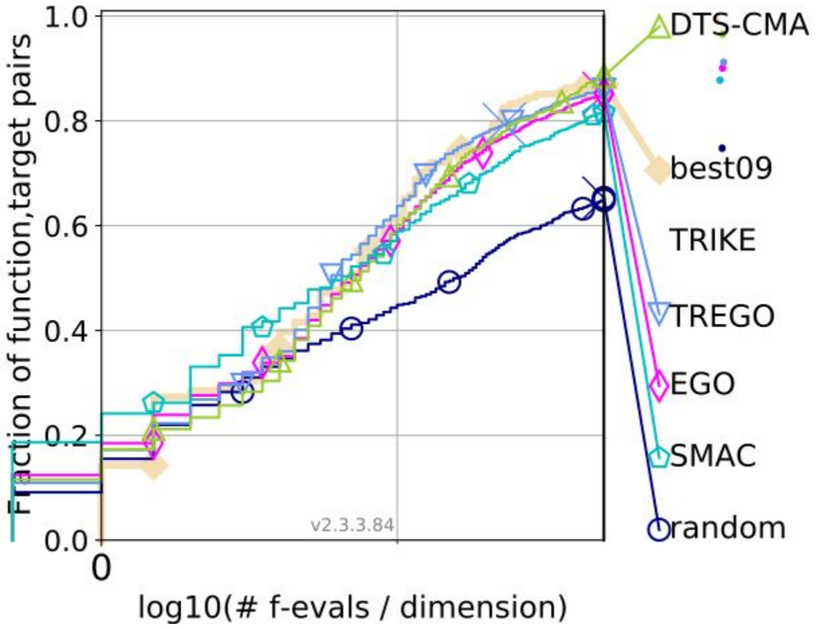
The analysis allows for **many variants**, such as unequal number of global and local steps, using a different model inside the trust region, not using BO inside the trust region, etc.

