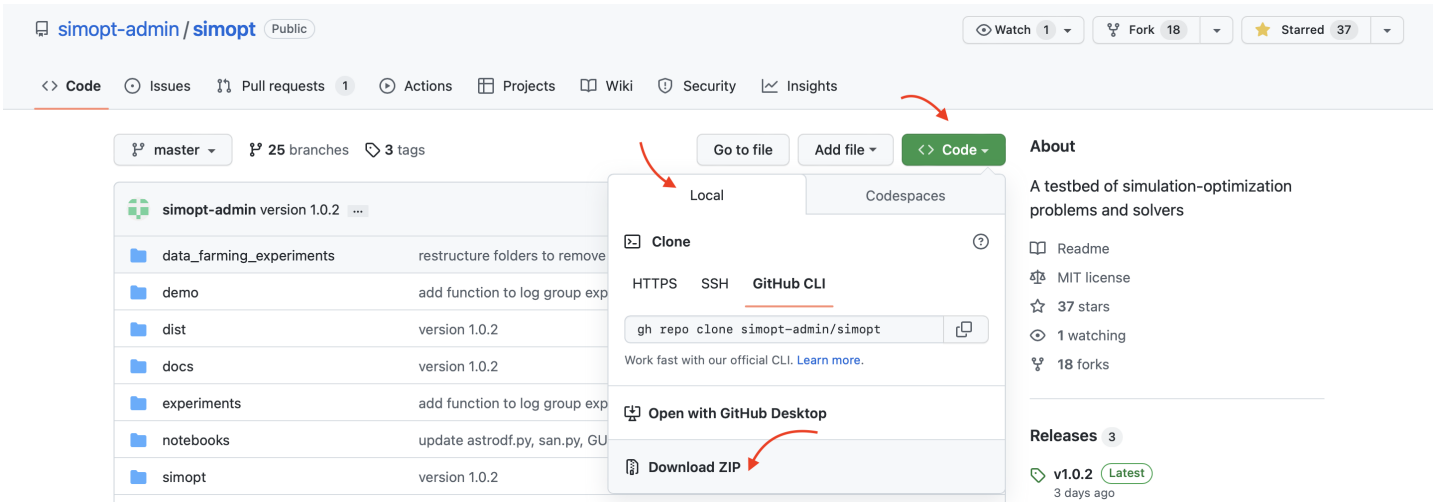
Reminder (you may have already done this)

1. In your browser, navigate to `https://github.com/simopt-admin/simopt`.
2. Click on "Download ZIP" as shown.



3. Unzip the folder `simopt-master` and open it in VS Code using "File > Open Folder".

Please create a virtual environment

Open a terminal inside VSCode by clicking on Terminal > New Terminal from the menu. Inside the terminal, type the following to create a virtual environment:

- `python -m venv venv`
- `venv\Scripts\activate`

  or on a Mac

  `source venv/bin/activate`
- `python -m pip install simoptlib`
- `python -m simopt.GUI`

SimOpt

WSC 2023 Workshop, San Antonio, TX

David J. Eckman, Texas A&M University
Shane G. Henderson, Cornell University
Sara Shashaani, North Carolina State University

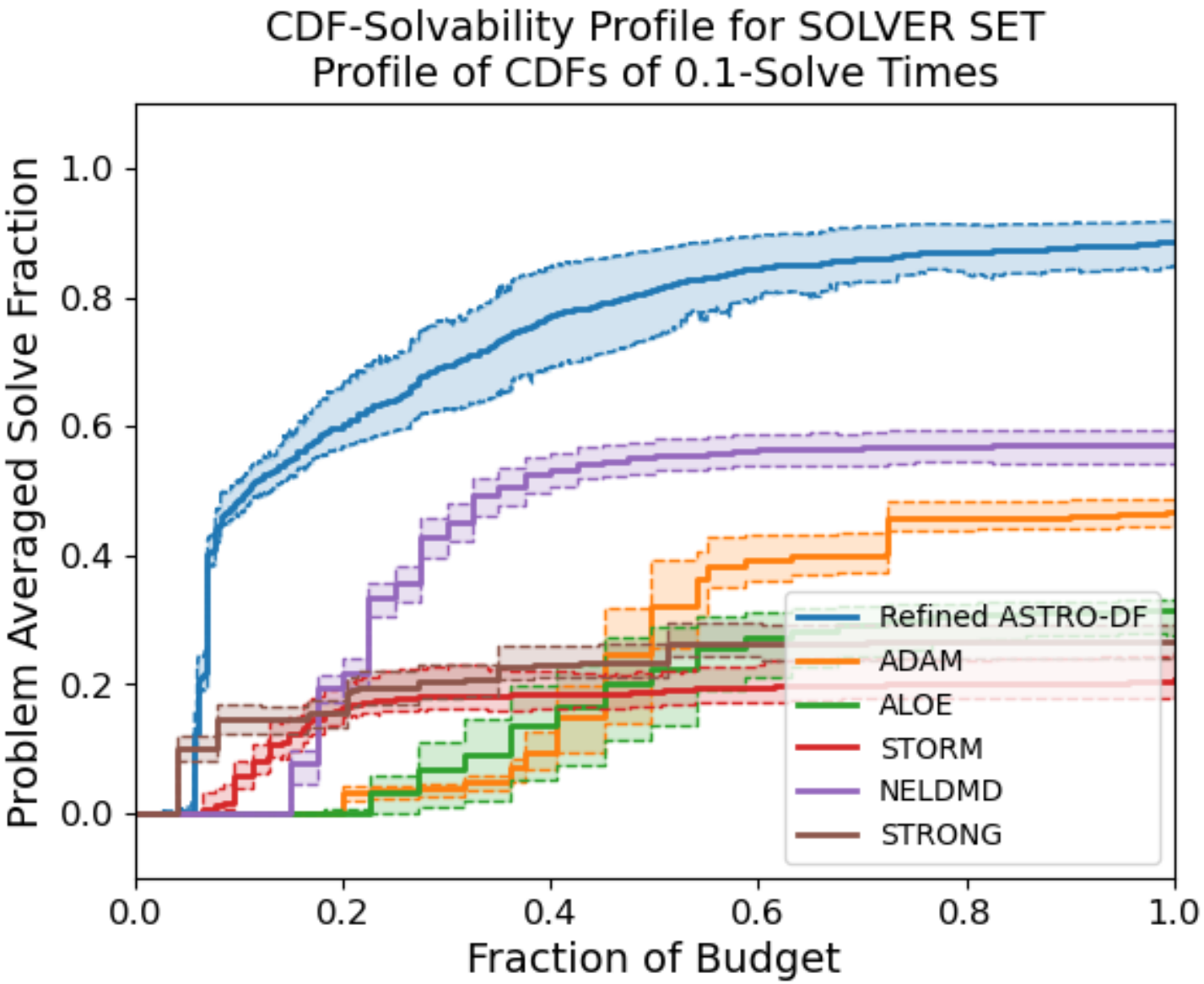
The Big Picture I

SimOpt is a library of simulation-optimization problems and solvers, intended

1. To benchmark solvers to aid with solver development
2. To focus attention on the finite-time performance of solvers
3. To identify important/difficult classes of problems
4. For use mostly by simulation-optimization researchers and solver developers

Users of SimOpt should be comfortable using Python tools and GitHub.

The Big Picture II



Goals and plan

Our goal in this workshop is for you to learn how to:

- Run SimOpt experiments
- Understand the plots
- Use the GUI
- Build and contribute your own models, problems and solvers
- Run a data farming experiment in SimOpt

The plan:

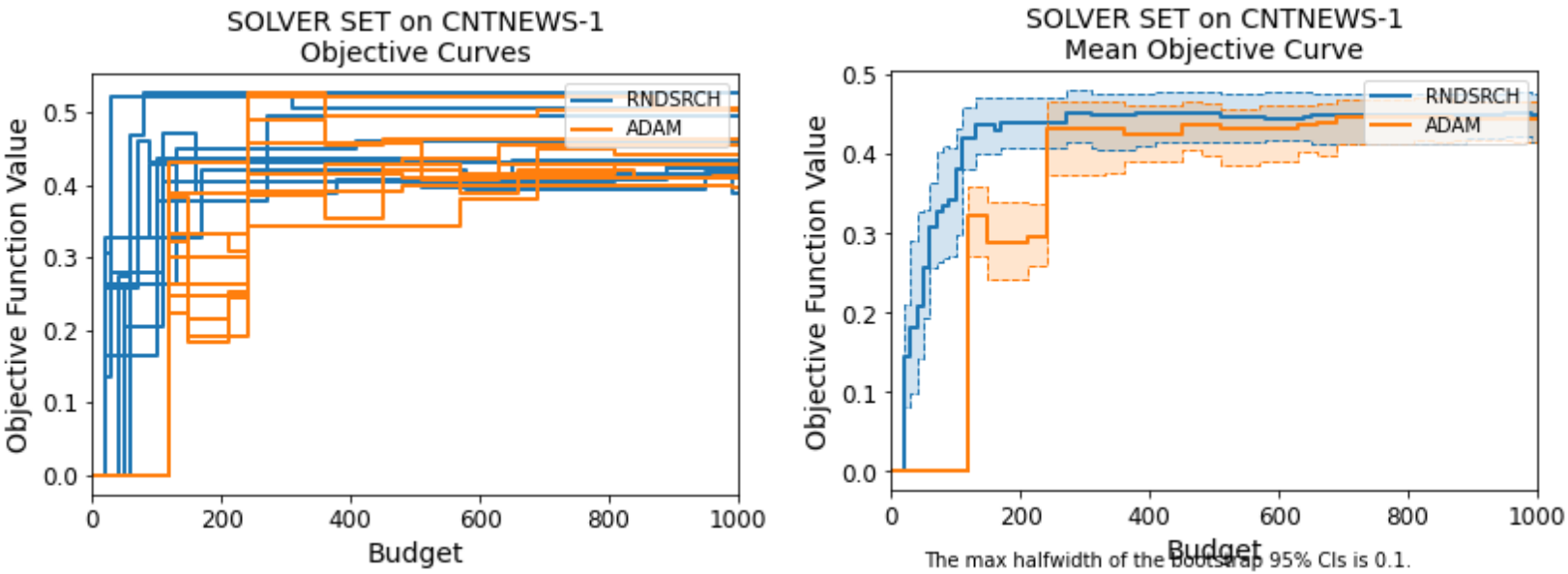
1. Comparing solvers with cool plots
2. Using the SimOpt GUI
3. Writing your own models, problems, and solvers
4. See an example of data farming

Comparing solvers

Newsvendor problem

- Per-unit order cost $c = 5$, selling price $s = 9$, salvage value $w = 1$
- Real-valued order quantities
- Demand $D \sim F(x) = 1 - (1 + x^{\alpha})^{-\beta}$ where $x \geq 0, \alpha = 2, \beta = 20$ (this is Burr Type XII)
- Maximize expected profit, where we use simulation to estimate expected profit rather than explicit formulas
- Which of the following two solvers is better?
  1. ADAM
  2. Random search (exp(1)), with 10 reps per solution
- Budget of 1000 replications, start at $x_0 = 0$

Progress curves



(a) for 10 macroreplications

(b) mean and CI

Figure: Unnormalized progress curves.

Important SimOpt Objects

model A simulation model has a replicate method that generates one or more replications. Models can have many factors like c, s, w, α, β in continuous newsvendor

problem Built from a model, with objective function, decision variables, constraints. Many problems can be generated from a single model.

solver solvers tackle one or more problems, take simulation replications sequentially, and report a "running estimated best solution"

generators SimOpt makes careful use of common random numbers through a custom implementation of the generator MRG32k3a.

Problems

$$\begin{aligned}
\min_x \; & f(x, w) = \mathbb{E} f(x, w, \xi) \\
\text{s/t } & g(x, w) = \mathbb{E} g(x, w, \xi) \leq 0 \\
& h(x, w) \leq 0 \\
& x \in \mathcal{D}(w).
\end{aligned}$$

Problems have a problem-specific *budget* T measured in simulation replications

Problems and Solvers

- Problems can have continuous variables, integer-ordered variables, or a mixture of these.
- We have many more solvers for continuous-variable problems than for integer-ordered-variable problems.
- We don't have solvers for stochastically constrained problems yet.
- These are research opportunities!
- Solvers can use knowledge of the budget T if they want, e.g., to set steplengths in SGD.
- Some solvers use a random budget, e.g., R&S with statistical guarantees. Currently, we don't support random-budget solvers

Time, macroreplications, postreplications, bootstrapping

We measure time t through the fraction of the budget T that has been used, $t \in [0, 1]$

A macroreplication is a single run of a single solver on a single problem. The solver generates an estimated best solution as a function of time $(X(t) : 0 \leq t \leq 1)$

After the macroreplications are complete we use postreplications to estimate $(f(X(t)) : 0 \leq t \leq 1)$ for each macroreplication

Both macroreplications and postreplications are stochastic. We use bootstrapping to provide confidence intervals for $\mathbb{E}f(X(t))$ and related quantities at each $t \in [0, 1]$

Your turn

Generate unnormalized progress curves, aggregated unnormalized progress curves and terminal progress violin plots.

1. Run the GUI using the steps in the README.
2. Click "Create Problem-Solver Group" and select the Random Search and ADAM solvers and "Max Profit for Continuous Newsvendor".
3. Click "Add Cross-Design Problem-Solver Group".
4. Click "Run" in the Workspace area below and wait until the other button are activated.
5. Click "Post Process and Normalize," and in the new window use the defaults and click "Post-Process".
6. Click "Plot".
7. From the new window, select the problem and two solvers, select the plot type, unselect the "Normalize by Relative Optimality Gap", and click "Add" then "View Plot" that appears below.

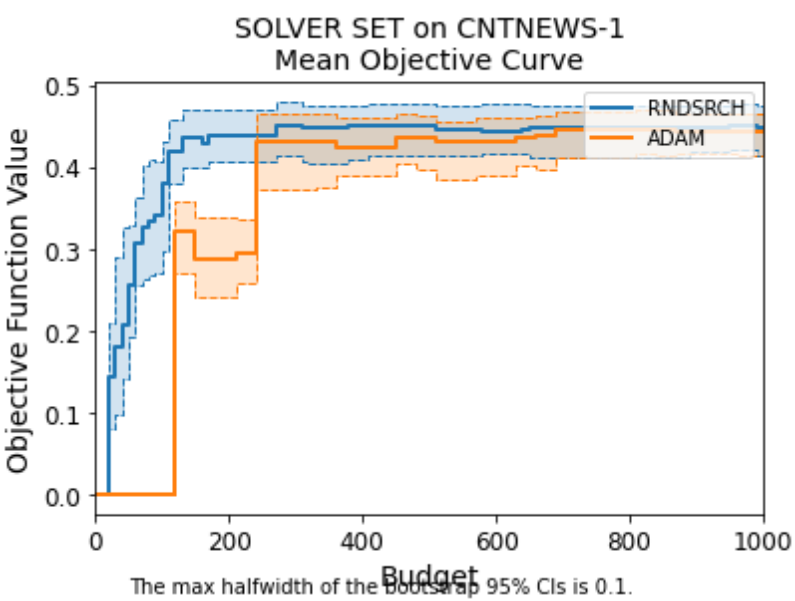
(Normalized) progress curves

- Recall that time is measured as a fraction of the budget T, so $t \in [0, 1]$.
- Rescale the objective function to

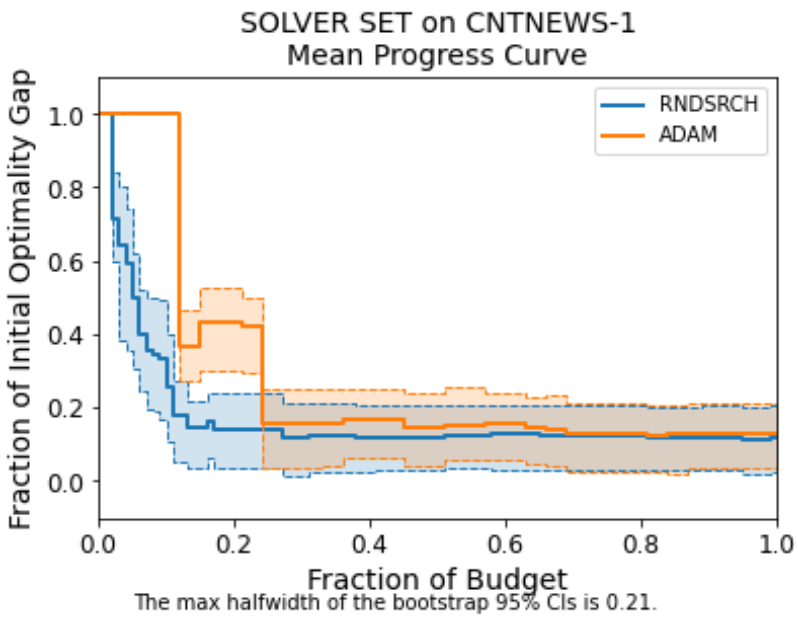
$$\frac{f(X(t)) - f(x^*)}{f(x_0) - f(x^*)}.$$

