Benchmarking the CMA-ES with Rank one update on the bbob noiseless test suite.

Presented by:

Raneem Madani, Ikhlas Enaeih, Zeidan Braik

Derivative Free Optimization Master 2 Optimization.

Evolution Strategies:

You evolve iteratively better to reach a desired target.

Motivation:

Some problems make a function difficult to solve:
Dimensionality, ill conditioning, non smooth level sets etc….

Blackbox Scenario:

You have inputs and outputs.

- minimization in this scenario requires finding:

Both the input that yields the minimum output.

General Scheme for minimizing $f: \mathbb{R}^n \to \mathbb{R}$ in black box search:

Initialize Distribution Parameters θ & Set Population Size $\lambda \in \mathbb{N}$

- Sample Distribution $P(x/\theta) \to x_1 \dots x_\lambda \in \mathbb{R}^n$
- Evaluate these points on f
- Update Parameters.

Next: CMA-ES Rank one……..

CMA – ES with Rank One Update

Initialize Distribution Parameters θ & Set Population Size $\lambda \in \mathbb{N}$

CMA – ES with Rank One Update

Initialize Distribution Parameters θ & Set Population Size $\lambda \in \mathbb{N}$

- λ Population Size (offspring) how many sample you will draw.
- μ Candidate Solution (Parents) the best μ points out of λ.

The Parameters:

- m The initial mean vector $\in \mathbb{R}^n$ represent favourite solution (Where to sample?)
- σ Step size $\in \mathbb{R}_+$ Control the Step Length (How far you will go?)
- $\mathbb{C}$ Covariance Matrix $\in \mathbb{R}^{n \times n}$ the shape of the distribution ellipsoid (which region will you sample in?)

WHY these Parameters?

General Scheme for minimizing $f: \mathbb{R}^n \to \mathbb{R}$ in black box search:

Initialize Distribution Parameters θ & Set Population Size $\lambda \in \mathbb{N}$

- Sample Distribution $P(x/\theta) \to x_1 \dots x_\lambda \in \mathbb{R}^n$
- Evaluate these points on f

Sample Distribution:

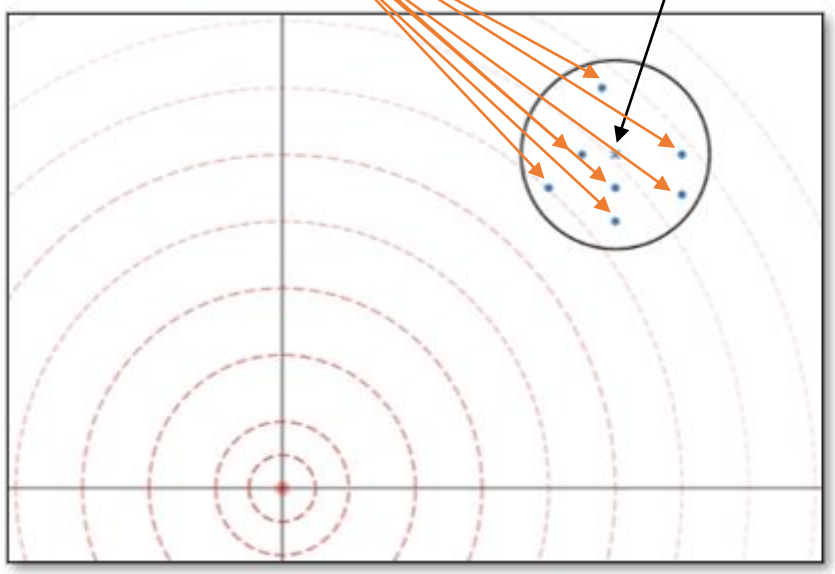
The CMA-ES use normal distribution to sample the λ points

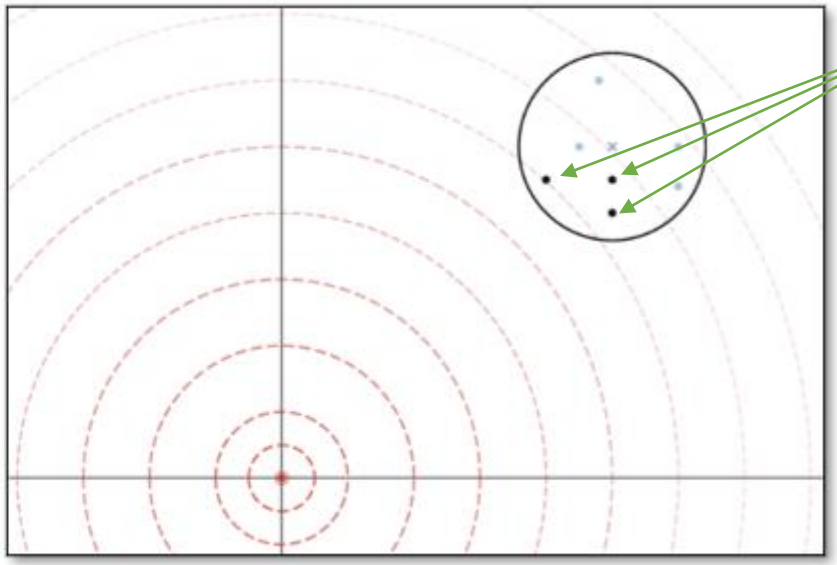
$$y_i \sim N(0, C)$$
$$x_i = m + \sigma y_i \sim N(\mathrm{m}, \sigma^2 C)$$

m

$$f(x_{1:\lambda}) < f(x_{2:\lambda}) < \ldots < f(x_{\lambda:\lambda})$$
$$\downarrow \qquad \downarrow \qquad \downarrow$$
$$y_{1:\lambda} < y_{2:\lambda} < \ldots < y_{\lambda:\lambda}$$

$y_{1:\lambda}, \ldots, y_{\mu:\lambda}$
The Ranked Directions





General Scheme for minimizing $f: \mathbb{R}^n \to \mathbb{R}$ in black box search:

Initialize Distribution Parameters θ & Set Population Size $\lambda \in \mathbb{N}$

- Sample Distribution $P(x/\theta) \to x_1 \dots x_\lambda \in \mathbb{R}^n$
- Evaluate these points on f
- Update Parameters.
  - Mean
  - Step Size
  - Covariance

General Scheme for minimizing $f: \mathbb{R}^n \to \mathbb{R}$ in black box search:

Initialize Distribution Parameters θ & Set Population Size $\lambda \in \mathbb{N}$

- Sample Distribution $P(x/\theta) \to x_1 \dots x_\lambda \in \mathbb{R}^n$
- Evaluate these points on f
- Update Parameters.
  - Mean

Mean Update

The Mean update is done using weighted intermediate recombination.

$$m_{t+1} = \sum_{i=1}^{\mu} w_i x_{i:\lambda}$$

$$= m_t + \sigma \boxed{\sum_{i=1}^{\mu} w_i y_{i:\lambda}} \rightarrow =: y_w$$

General Scheme for minimizing $f: \mathbb{R}^n \rightarrow \mathbb{R}$ in black box search:

Initialize Distribution Parameters θ & Set Population Size $\lambda \in \mathbb{N}$

- Sample Distribution $P(x/\theta) \rightarrow x_1 \dots x_\lambda \in \mathbb{R}^n$
- Evaluate these points on f
- Update Parameters.
  - Mean
  - Step Size

Step Size Update:

Several methods in general are used:

- 1/5-th success rule
- σ – self – adaptation
- Path length control (CSA)

In CMA-ES we will use CSA to update the Step Size.

Evolution path is used p_σ then we update σ

General Scheme for minimizing $f: \mathbb{R}^n \to \mathbb{R}$ in black box search:

Initialize Distribution Parameters θ & Set Population Size $\lambda \in \mathbb{N}$

- Sample Distribution $P(x/\theta) \to x_1 \dots x_\lambda \in \mathbb{R}^n$
- Evaluate these points on f
- Update Parameters.
  - Mean
  - Step Size
  - Covariance

Now The Covariance Matrix Update.

Motivation:

Step size control is not enough (why?)

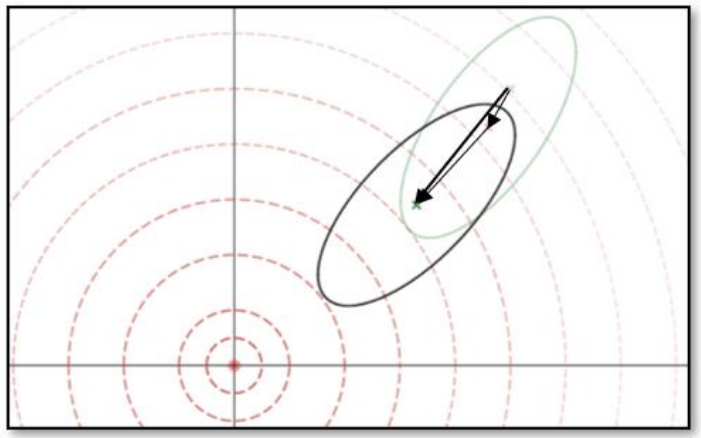
Because convergence speed of CSA-ES becomes lower as function become ill conditioned

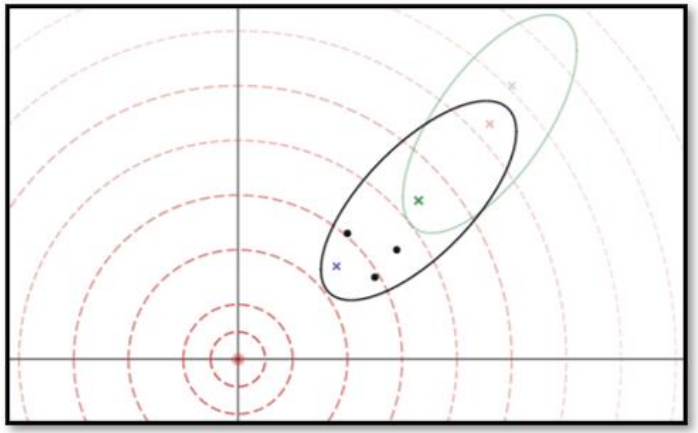
Cumulation….

Same approach used in step size update is followed:

We find the evolution path where history information is accumulated.

- We initialize $p_c = 0$
- $p_{c\,t+1} = (1 - c_c)p_{c\,t} + \sqrt{1 - (1 - c_c)}\,\sqrt{\mu_w}\,y_w$



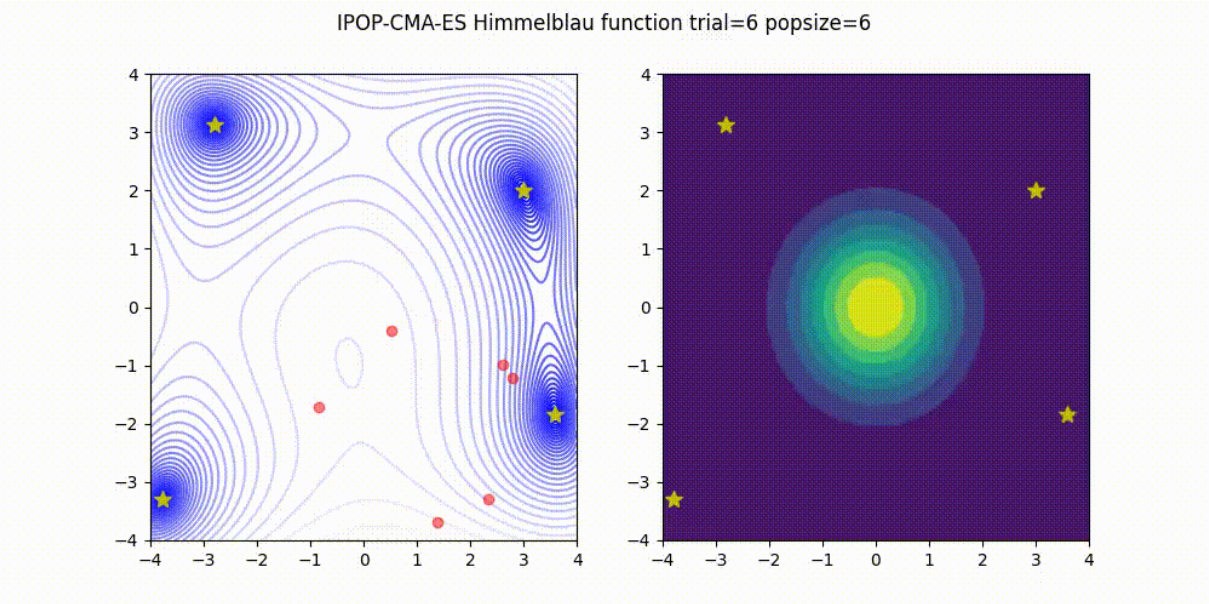


Finally The Covariance Update:

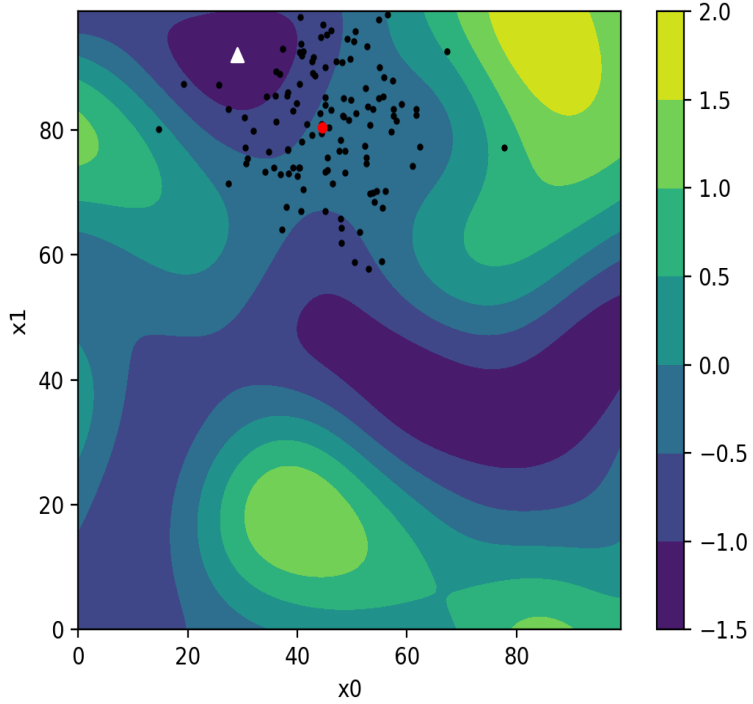
$$C_{t+1} = (1 - c_{cov})C_t + C_{cov}\, p_c p_c^T$$

This is a Matrix of Rank one

Where $\mu_w = \frac{1}{\sum w_i^2}$, $c_{cov} \ll c_c \ll 1$



Source1: https://medium.com/optuna/introduction-to-cma-es-sampler-ee68194c8f88



Source2: https://thurinj.github.io/CMA-ES.html

General Scheme for minimizing $f: \mathbb{R}^n \to \mathbb{R}$ in black box search:

Initialize Distribution Parameters θ & Set Population Size $\lambda \in \mathbb{N}$

- Sample Distribution $P(x/\theta) \to x_1 \dots x_\lambda \in \mathbb{R}^n$
- Evaluate these points on f
- Update Parameters.
  - Mean
  - Step Size
  - Covariance

Benchmark?

How to track the performance of CMA-ES rank one?

In Order to study the performance of CMA-ES with rank one update we used COCO platform and benchmarked it on BBOB Noiseless Suite.

Tests Conducted:

- Single Algorithm Benchmark with:
  - λ default = $4 + \lfloor 3 \log n \rfloor$ / μ default = $\frac{\lambda}{2}$ / restarts = 30 / budget = 10^5 / no increase popsize
  - λ = 300 / μ default = $\frac{\lambda}{2}$ / restarts = 30 / budget = 10^4 / increase popsize of factor 2
- Multiple Algorithms Benchmark:
  - Low population (λ =default) CMA-ES rank one with (Random search $[-5,5]^n$ / BFGS / Default CMA-ES)
  - Large population ($\lambda = 300$)CMA-ES rank one with (Random search $[-5,5]^n$ / BFGS / Default CMA-ES)

BBOB Test Suite:

24 Noiseless Functions Separated to:

1 Separable Functions

f1	Sphere Function
f2	Separable Ellipsoidal Function
f3	Rastrigin Function
f4	Büche-Rastrigin Function
f5	Linear Slope

2 Functions with low or moderate conditioning

f6	Attractive Sector Function
f7	Step Ellipsoidal Function
f8	Rosenbrock Function, original
f9	Rosenbrock Function, rotated

3 Functions with high conditioning and unimodal

f10	Ellipsoidal Function
f11	Discus Function
f12	Bent Cigar Function
f13	Sharp Ridge Function
f14	Different Powers Function

4 Multi-modal functions with adequate global structure

f15	Rastrigin Function
f16	Weierstrass Function
f17	Schaffer's F7 Function
f18	Schaffer's F7 Function, moderately ill-conditioned
f19	Composite Griewank-Rosenbrock Function F8F2

5 Multi-modal functions with weak global structure

f20	Schwefel Function
f21	Gallagher's Gaussian 101-me Peaks Function
f22	Gallagher's Gaussian 21-hi Peaks Function
f23	Katsuura Function
f24	Lunacek bi-Rastrigin Function

(Instance, Target) Pairs

The ECDFs Plots resulting from COCO were used to conclude our results

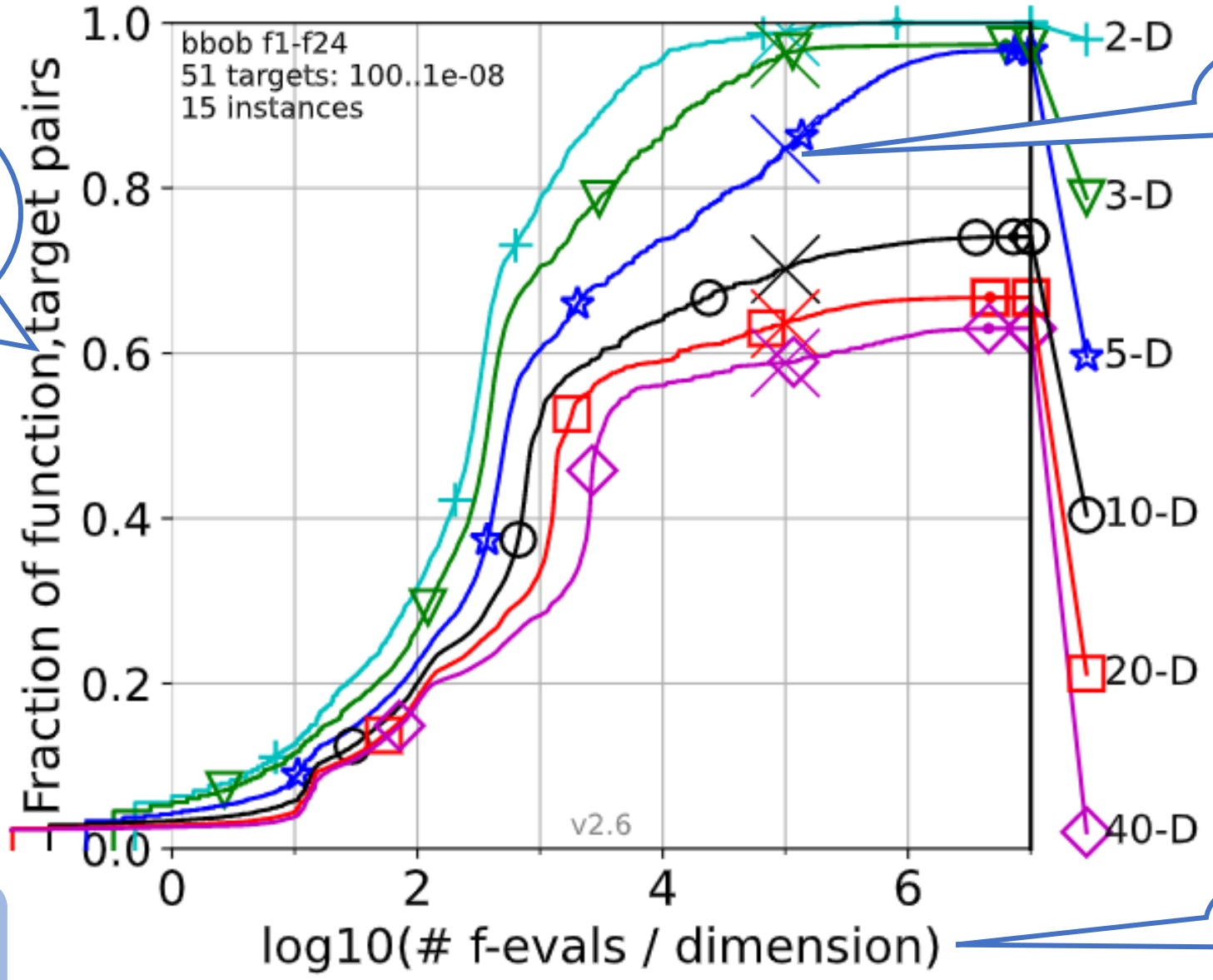
- Runtime ECDFs Over all targets
- Runtime ECDFs per function
- Runtime ECDFs per group

We have 24 functions, each function has 15 Instances (example) and 51 target ranging from $10^{2} - 10^{-8}$

Next, Results…….

With Low Population Size

Runtime distributions (ECDFs) over all targets



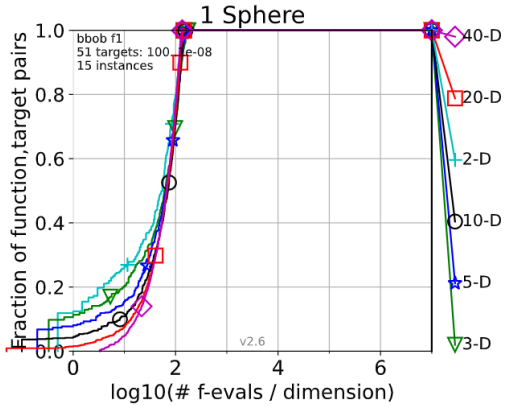
Percentage of (Instance, Target) pairs that were met (solved)

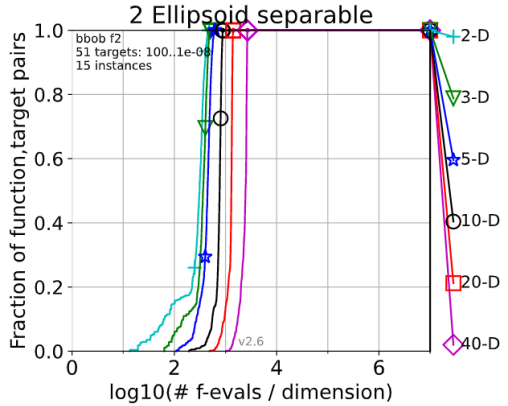
Where the algorithm in dimension D stop

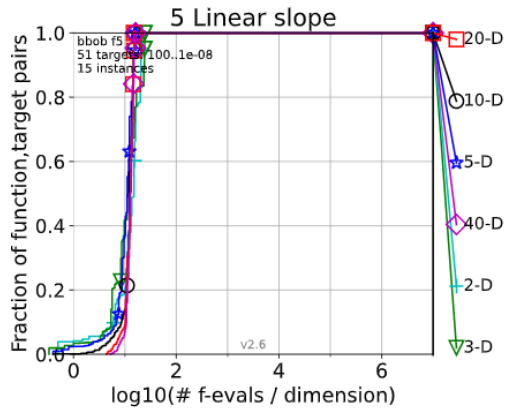
#F-eval=Budget Multiplier*D

Budget Multiplier

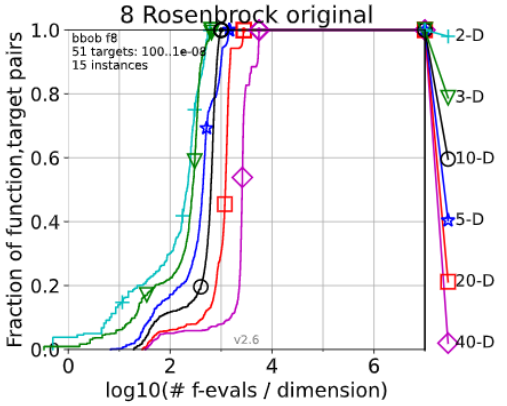
The Functions (f1, f2, f5, f6, f8, f9, f10, f11, f12, f21) have been solved!

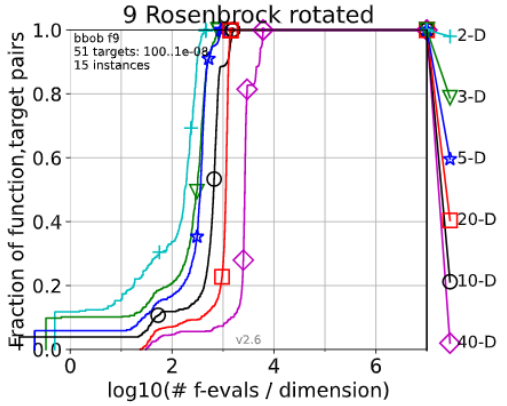


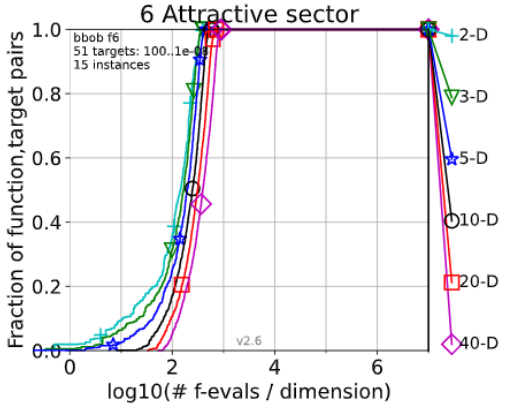




Separable Functions

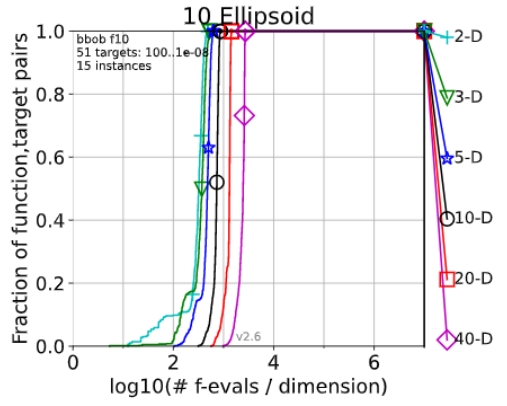


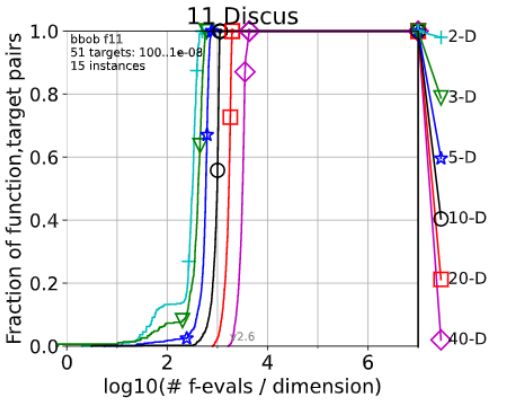


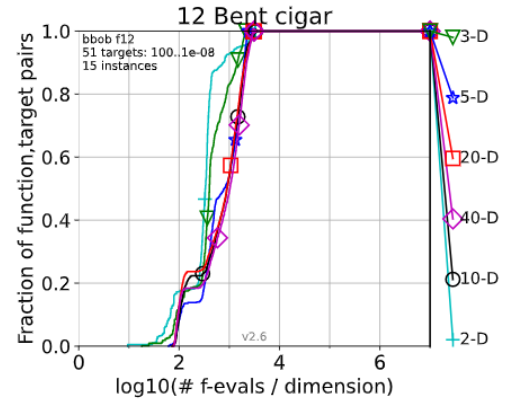


Functions with Low or Moderate Conditioning

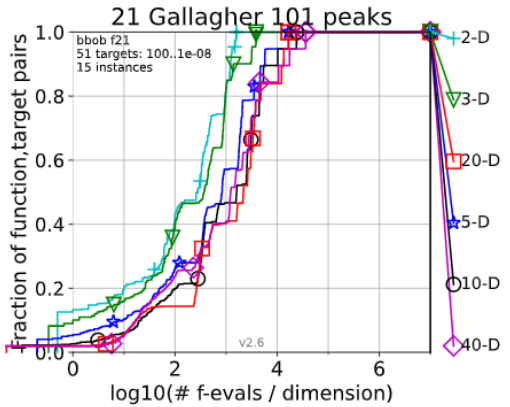
Multi-Modal Functions with Weak Global Structure



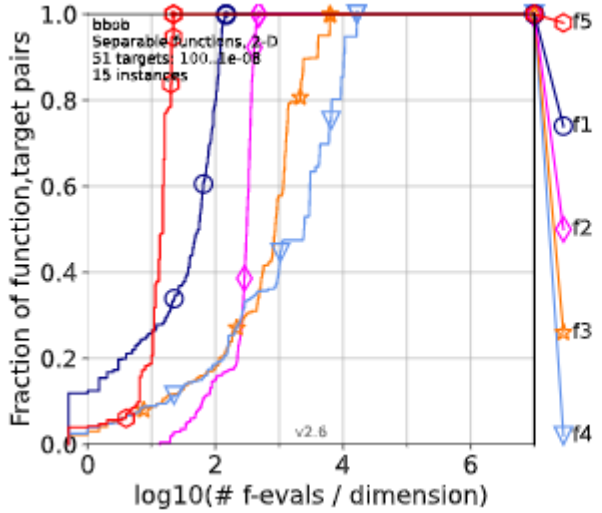


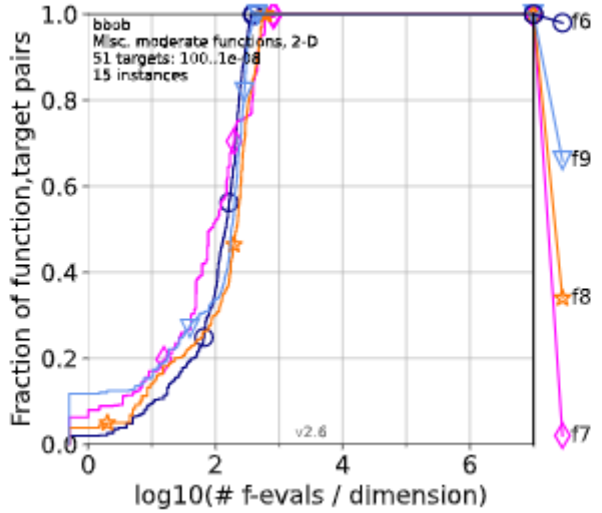


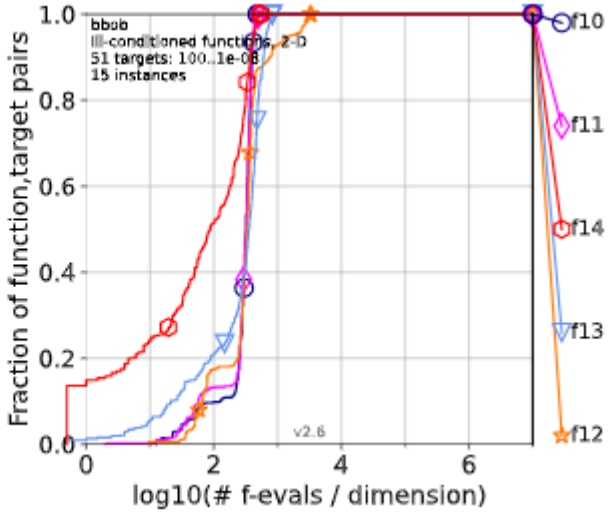
Functions with High Conditioning and Uni-modal

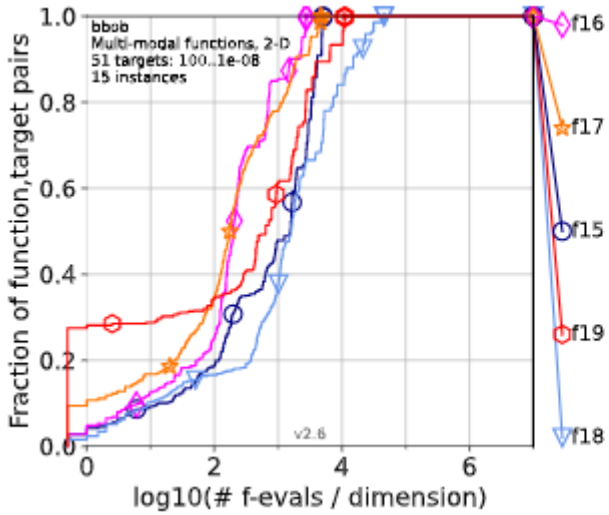


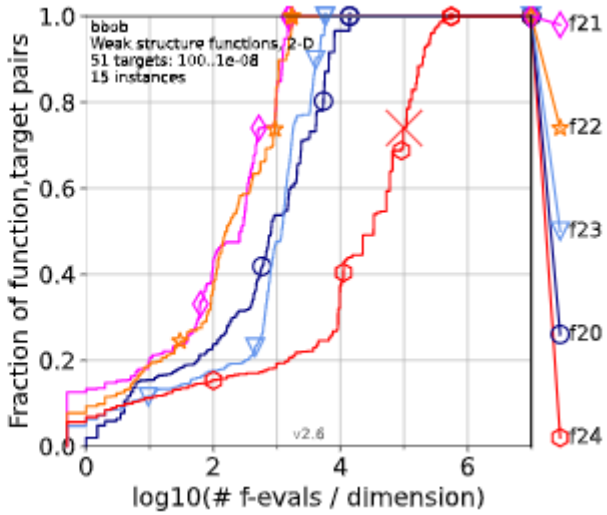
In 2D all functions were solved except f24





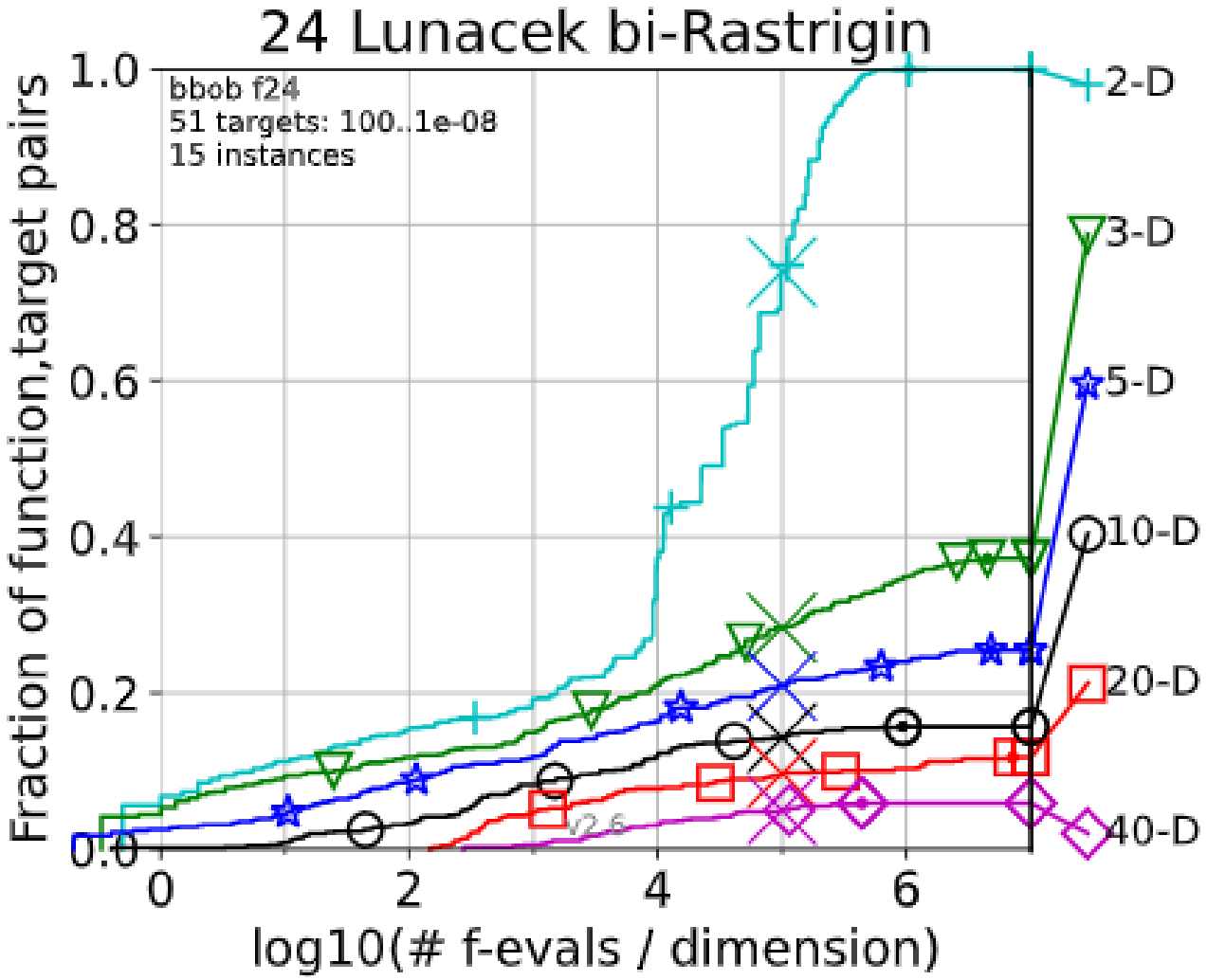




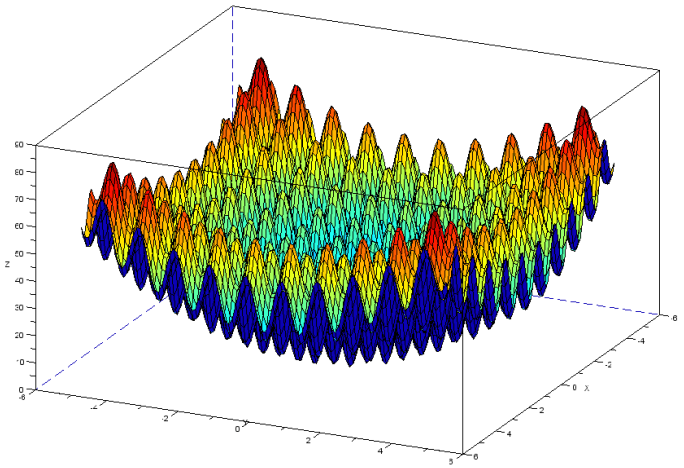


In Higher Dimension Curies of Dimensionality!