Secondmind

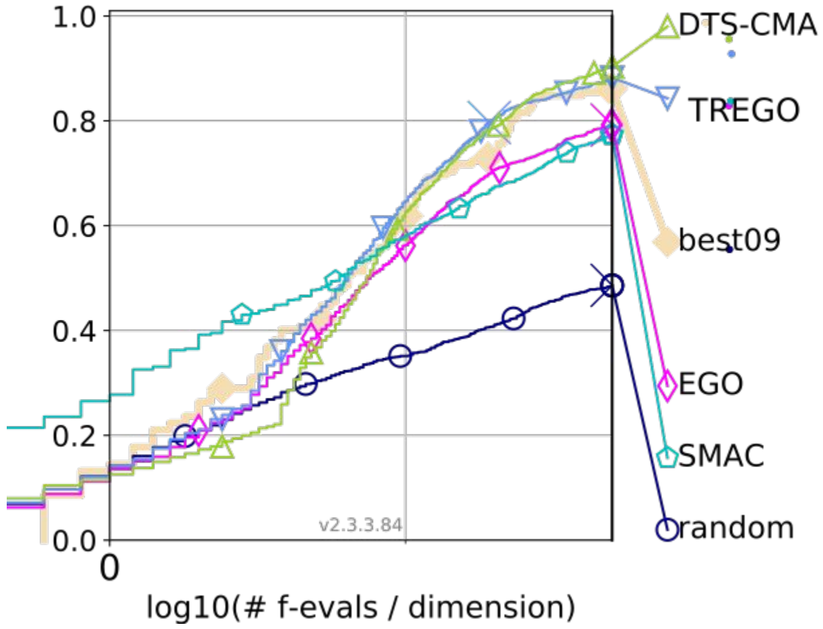
Results on the COCO benchmark

Over all functions, by dimension

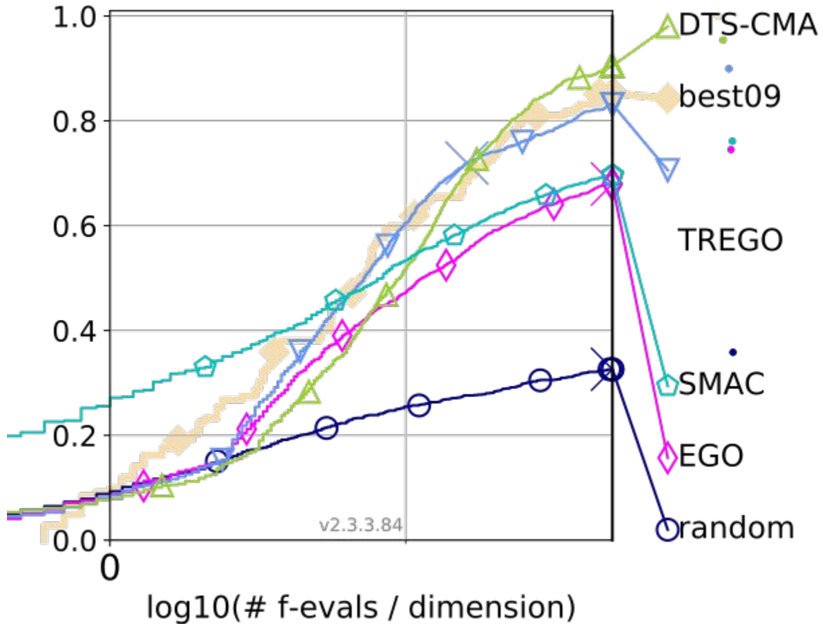
2D



5D



10D



TREGO outperforms EGO in all scenarii - but in particular in d > 3

TREGO closes the gap with DTS-CMA… up to a certain budget

Secondmind

Secondmind

Conclusion

Conclusion

Bayesian optimisation claims state-of-the-art performance on black-box expensive problems

Extensive experiments on the COCO benchmark show that this is partially true

Main drawback identified: loss of performance with dimension

Global perspectives:

- Warping (non-stationary models): we need fast, automatic approaches
- “Cold-start”: reproduce SMAC performance (unfortunately undocumented)
- Better scaling with dimension

TREGO:

- Improved trust regions schemes (ellipsoidal trust regions, optimised updates)
- Noisy problems

Useful links

TREGO is available in R (**DiceOptim**) and python (**trieste**)

https://cran.r-project.org/web/packages/DiceOptim/index.html

https://secondmind-labs.github.io/trieste/

COCO-compatible version of DiceOptim: look for easyEGO, fastEGO.nsteps, TREGO.nsteps

COCO toolbox:

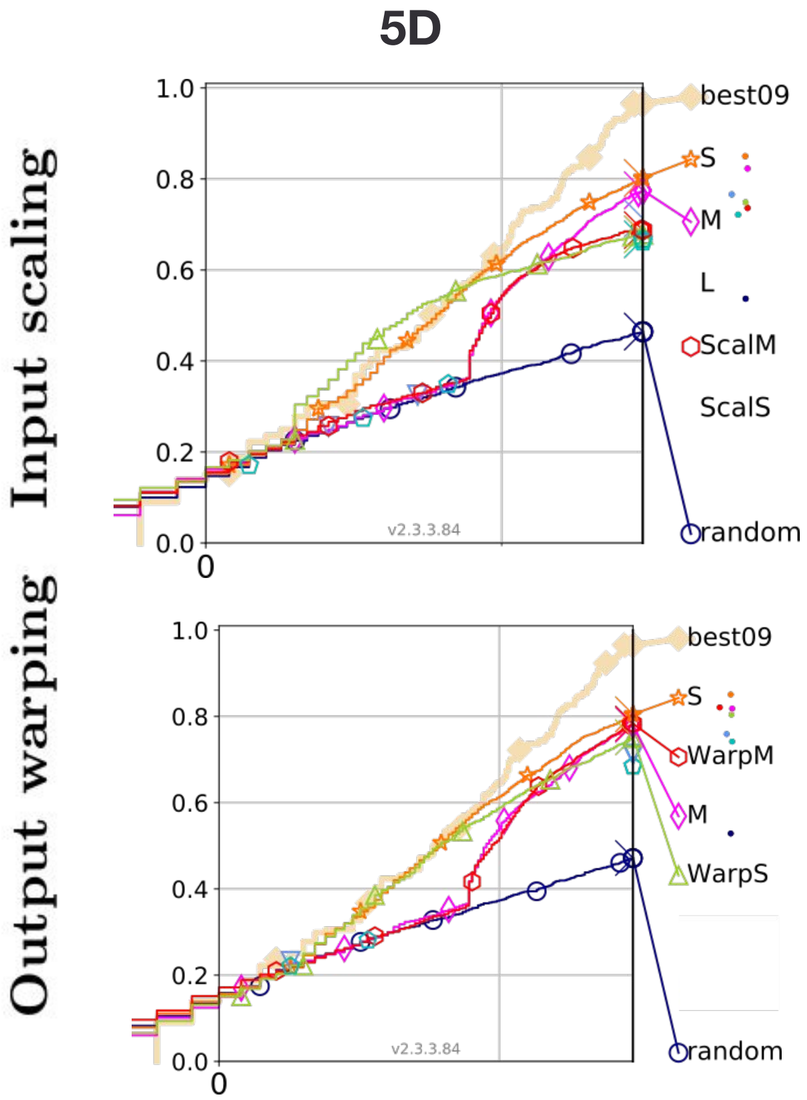
https://coco.gforge.inria.fr/

TREGO preprint:

https://arxiv.org/abs/2101.06808

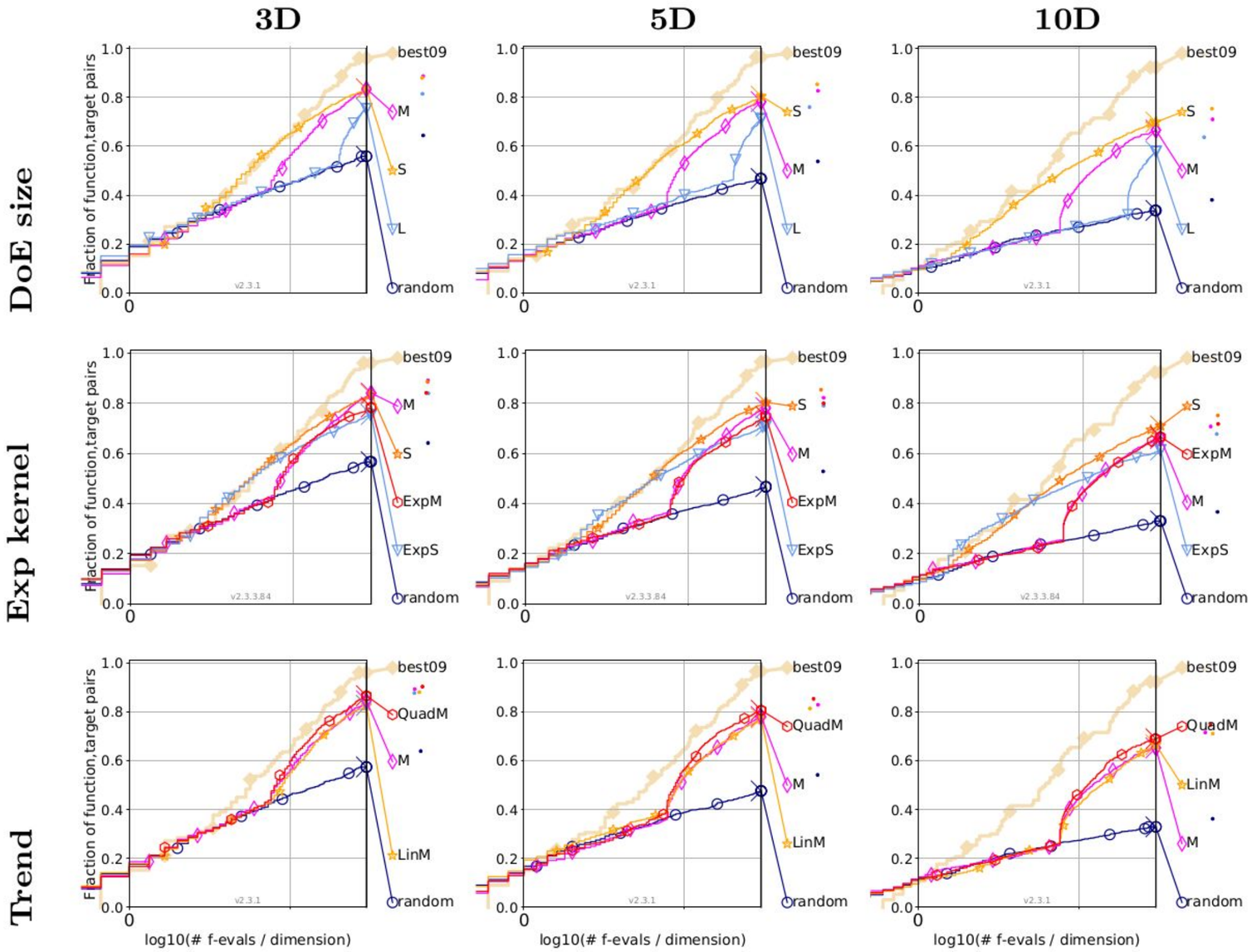

Secondmind

Results (excerpt): ablation study



Both are generally bad, with a couple of exceptions (on functions, not groups).

Results (excerpt): BO pending questions



Small is **always** better

(except on super tricky problems, high dim)

Matern52 is a much better default choice…

but Exp quite good on multimodal functions

(Surprisingly) quadratic trend helps!

Mostly on easy functions: