COCOpf: An Algorithm Portfolio Framework

Petr Baudiš baudipet@fel.cvut.cz

Department of Cybernetics, Czech Technical University, Prague

Goal: Find minimum of a black-box function that we haven't seen before, when we have multiple optimization algorithms available.

Problem: How to switch between the available algorithms so that we don't reach the minimum much slower than the best one?

Contribution: A framework that wraps already implemented algorithms and allows easy testing of selection strategies with a popular benchmark.

Keywords: **Algorithm portfolio, continuous black-box optimization, hyperheuristics, software engineering**

Background

Algorithm Portfolios

Often, we have multiple heuristic algorithms, each suited to a different class of problems. *Algorithm Portfolios* aim to combine them within a single general solver that will choose the best suited algorithm for each input.

For each problem instance on input, we apply a *selection strategy* to pick an algorithm from this portfolio:

- Once or along a fixed schedule *(offline selection)* based on one-time measured features.
- In multiple rounds *(online selection)* allocating time based on their previous performance.

These approaches are not yet combined very often. Here, we focus on the online selection strategies.

Black-box Optimization

Continuous black-box optimization solves the problem of finding a minimum value of a function that is continuous and hidden analytical form.

Vast applications range from operations research to machine learning. Many algorithms are available — simplex algorithms, gradient descent methods or population-based methods.

The de-facto standard for benchmarking optimization methods is the *COmparing Continuous Optimisers* COCO platform. It provides the infrastructure, glue code for both running experiments and preparing high quality figures, a set of common reference results and the code for a set of benchmark functions.

Applying algorithm portfolios on continuous black-box optimization is still a fresh area of research. The main results so far lie either in population methods, combining a variety of genetic algorithms together in non-black-box fashion, or in offline methods based chiefly on exploratory landscape analysis.

The COCOpf Python Framework

Paradigm: Algorithms are *black-box*, i.e. we completely avoid modifying their actual code and we simply call them to retrieve a result of single iteration.

Platform: We extend the Python implementation of the COCO platform. We can easily benchmark different selection strategies and can base our portfolio on optimization algorithms from the SciPy library.

Implementation: Multi-class Python module that can:

- Run the experiment on a given set of benchmark scenarios, much more conveniently than the default COCO approach
- Maintain a population of instances (that is, algorithms from the portfolio in a particular state)
- Wrap third-party minimization algorithms to support resume and suspend after every iteration using a simple callback function; we use Python threads, queues and exceptions in a fancy way
- Generate dataset summaries and plots for publication

Each strategy is a standalone script that uses the `cocopf` module for heavy lifting.

Algorithm wrappers for the SciPy optimization methods and the reference implementation of CMA are available out-of-the box as well as two example algorithm selection strategies that are benchmarked below.

Availability: The framework is publicly available as free software under the permissible MIT licence at `https://github.com/pasky/cocopf`.

Our Reference Portfolio

We have chosen the six stock minimizers provided by SciPy v0.13 that are available for direct use:

- **Nelder-Mead**, the Simplex algorithm.
- **Powell**, the tweaked Powell's conjugate direction method.
- **CG**, the nonlinear conjugate gradient Fletcher-Reeves method.
- **BFGS**, the quasi-Newton method of Broyden, Fletcher, Goldfarb and Shanno.
- **L-BFGS-B**, the limited-memory variant of BFGS with box constraints.
- **SLSQP**, the Sequential Least SQuares Programming with box constraints.

These are local minimizers, therefore we use a SciPy wrapper of the **Basin Hopping** restart strategy; conceptually similar to Simulated Annealing with a fixed temperature.

We also included the popular **CMA** algorithm (genetic algorithm with the Covariance Matrix Adaptation evolution strategy). It converges slowly on reasonable functions but it can beat even many difficult targets.

Selection Strategies

We implemented some reference selection strategies to demonstrate the usage of our framework.

- The **UNIF** strategy performs a number of rounds where in every round, a uniformly randomly selected algorithm is iterated once.
- The **EG50** strategy performs a number of rounds where in every round, the *epsilon-greedy policy* selects an algorithm to be iterated once. The algorithm with the currently best solution is run with $p = 0.5$, a randomly chosen algorithm otherwise.

Sneak peek: Later research using the COCOpf framework has shown that EG50 actually beats most other strategies we tried! Only two strategies based on the UCB1 multi-armed bandit with smart value rescaling have beaten EG50.

Future Work

The currently chosen reference portfolio is somewhat ad hoc and unbalanced with CMA dominating in many functions. This makes it actually an interesting testbed, but we need to add more high performance algorithms anyway.

It turns out that most selection strategies can be decomposed to mostly independent credit assignment and credit accrual schemas. COCOpf support for this may enable easy recombination of strategies.

We would like to contribute at least parts of our code back to the COCO framework, and eventually allow Scipy users build optimization algorithm portfolios.

Acknowledgements

This research is supervised by Dr. Petr Pošík and supported by the CTU grant SGS14/194/ OHK3/3T/13 "Automatic adaptation of search algorithms".

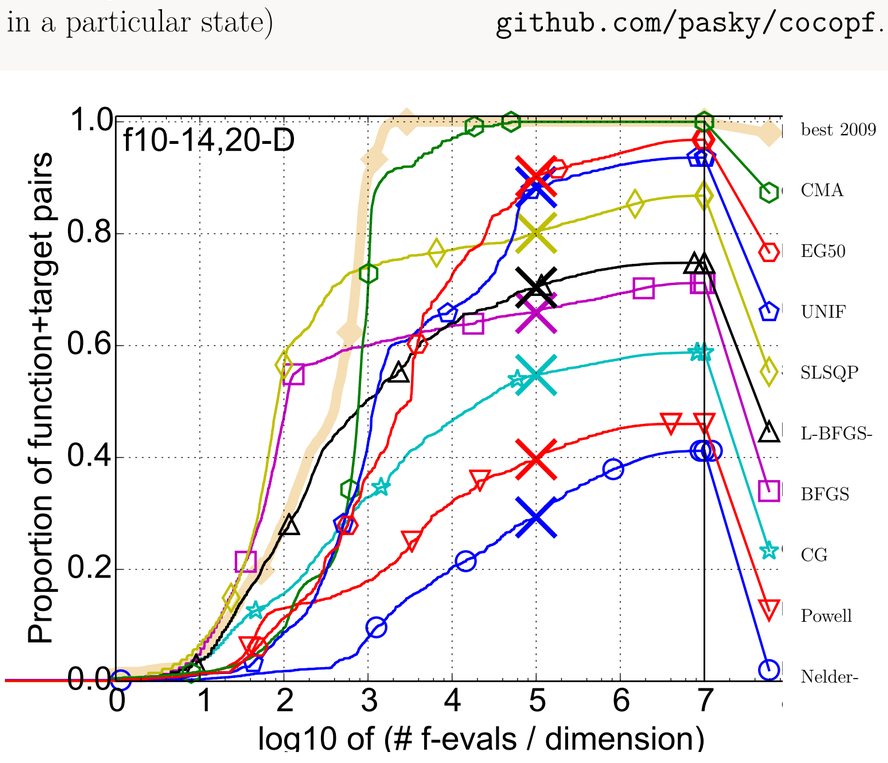

Figure: Bootstrapped empirical cumulative distribution of FEvals/D for 50 targets in $10^{[2,-8]}$ of *ill-conditioned functions* in 20-D. All algorithm portfolio members and the UNIF and EG50 strategies are shown. Note that for some early targets, BFGS and SLSQP beat the "best 2009" reference; we believe this is something not known in the optimization research community.

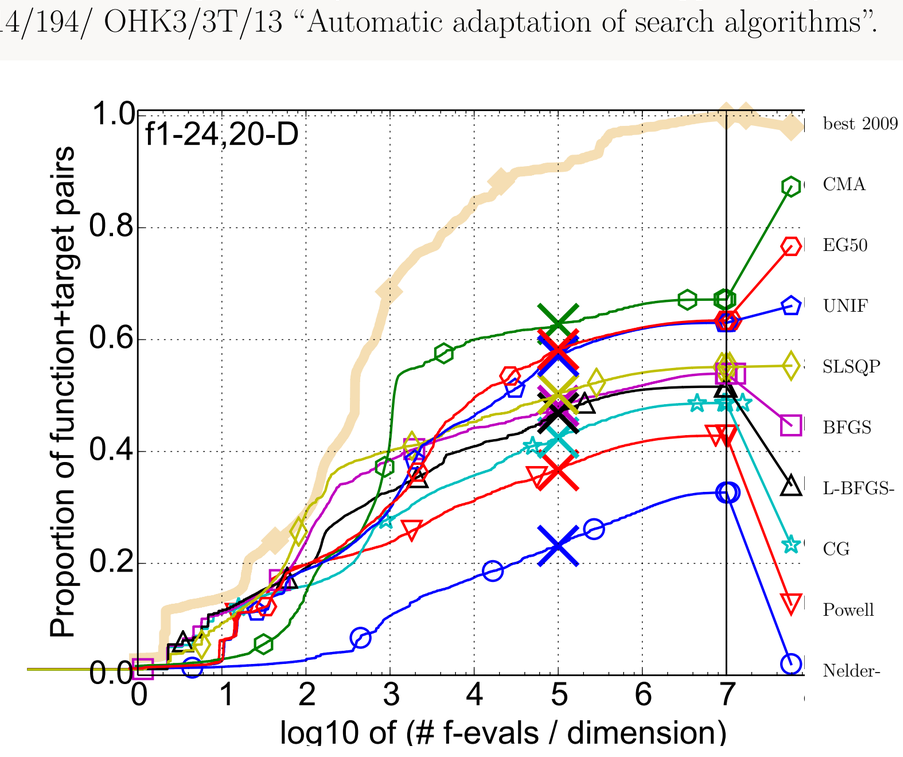

Figure: Bootstrapped empirical cumulative distribution of FEvals/D for 50 targets in $10^{[2,-8]}$ of *all functions* in 20-D. All algorithm portfolio members and the UNIF and EG50 strategies are shown.