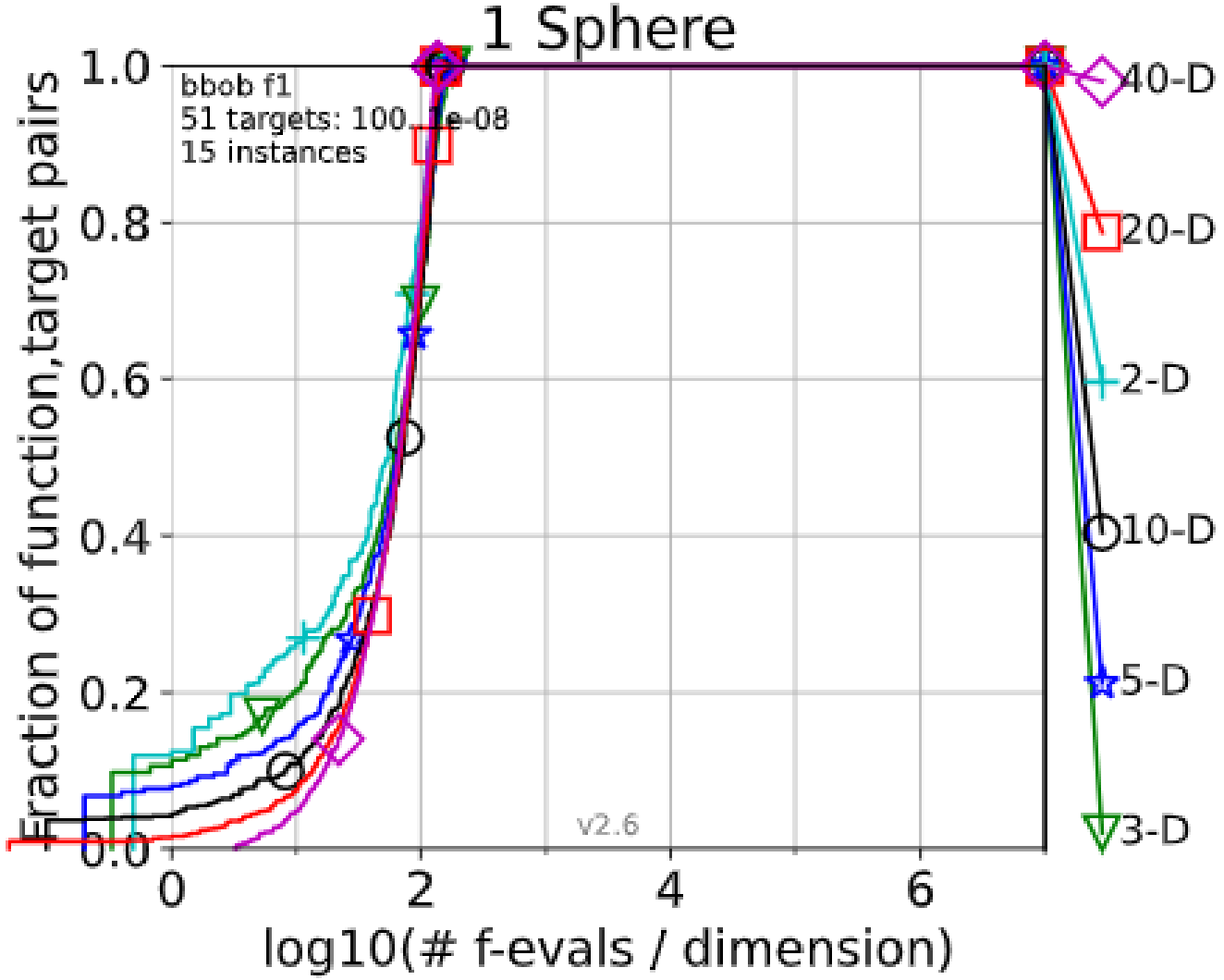
F24 Lunacek bi-Rastrigin unsolved in all for all dimensions



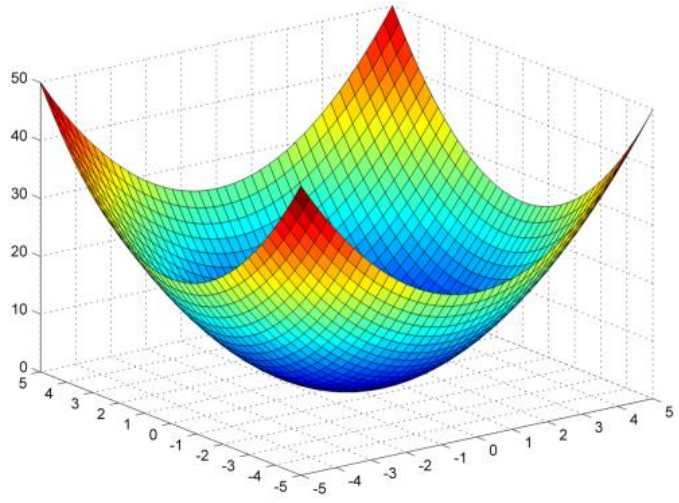
- Multi-Modal Function
- Need high population size



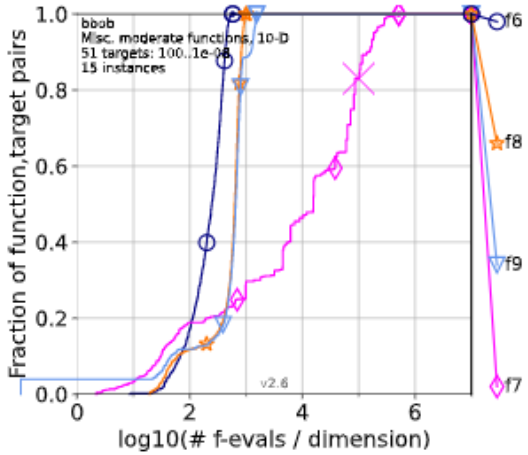
F1 sphere function was completely solved in all dimensions in budget $=10^2$

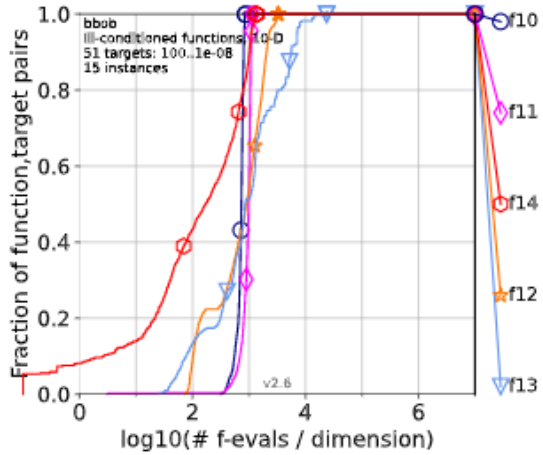


- Separable Function
- Uni-Modal Function

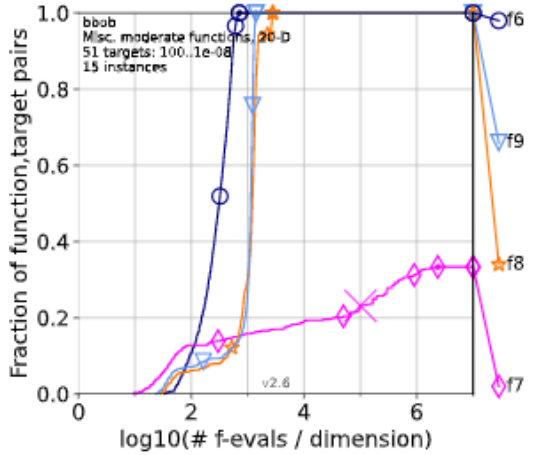


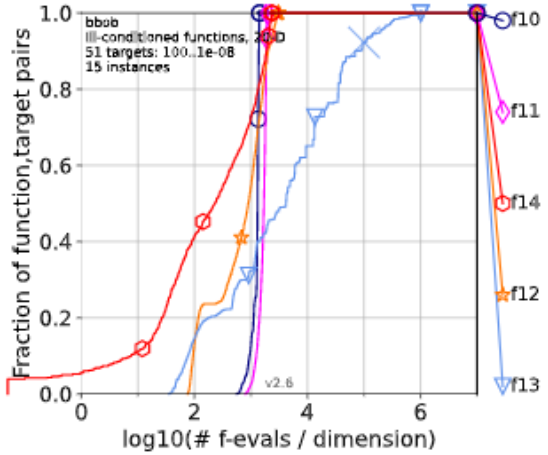
10D



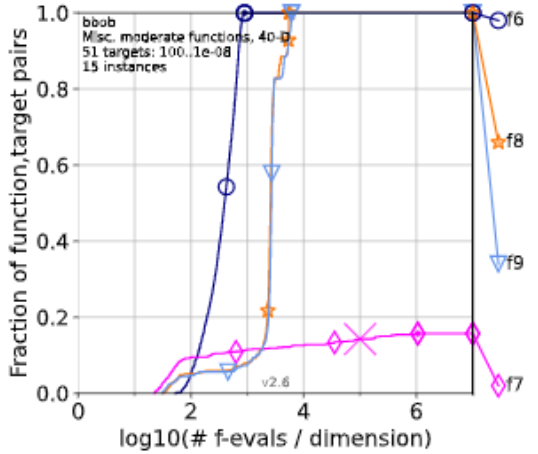


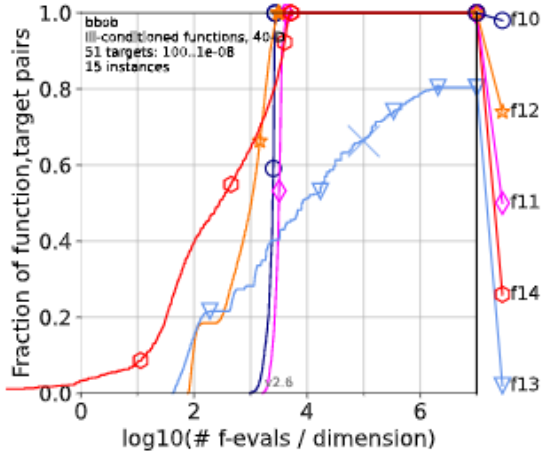
20D





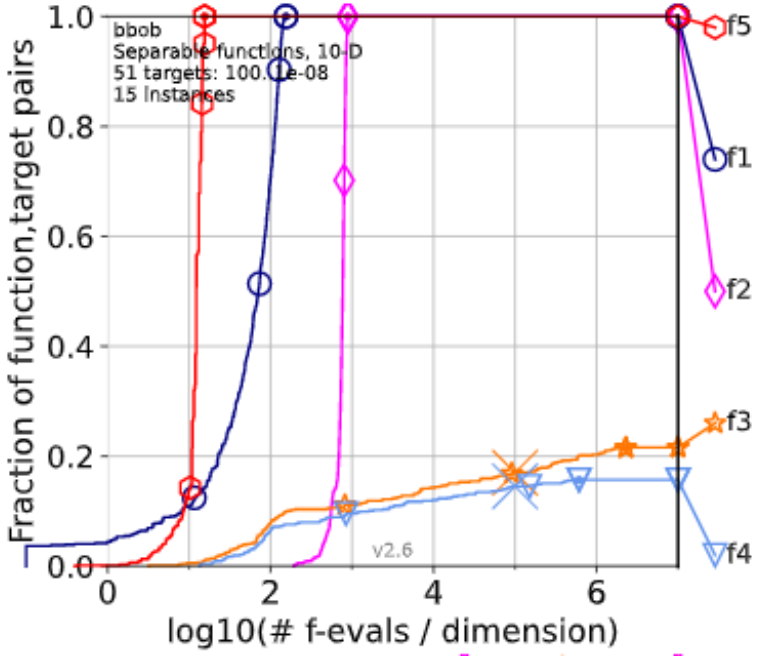
40D



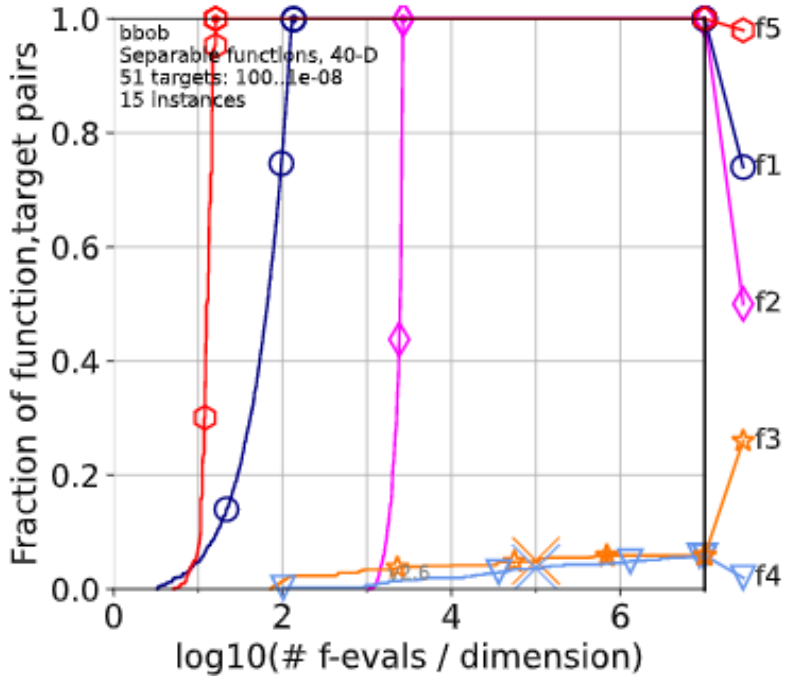


Class two and three were all solved in all dimensions except f7 in 10,20 and 40D with very bad performance in 20,40D and also f13 in 20,40D it needed more budget.

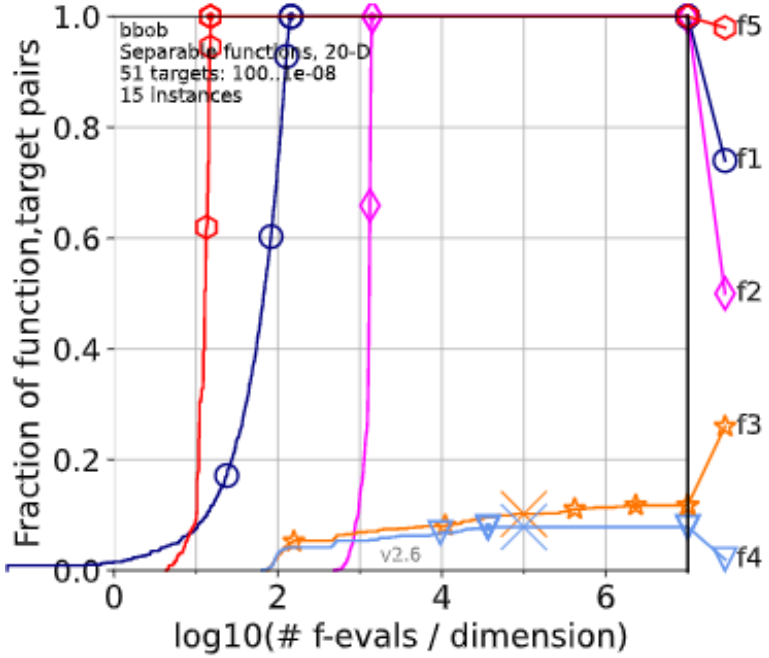
10D



20D

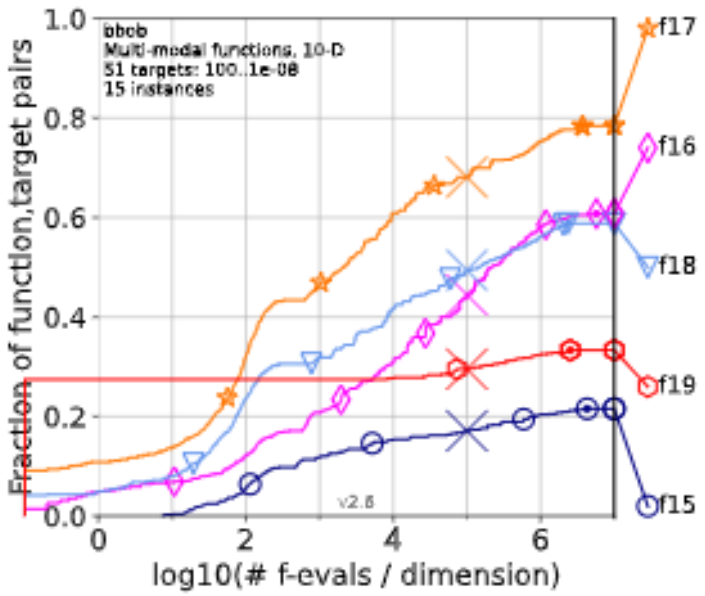


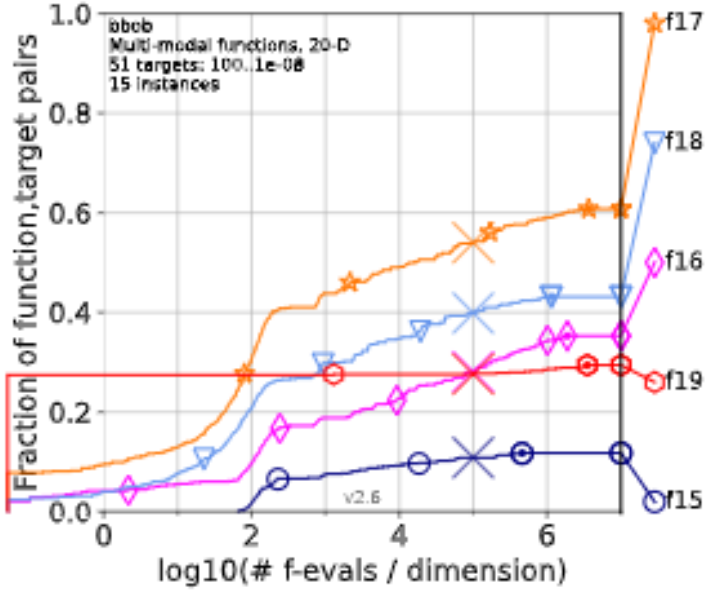
40D

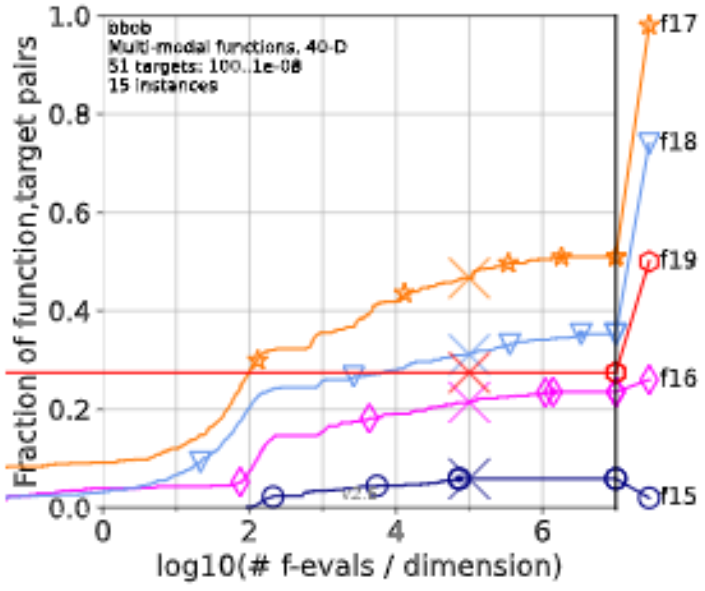


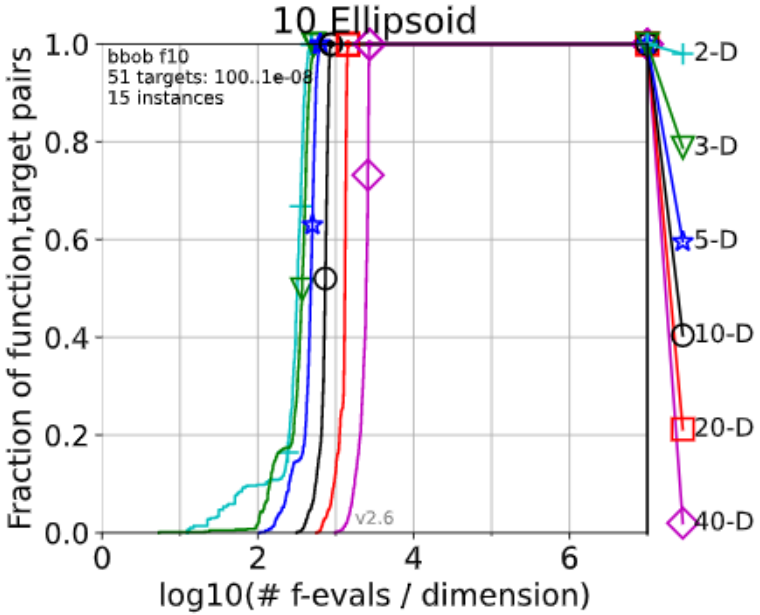
Separable functions did a very high performance except for: F3,f4 not even 20% of them was solved in 10,20,40D

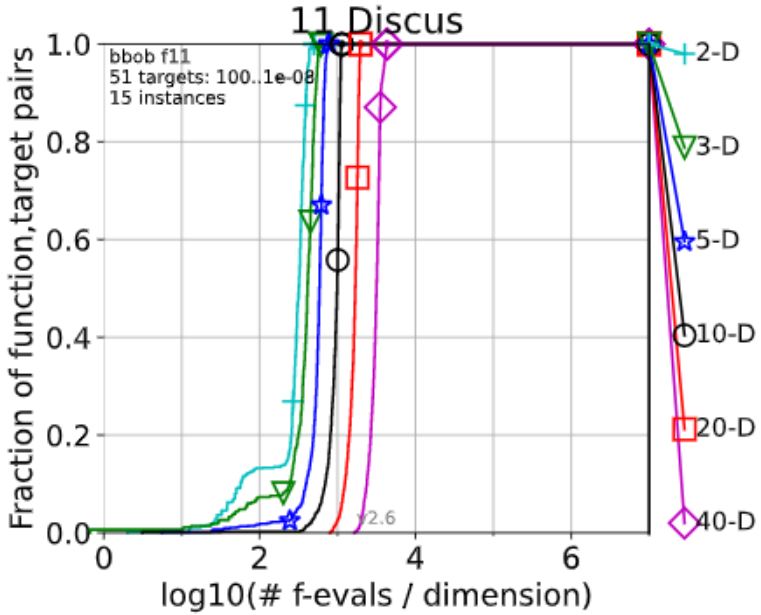
Multimodal functions in dimension $\geq$ 10D
non of them was solved only (20 - 60)%

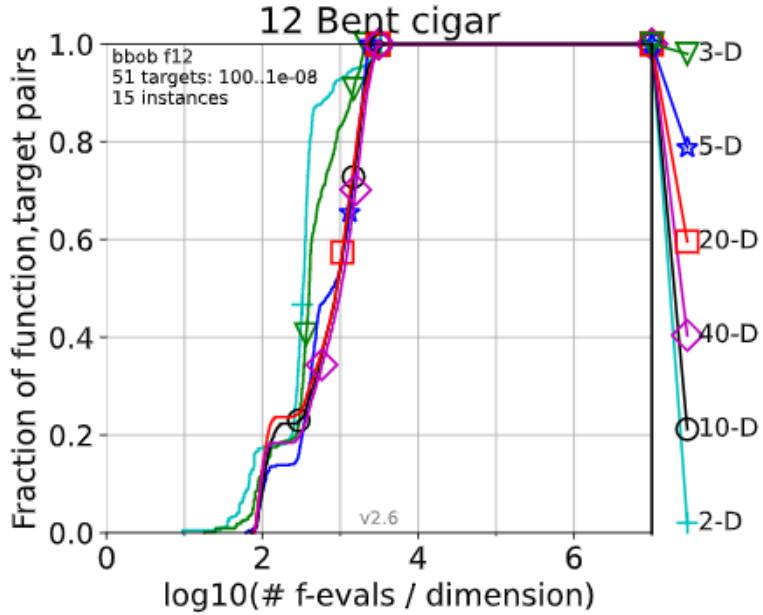


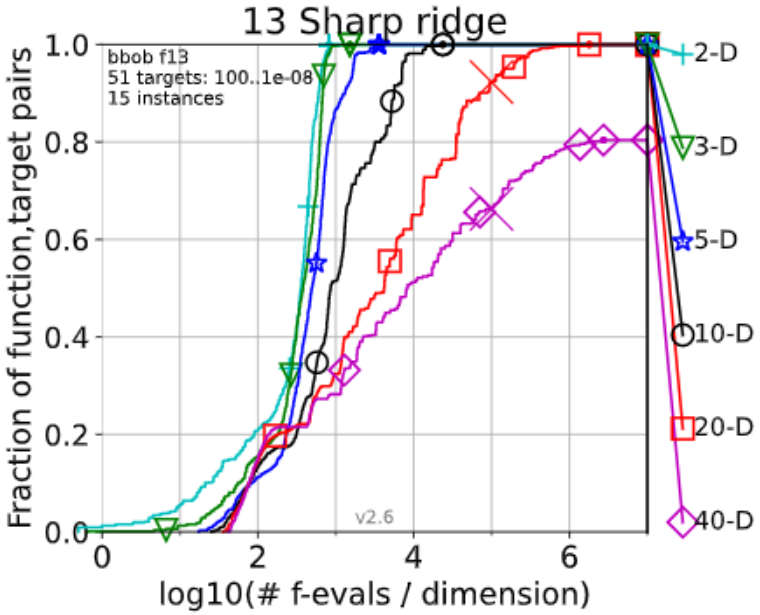


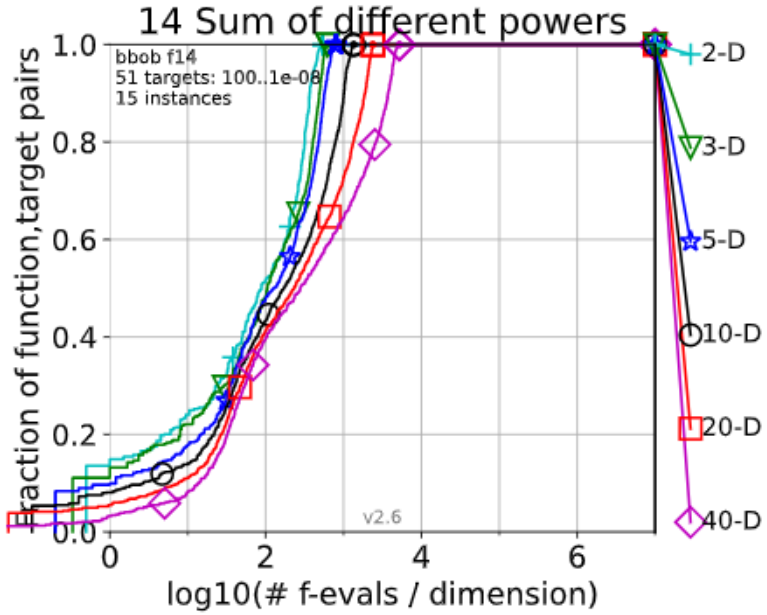










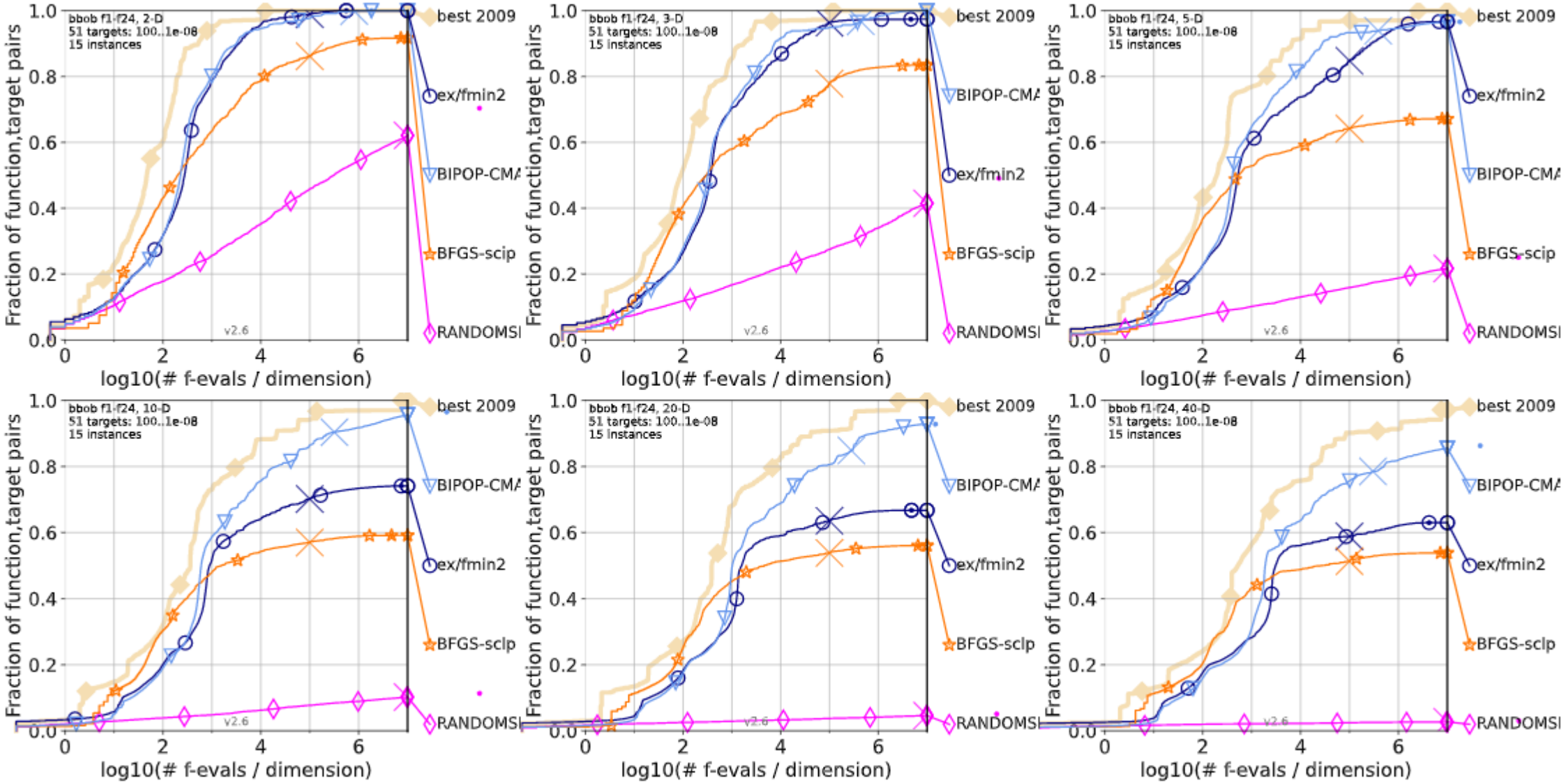


Uni-modal Functions on low dimensions were completely solved except f13 because it is not smooth

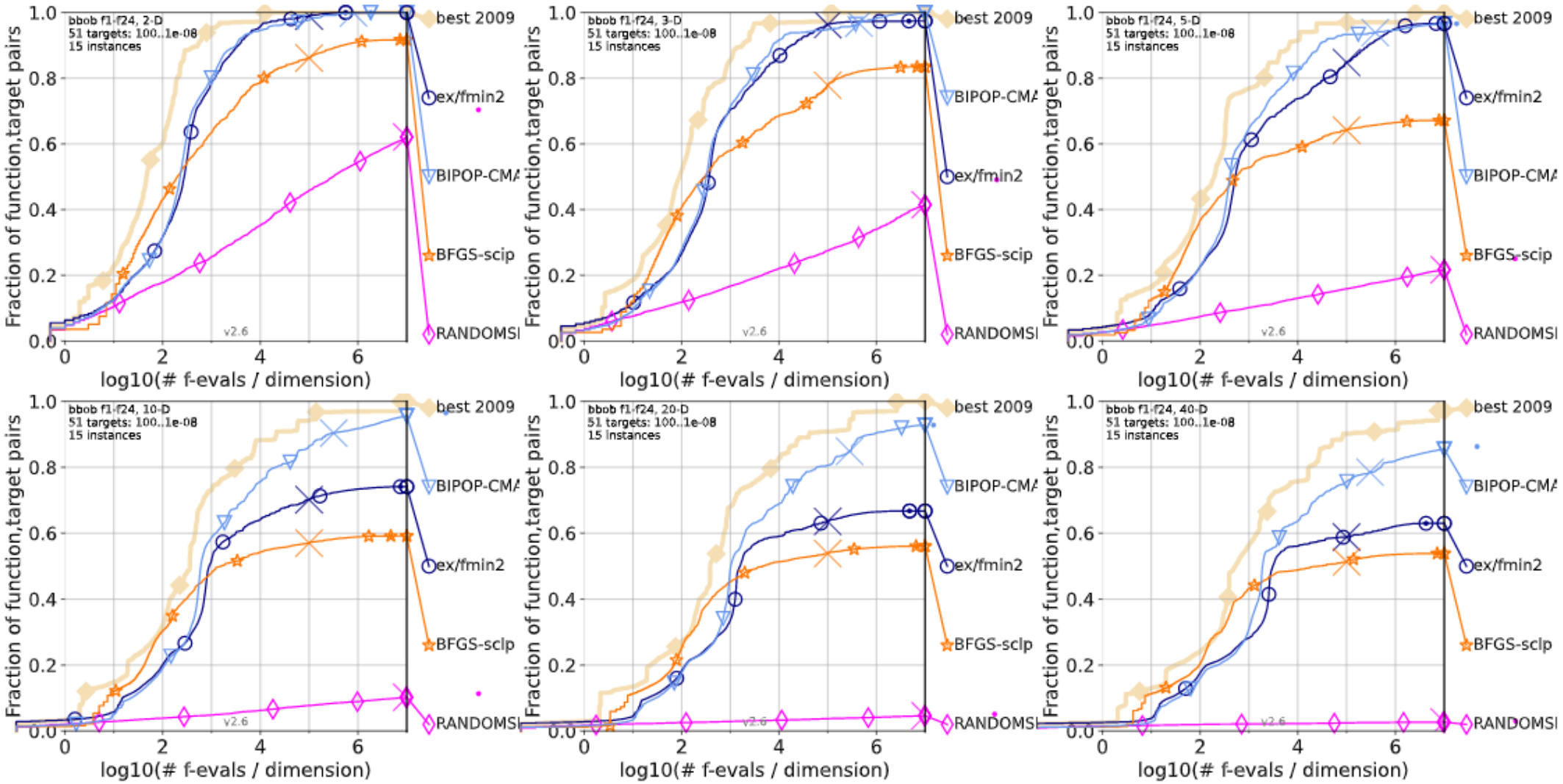
- RANDOMSEARCH-5
- BFGS SciPy 2019
- BIPOP CMA ES Hansen noiseless
- best 2009: which is an algorithm that takes the best parameters among a 31 Algorithms

Now comparing the algorithm with the other algorithms

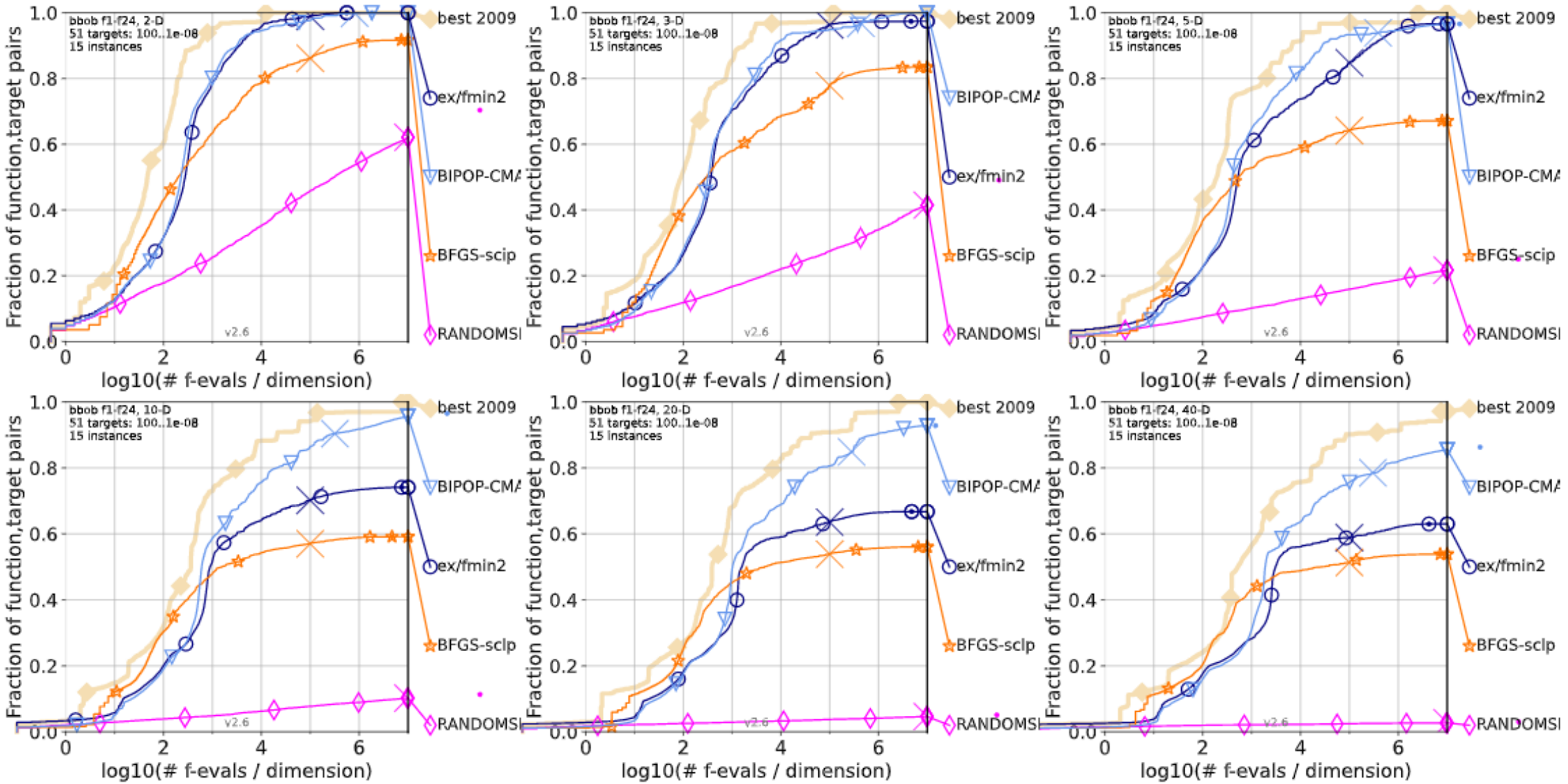
Performance of rank one was very close to performance of BIPOP-CMA-ES for all functions over all targets up to 5D



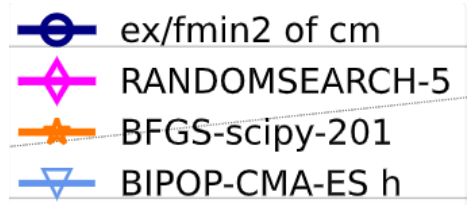
Up to budget 10^3 in all cases default CMA and rank one had same performance in general.

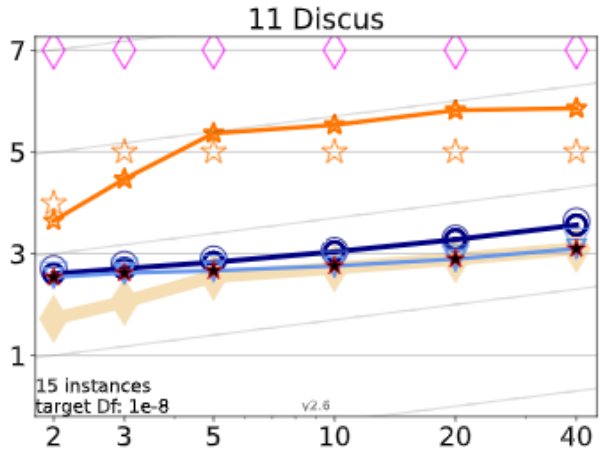


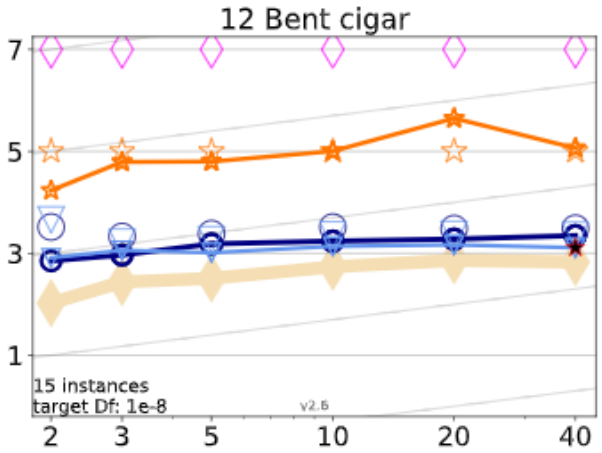
The BIBPOP CMA and CMA-Rank-One For dimensions larger that 5D the performance of the two algorithms was close in the beginning up to budget 10^3 and then BIPOP CMA enhanced way better.

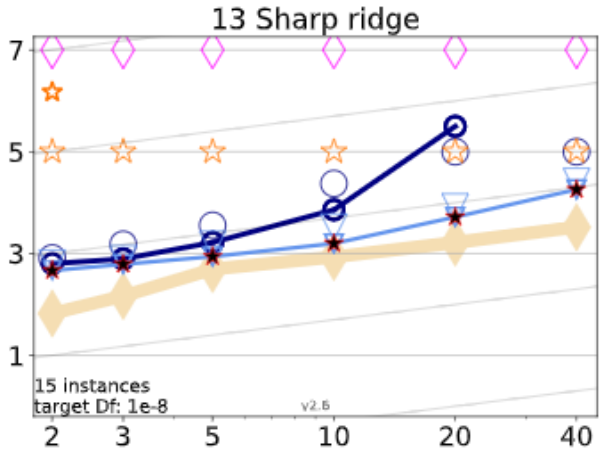


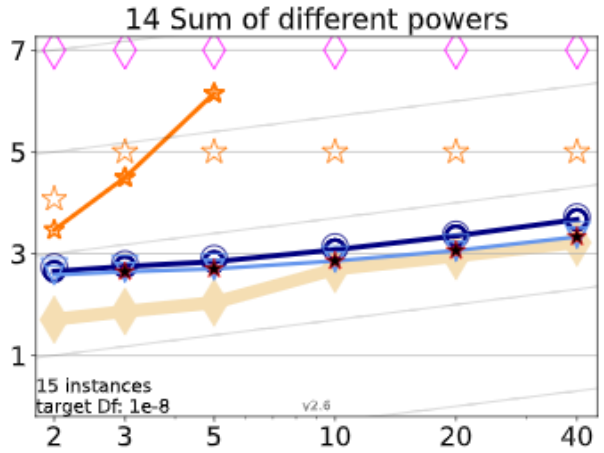
Generally speaking rank one update did better than BFGS-SciPy and Random-search in the most of the Functions.

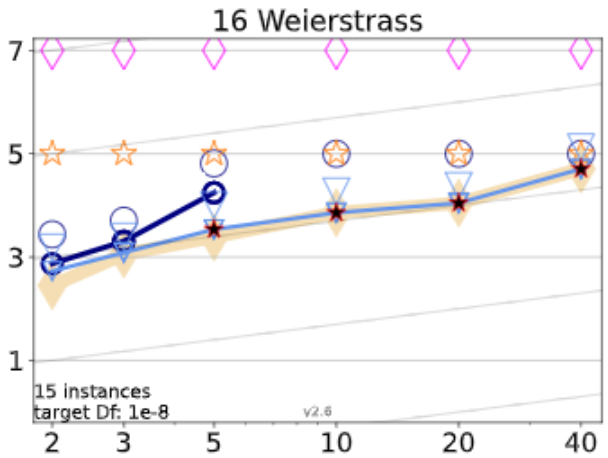


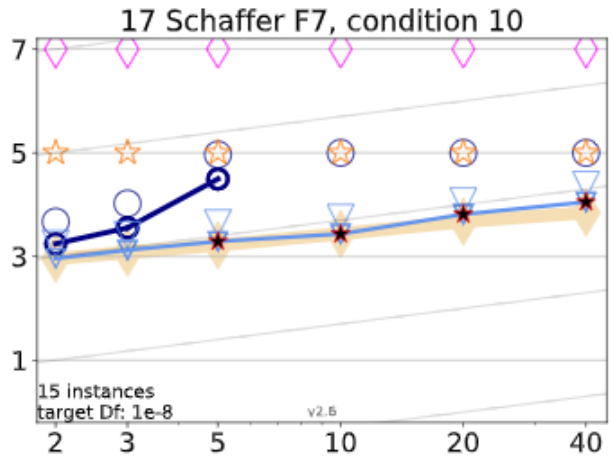


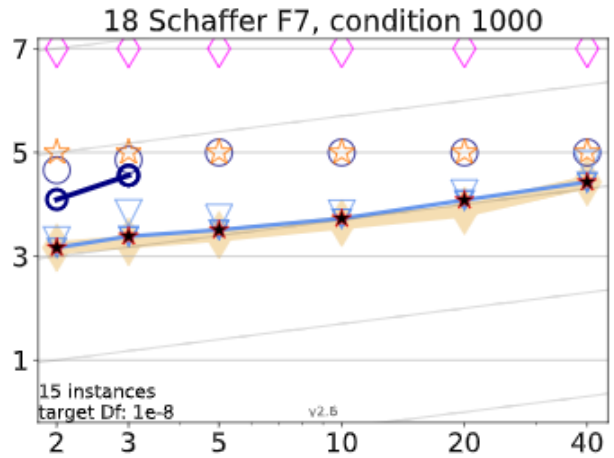


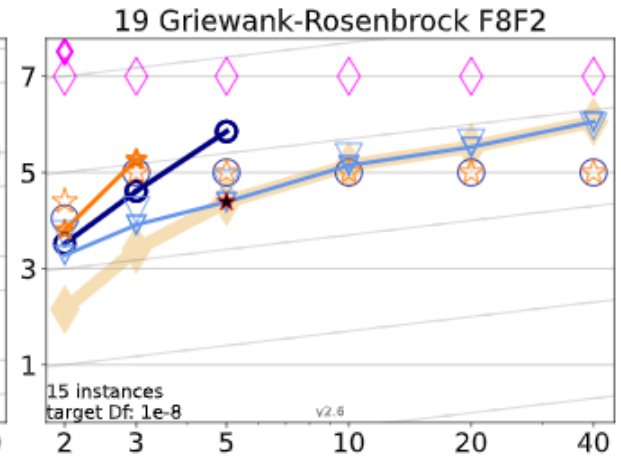




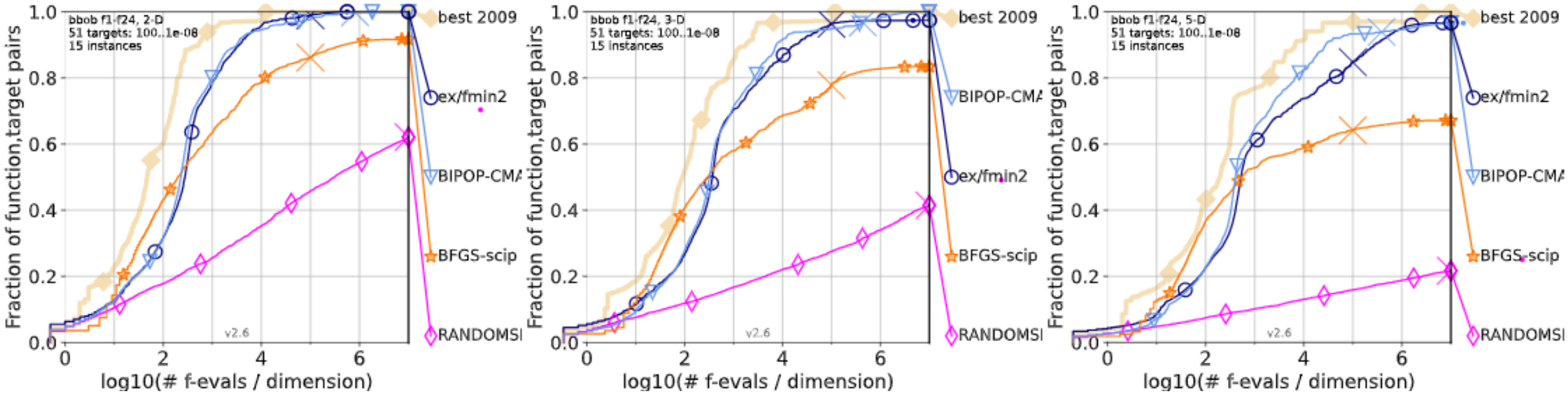




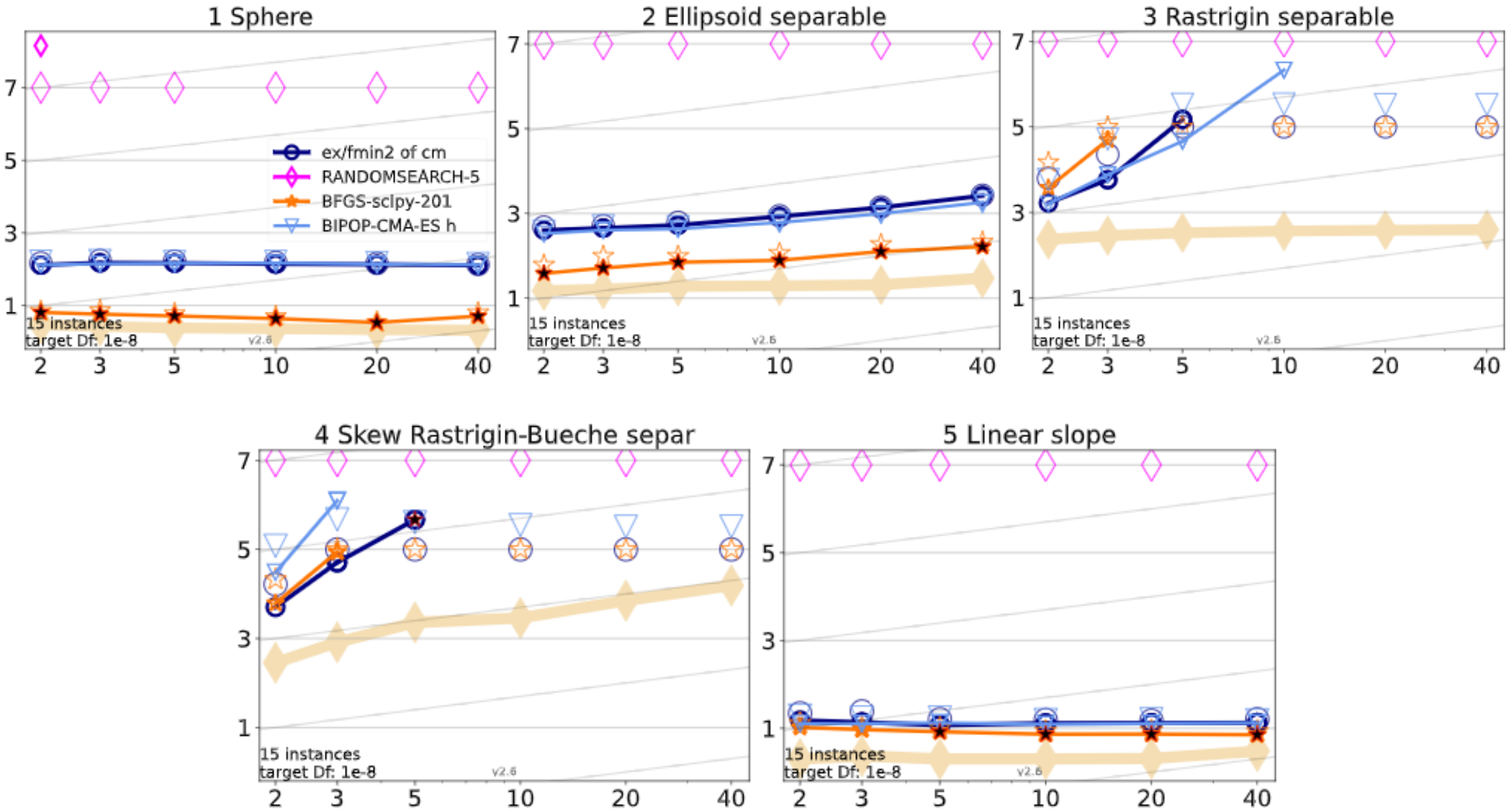




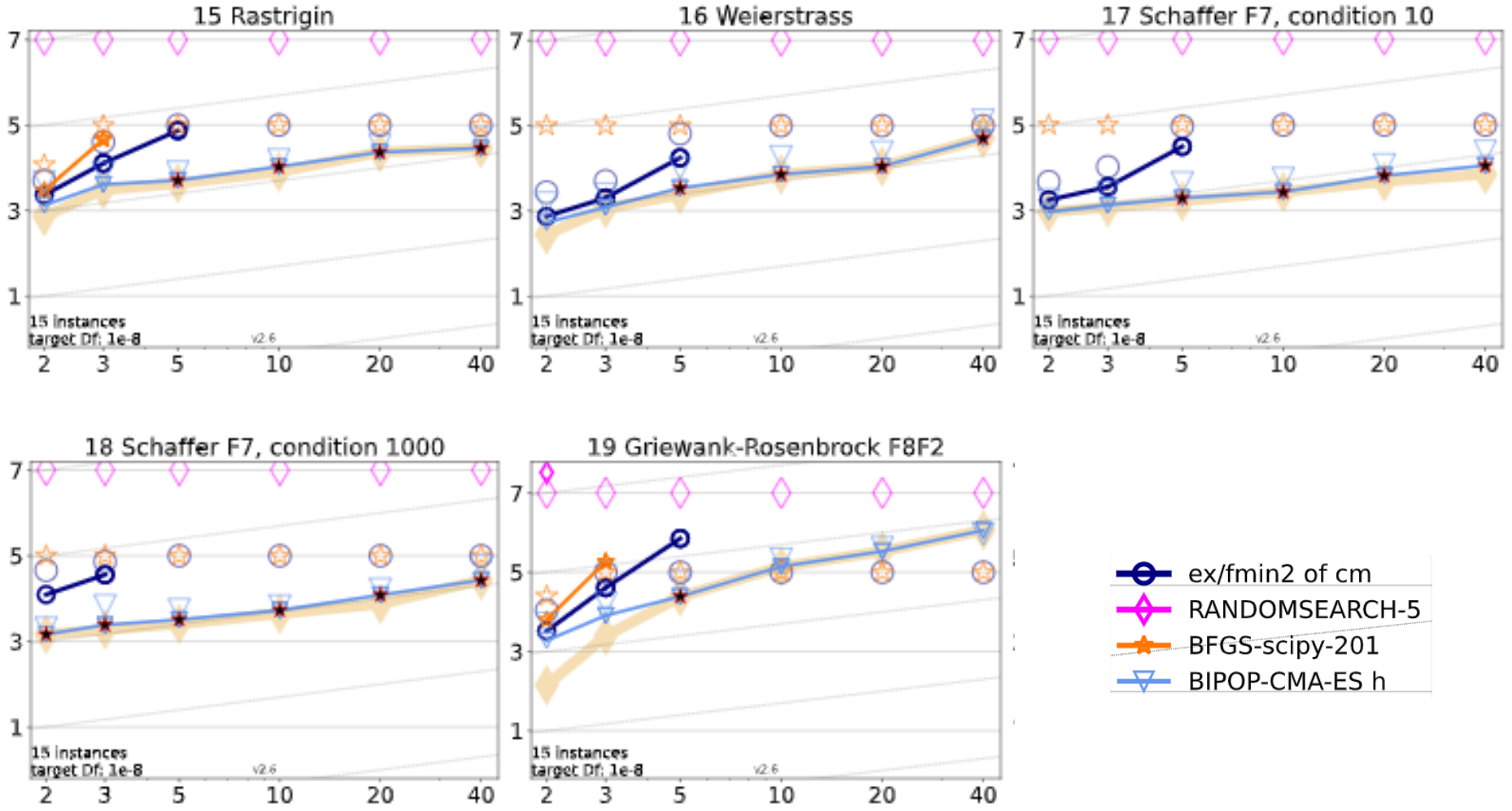
performance of rank one update was very close to Best 2009 for 2D, 3D and 5D.



For separable functions default CMA, rank one and BFGS had same performance among all dimensions.

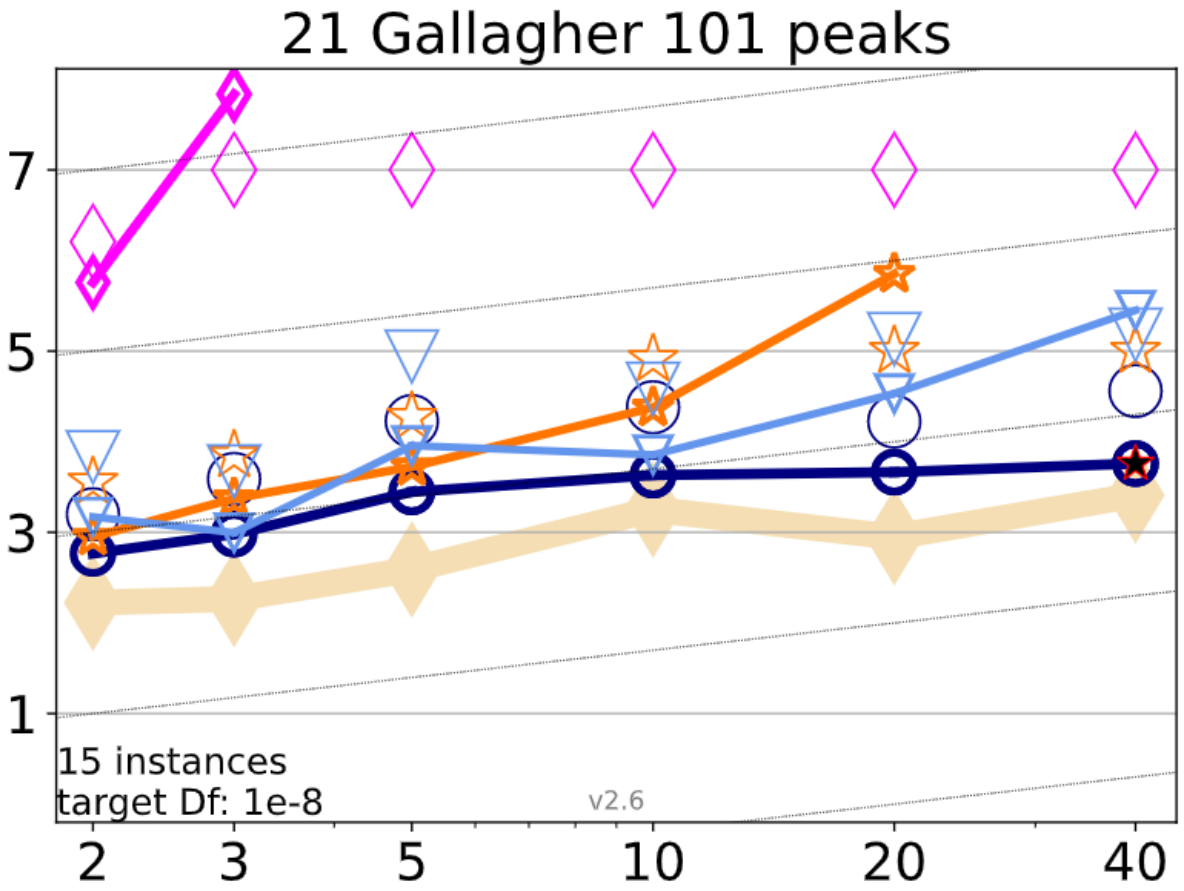


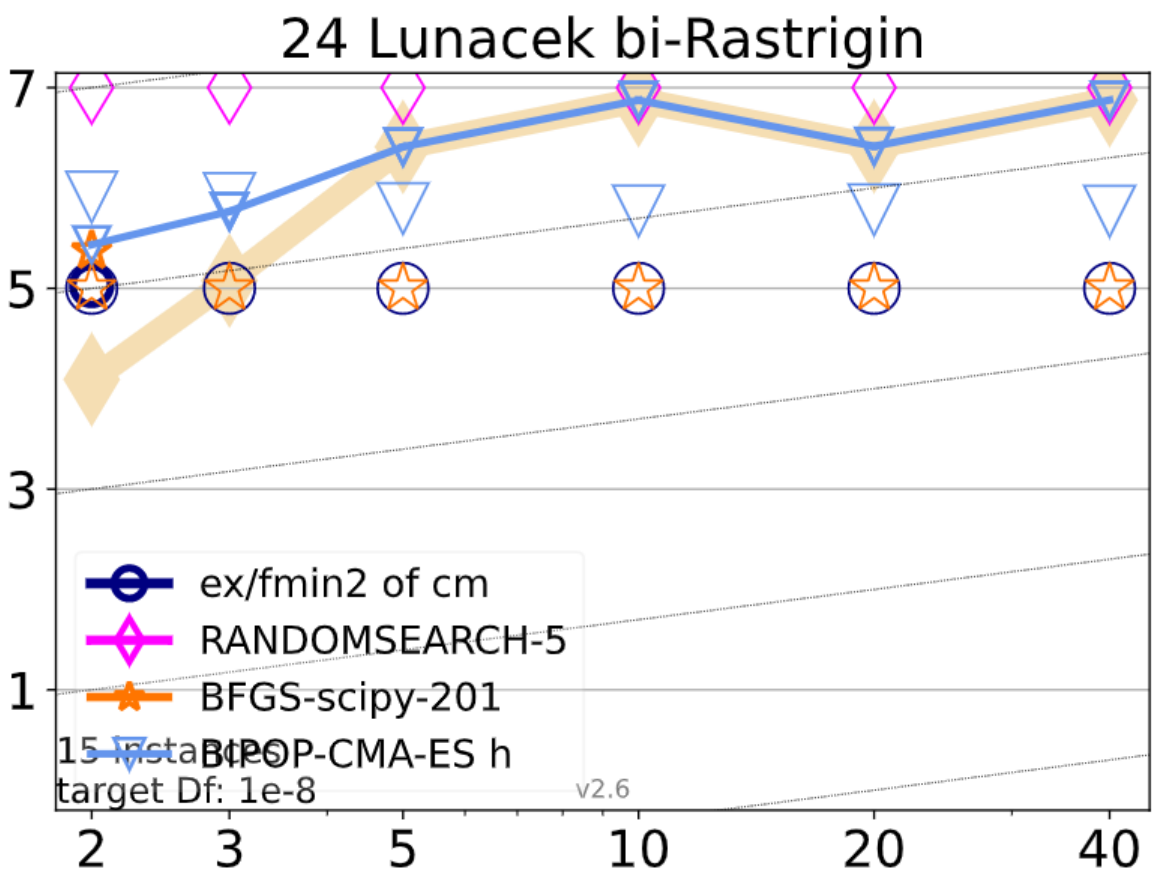
Multimodal functions with adequate global structure, rank one did not solve all dimensions unlike default.



F21 was completely solved with rank one with a performance very close to best 2009 in all dimensions.

Rank-one fails to solve f24 because this function was designed to make evolution strategy fail.





Next: Results With High Population Size

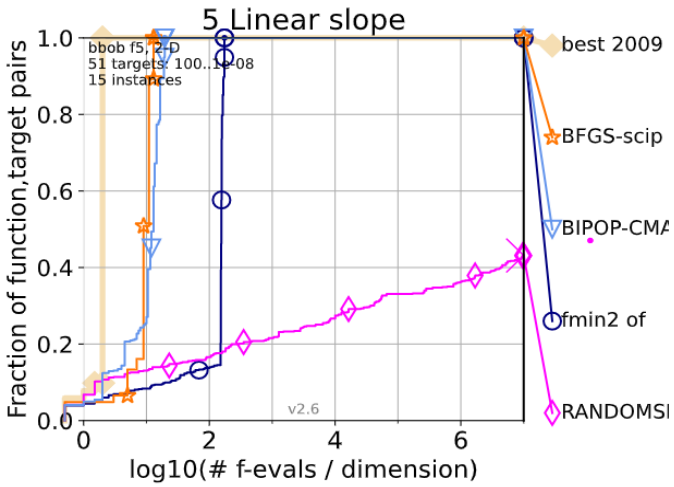
Results for High Population Trial

- **λ very high**

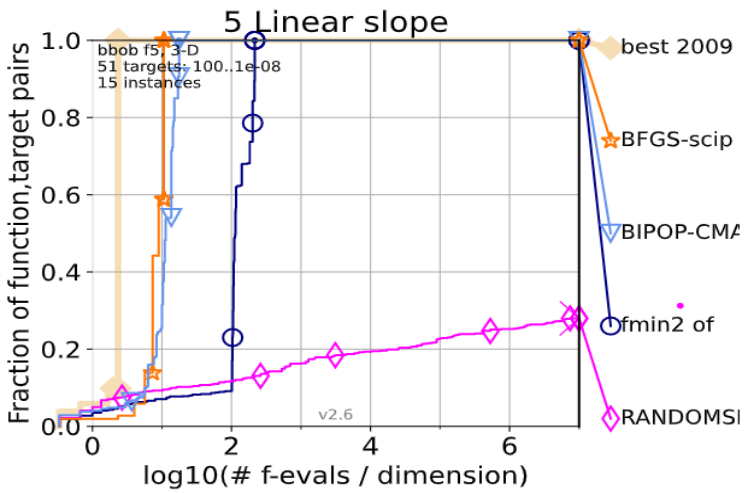
21 Functions which their (instance,target) pairs were not solved

Except for f5 which is linear slope function.

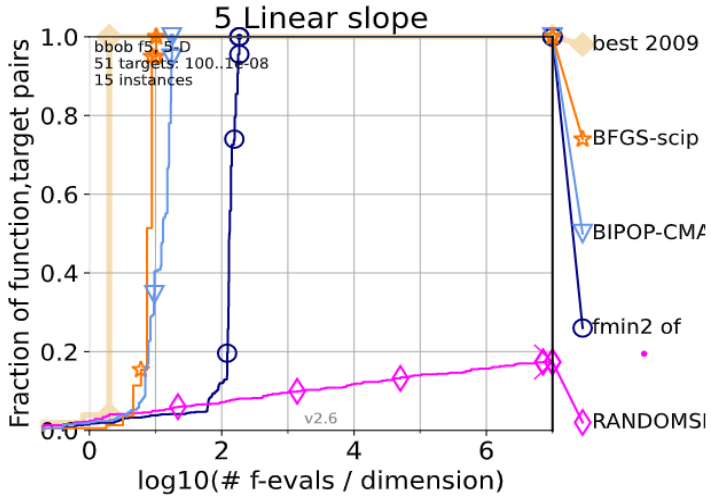
2D



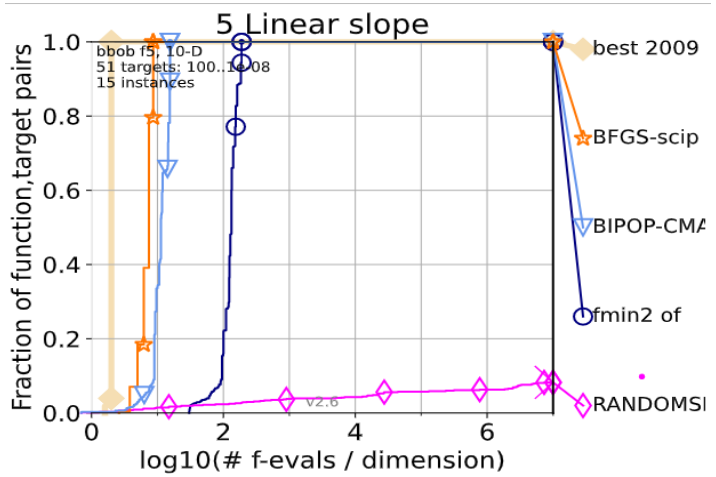
3D



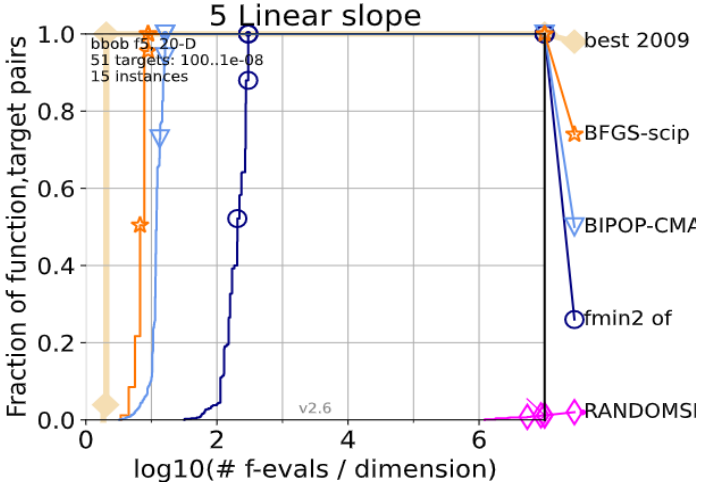
5D



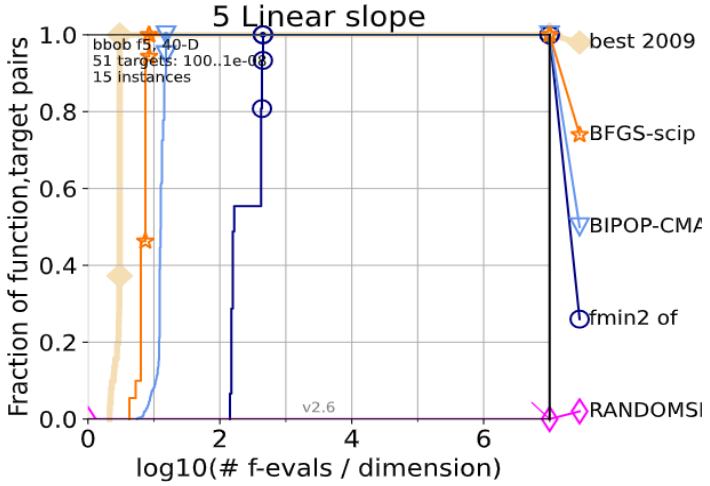
10D



20D



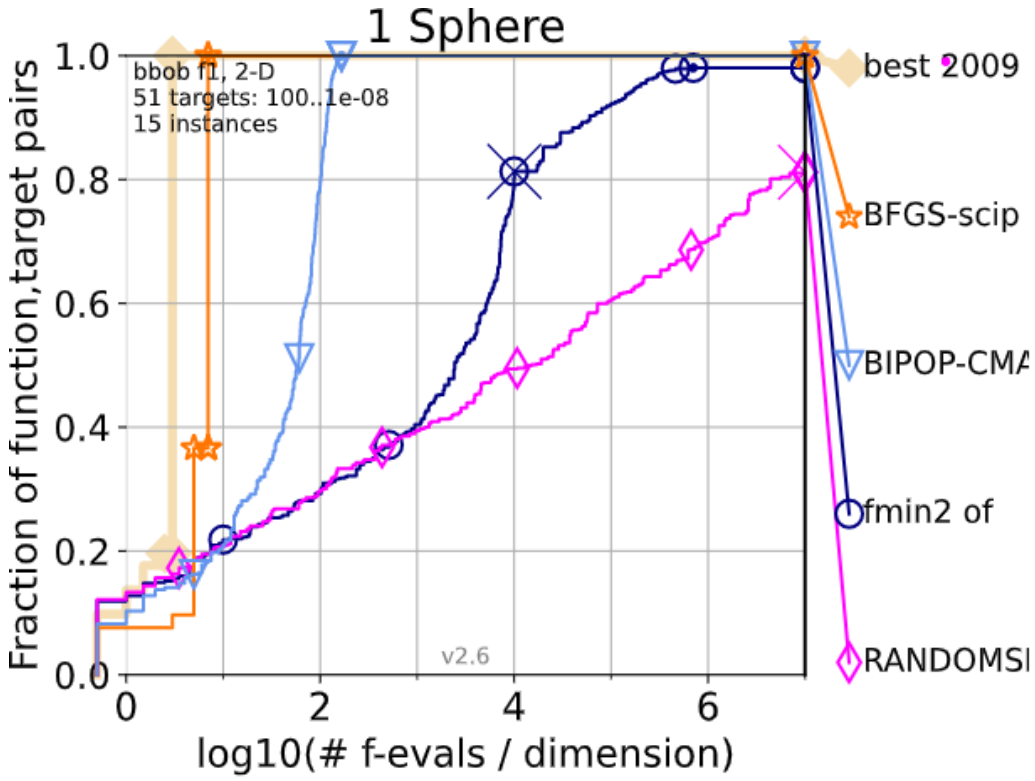
40D



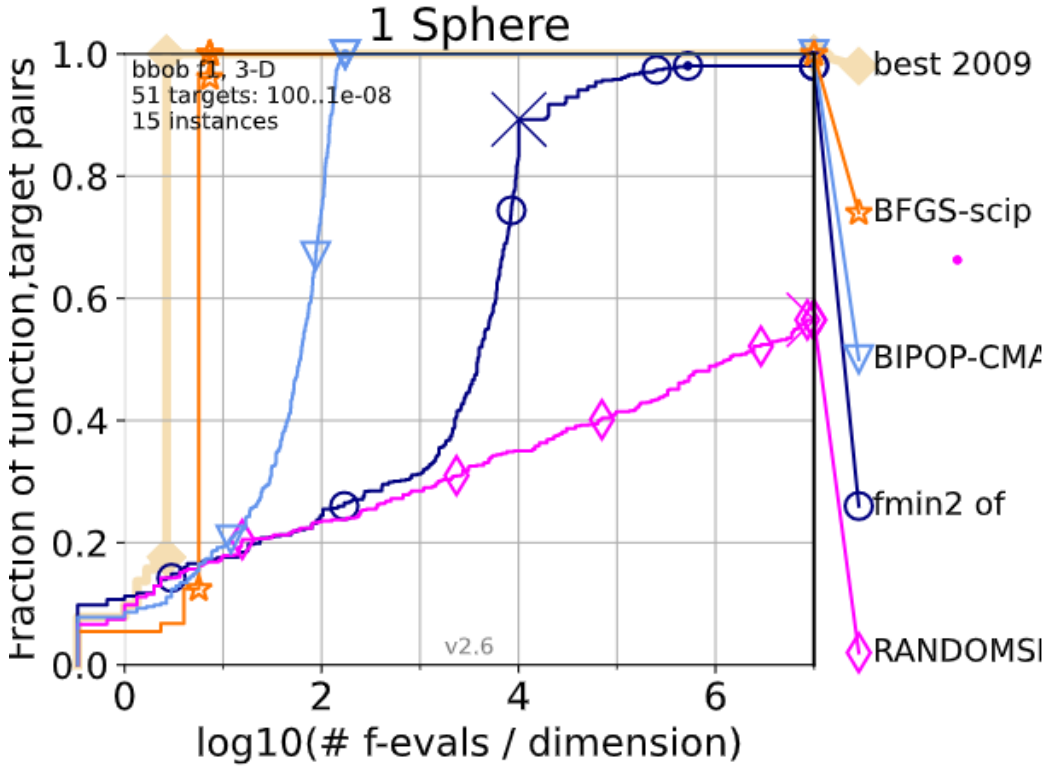
Unusual Notes !!!!

- We saw that the (instance,target) pairs of f1 function (**sphere function**) were not solved in 2D,3D with D*10^4 function evaluations.

2D

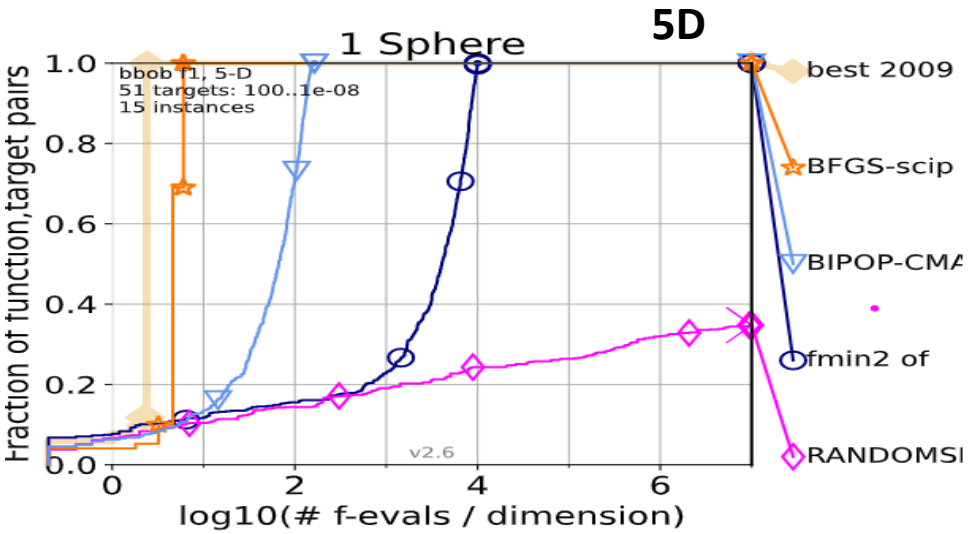


3D

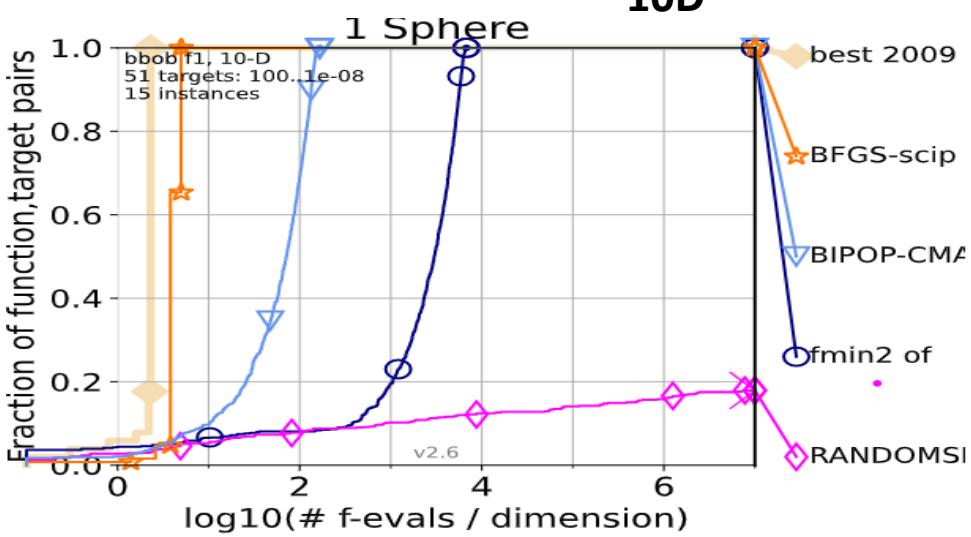


Which for 5D,10D,20D & 40D all the pairs were completely solved with the budget we have 10^4.

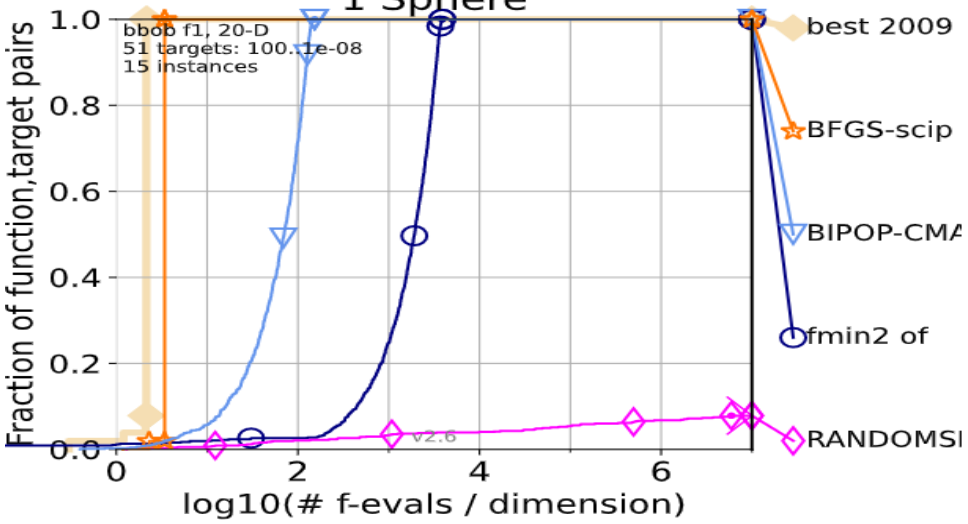
5D



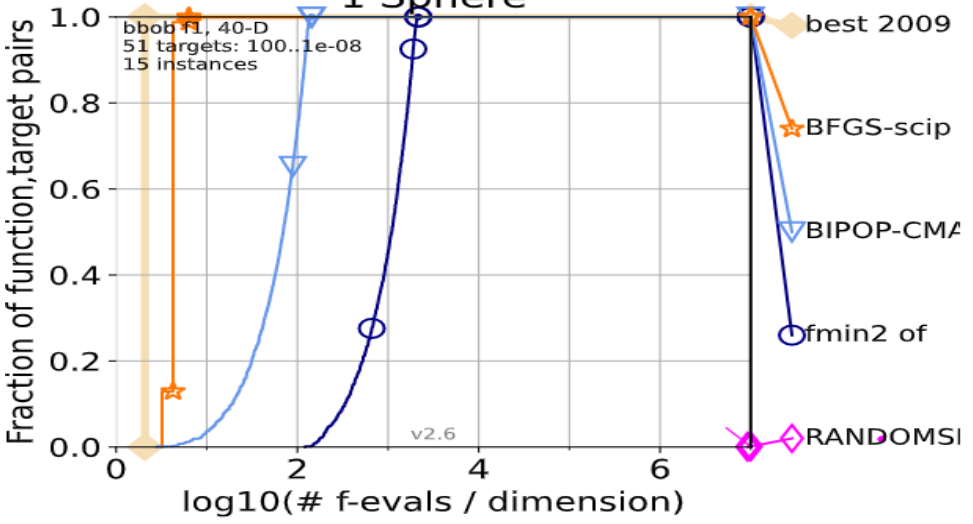
10D



20D

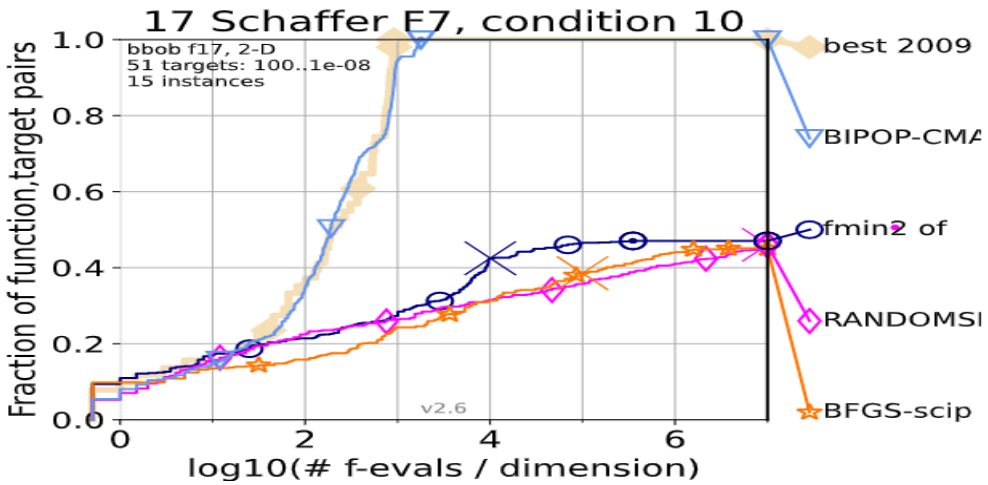


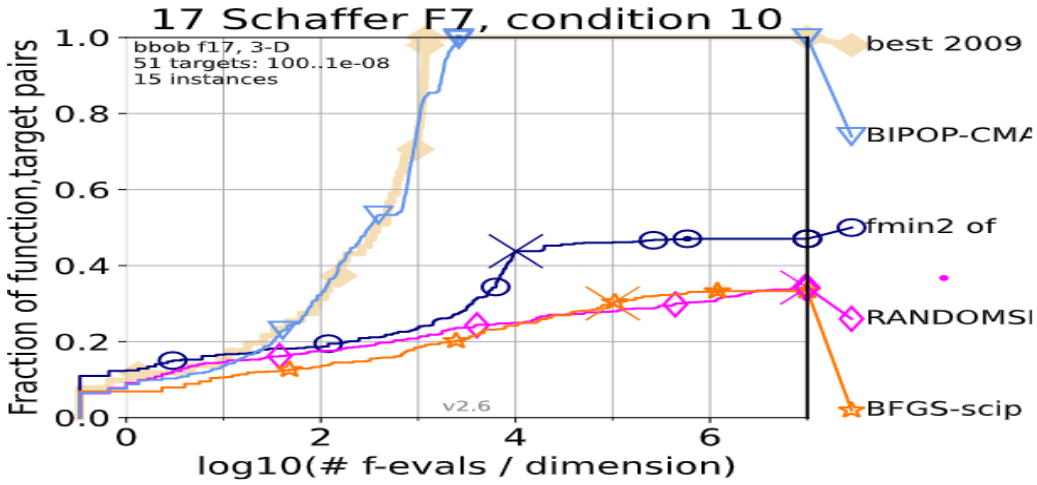
40D

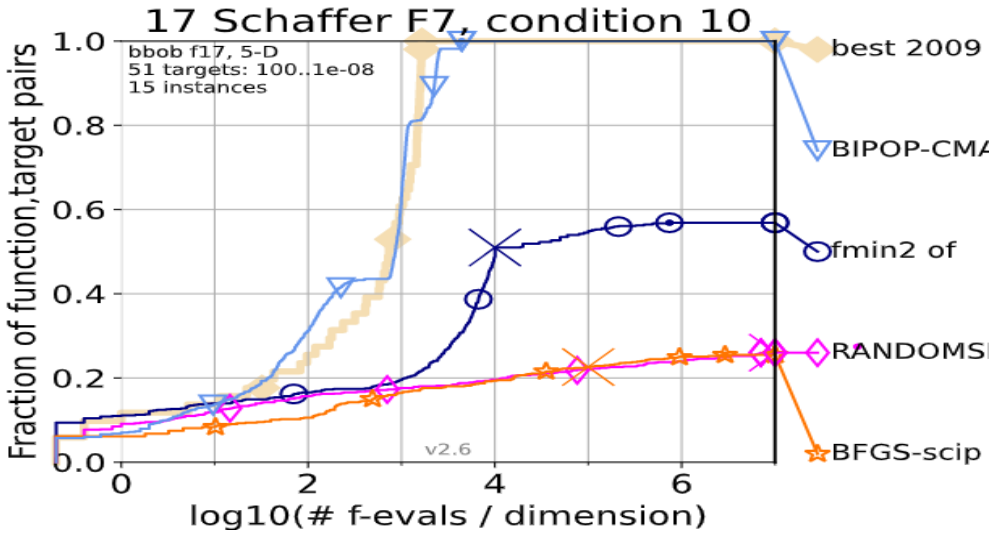


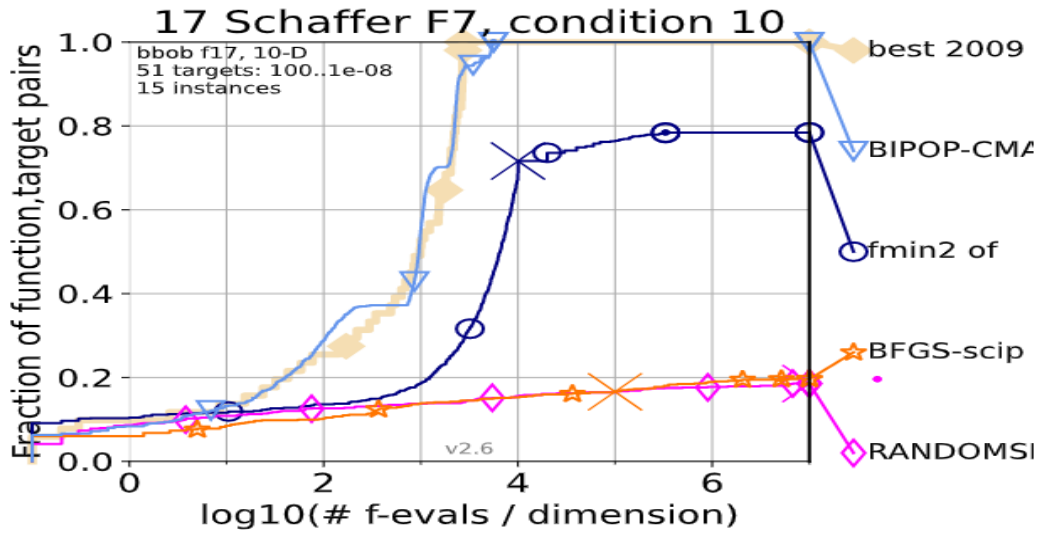
- Also for f17(Schaffers F7 Function)

The pairs were not solved in 2D,3D,5D & 10D

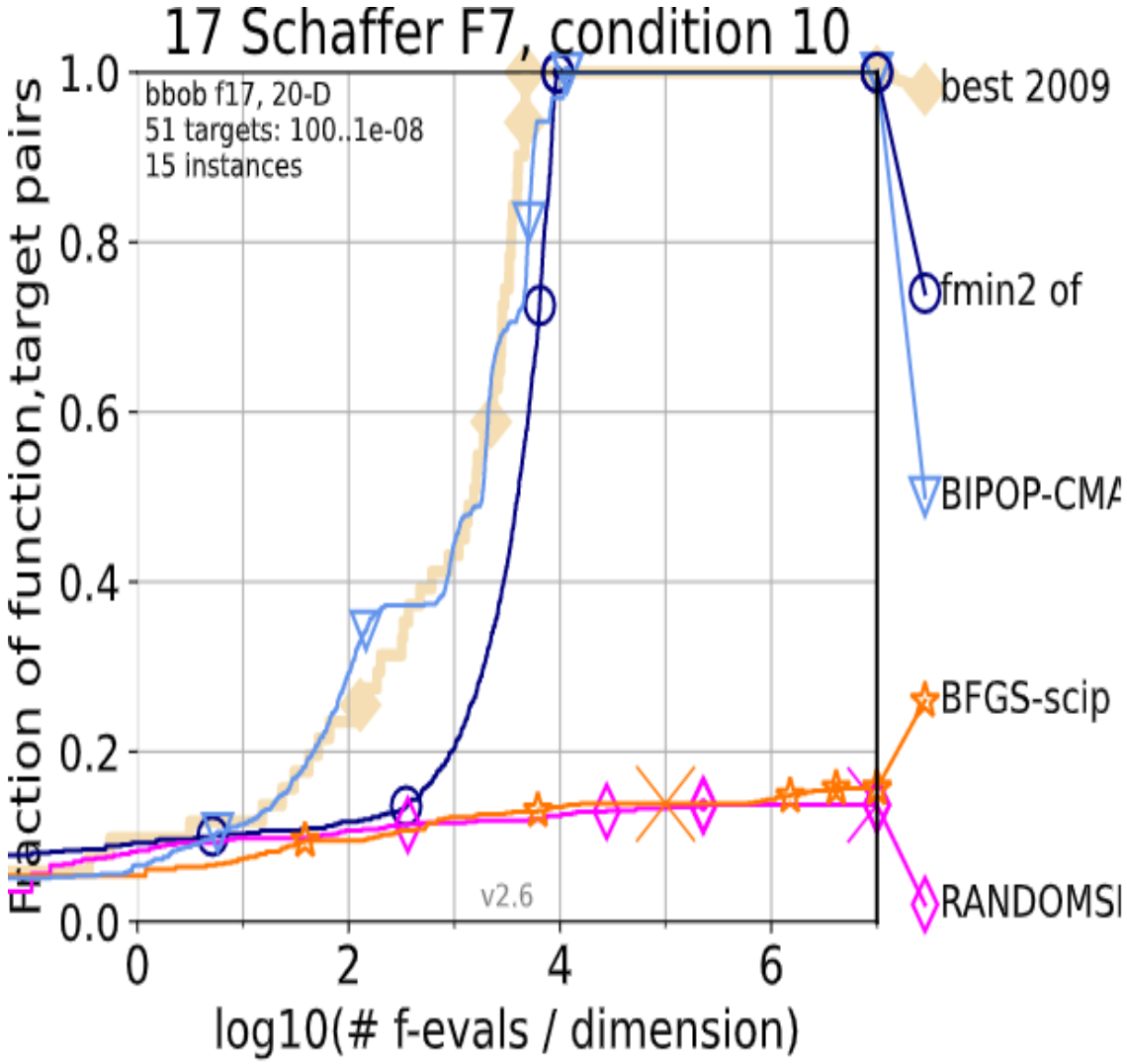


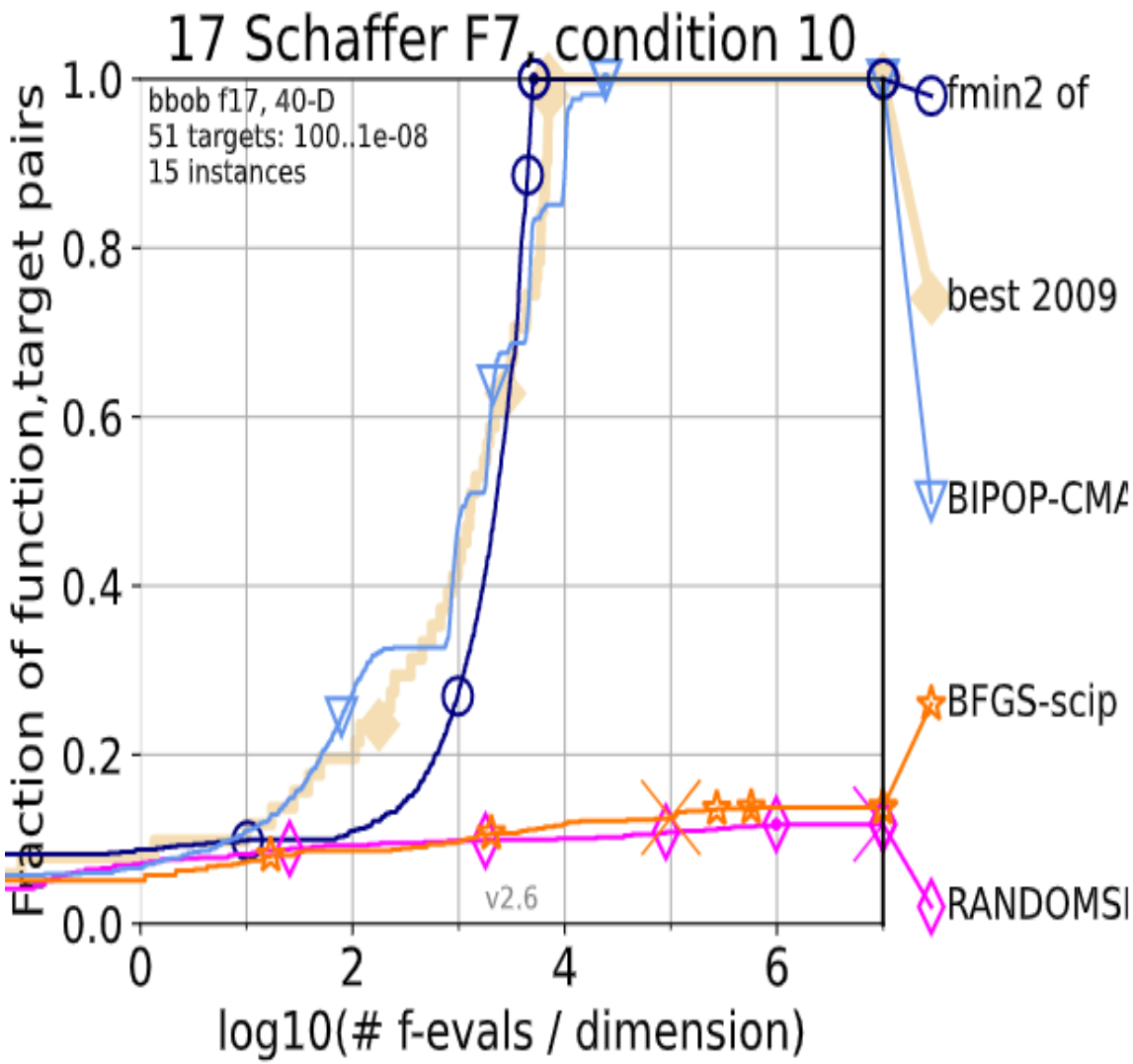




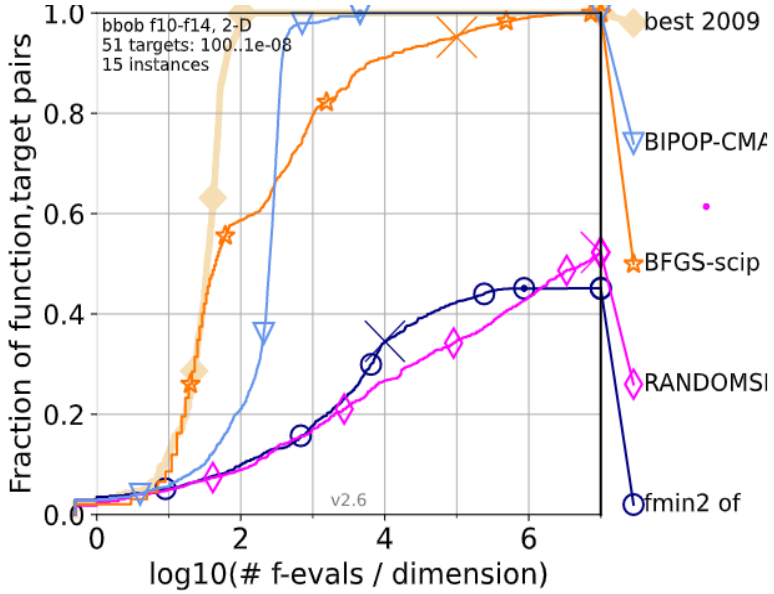


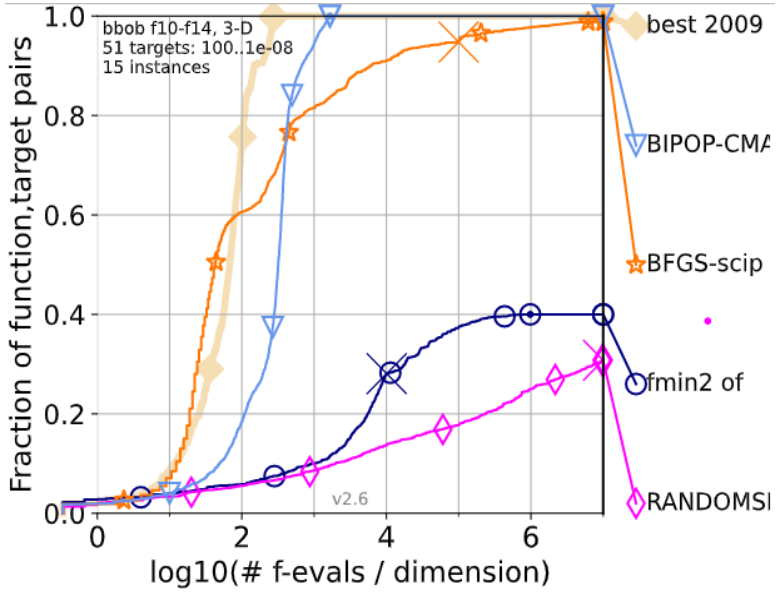
While it got completely solved in 20D & 40D

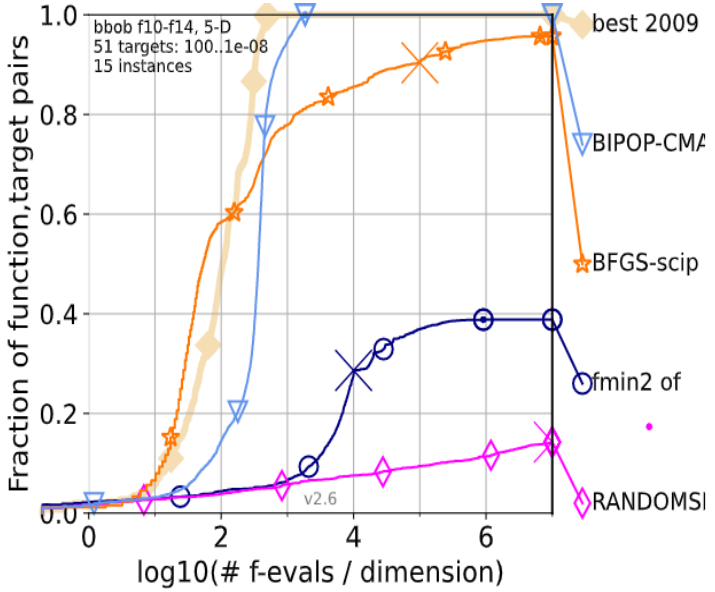


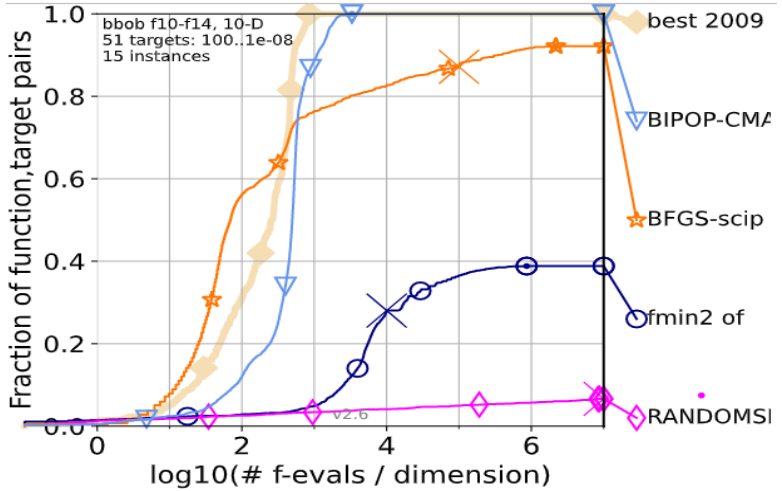


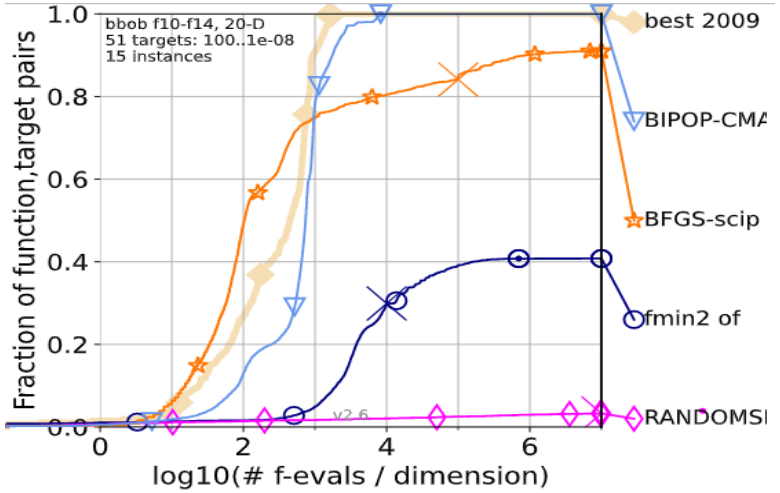
- For the 3rd class (High conditioning and unimodal functions) less than 40% of (instance,target) pairs were solved even with being unimodal , the performance was very low!!

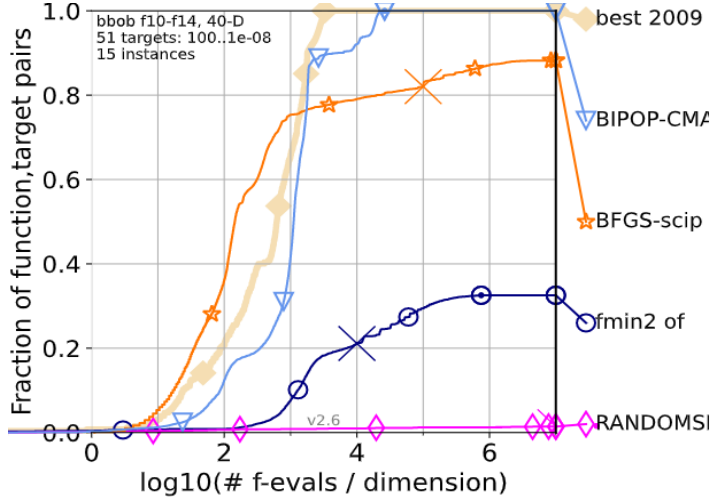




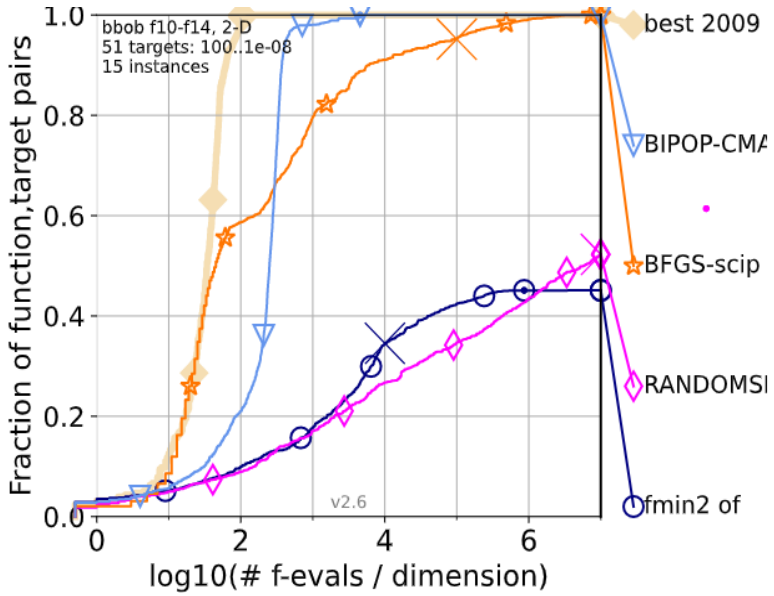


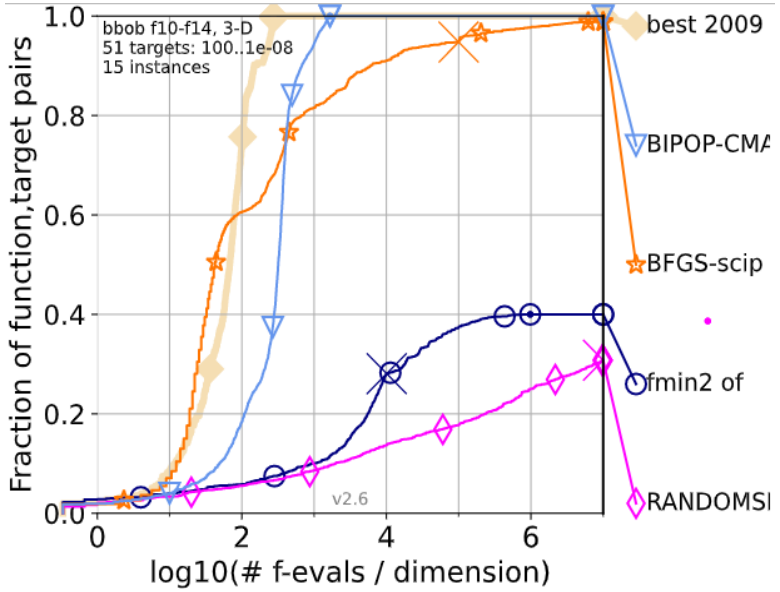


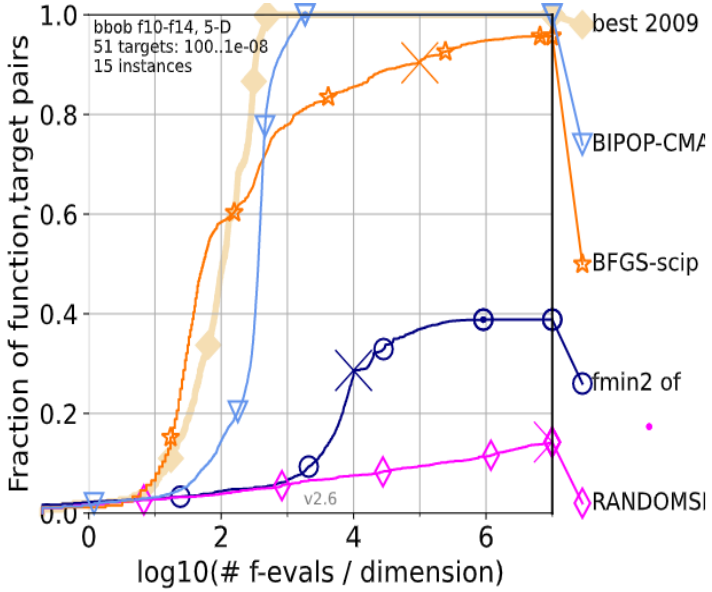


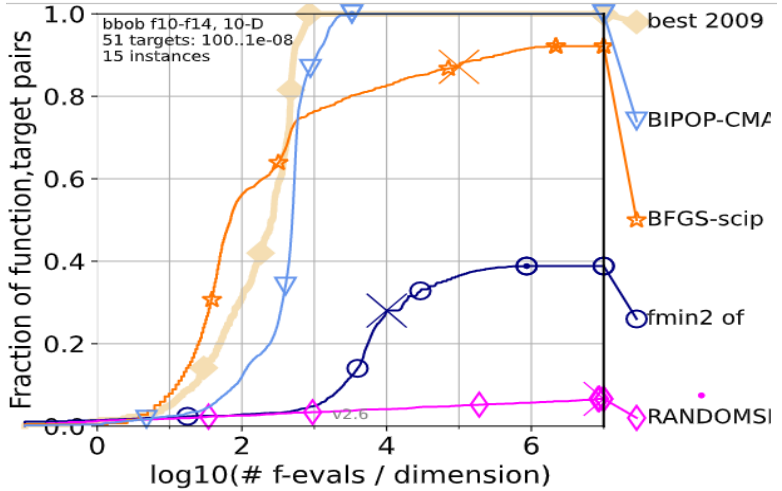


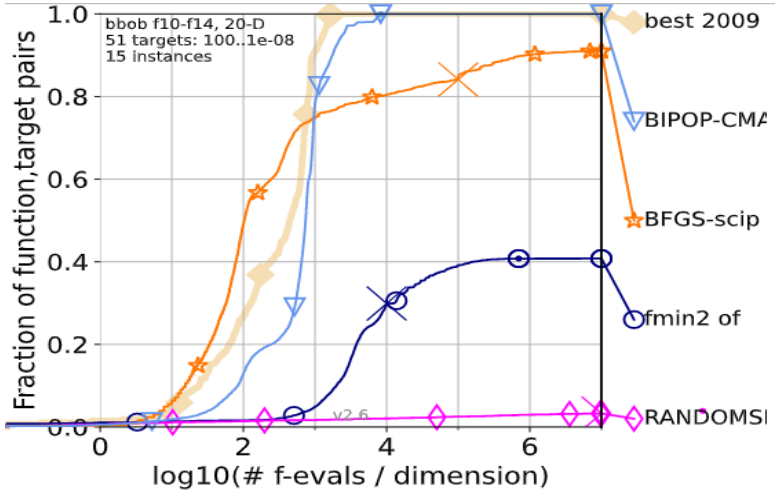
- On the other hand for this class the Scipy algorithm has a very high performance.

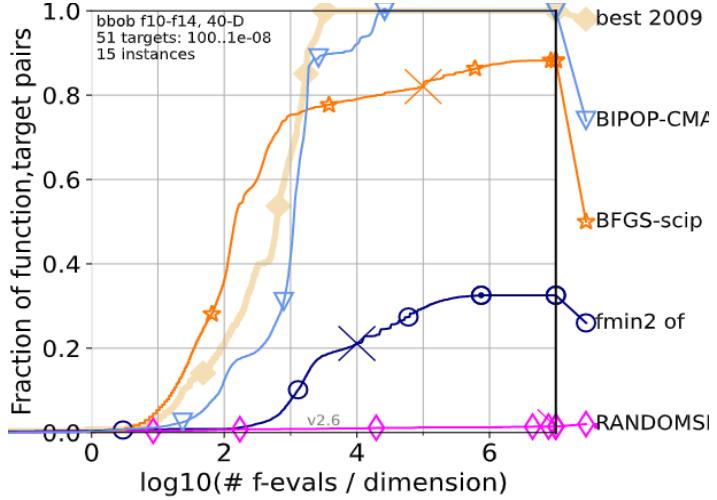




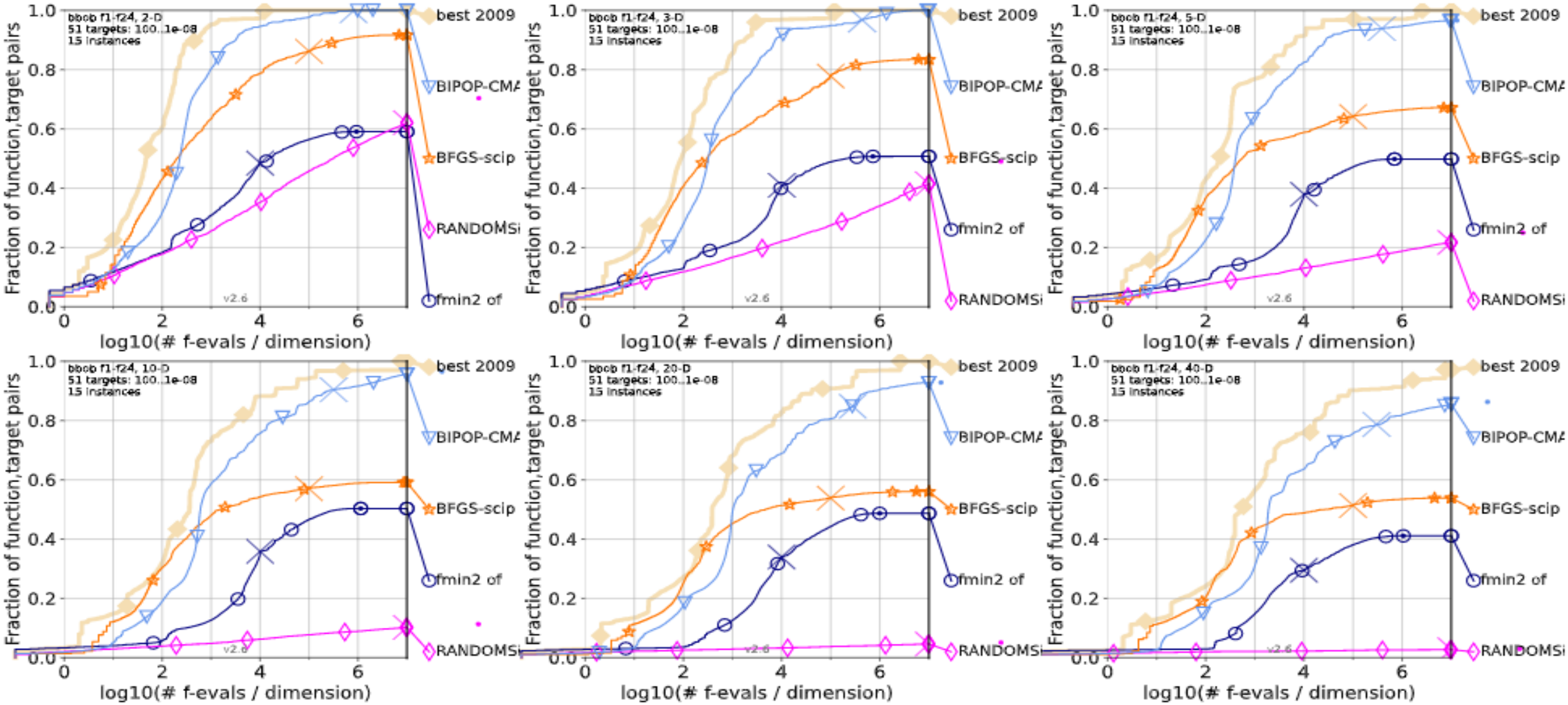






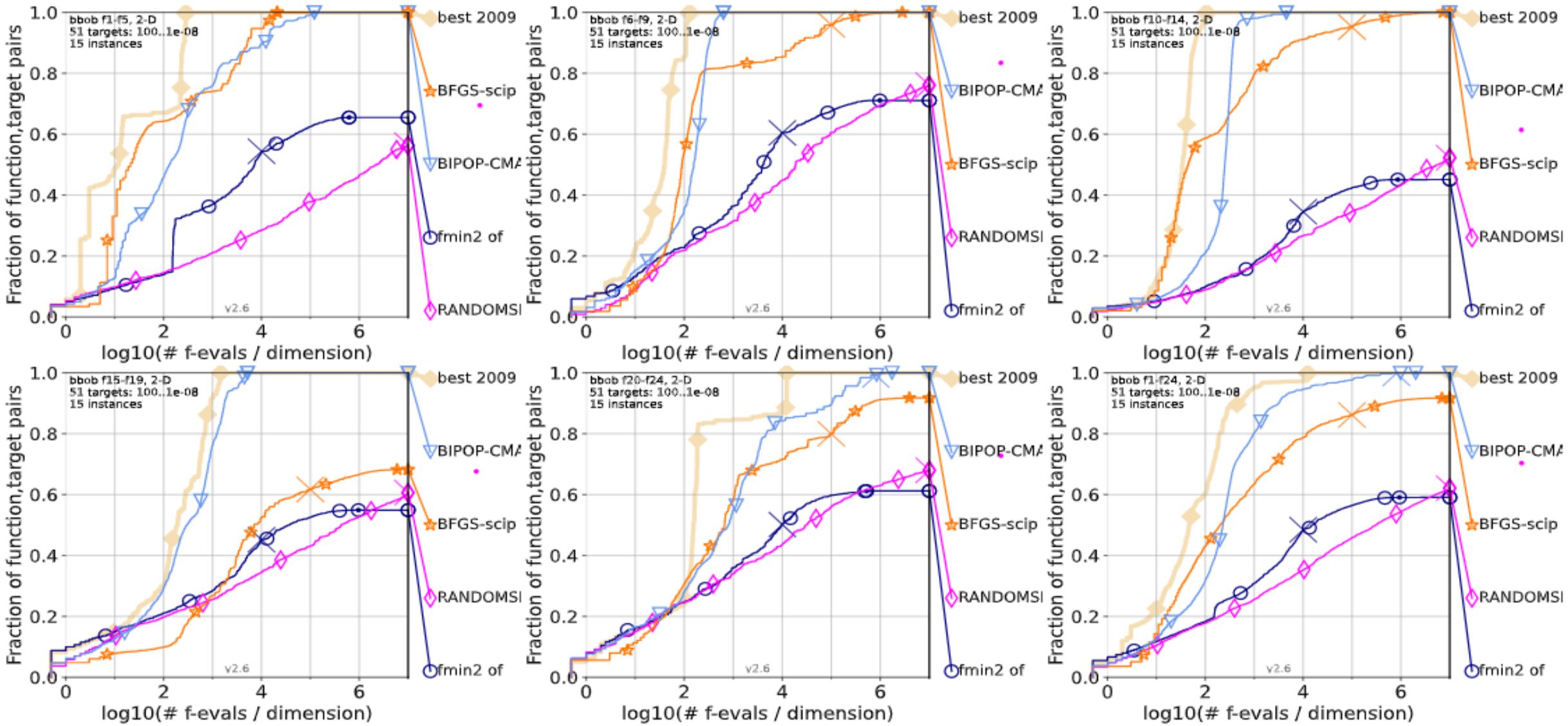


Generally Speaking around 40% of the (instance,target) pairs were solved with rank1 on all functions in all dimensions.

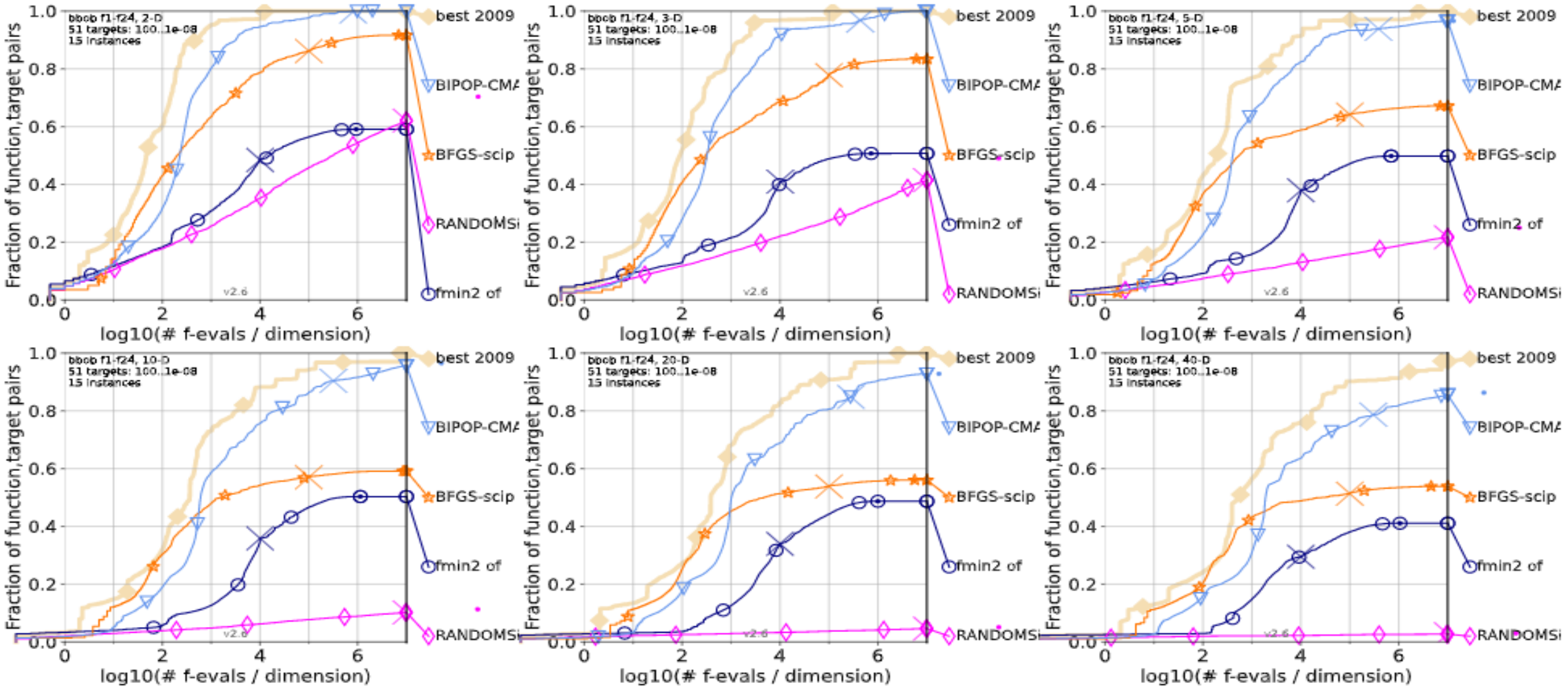


Compare between Rank 1 and other algorithms

- The performance of CMAES with rank 1 update is close to random search algorithm in 2D which was really bad!!



- The performance of BFGs scipy algorithm went down on larger dimensions and become close to CMAES rank 1 update.
- The gap between rank 1 and best 2009 algorithm was very large.



When it’s good to use CMAES with rank1

- Conclusion:

1- Low population size :

- If f is unimodal.
- If f is separable +unimodal.
- In low D : all classes of functions.

2- High population size:

- If f is linear slope function.
- In High D : just 2 functions : f is sphere or chaffers F7.