- $f(x^*)$ = optimal objective function value, part of problem. If unknown, estimated as best solution seen.

Normalized progress curves



(a) unnormalized



(b) normalized

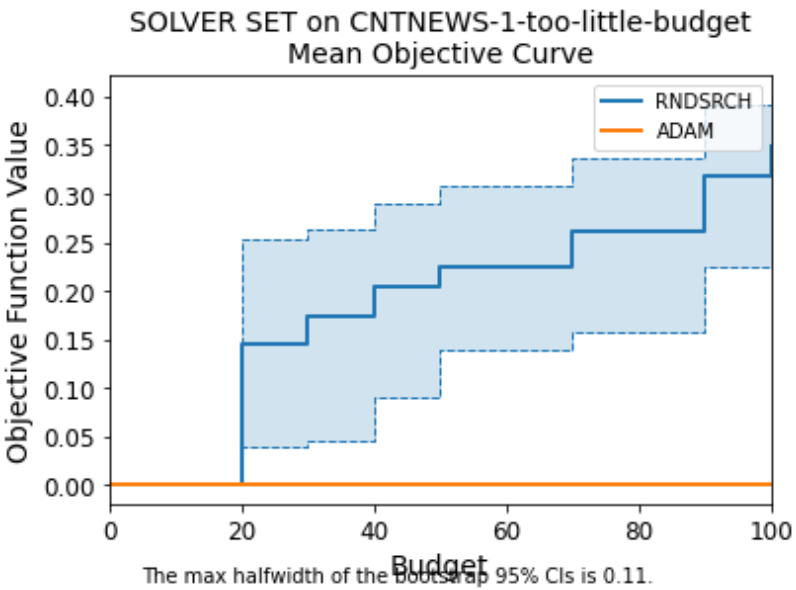
Figure: Mean+CI progress curves.

Your turn

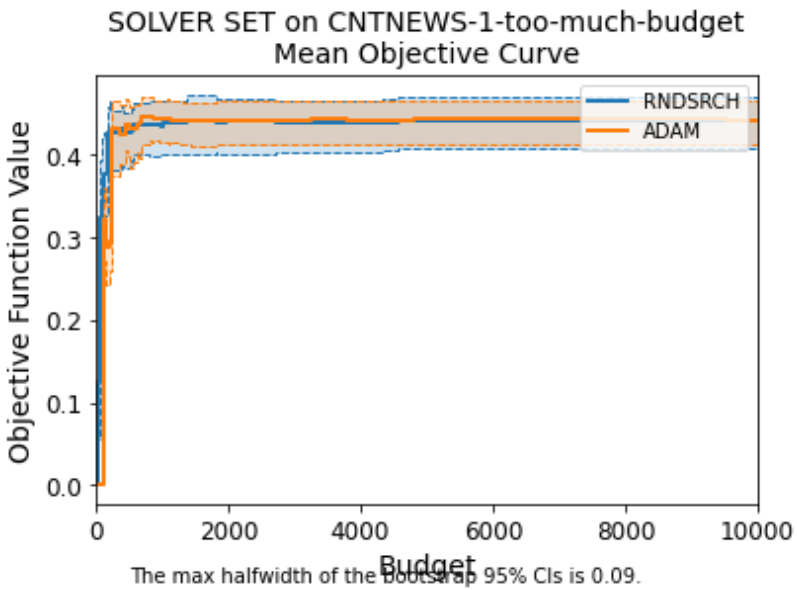
Generate normalized plots

1. From the Plot window, select the problem and two solvers, select the plot type and click "Add"
2. Click "View Plot," which appears below

Bad budgets



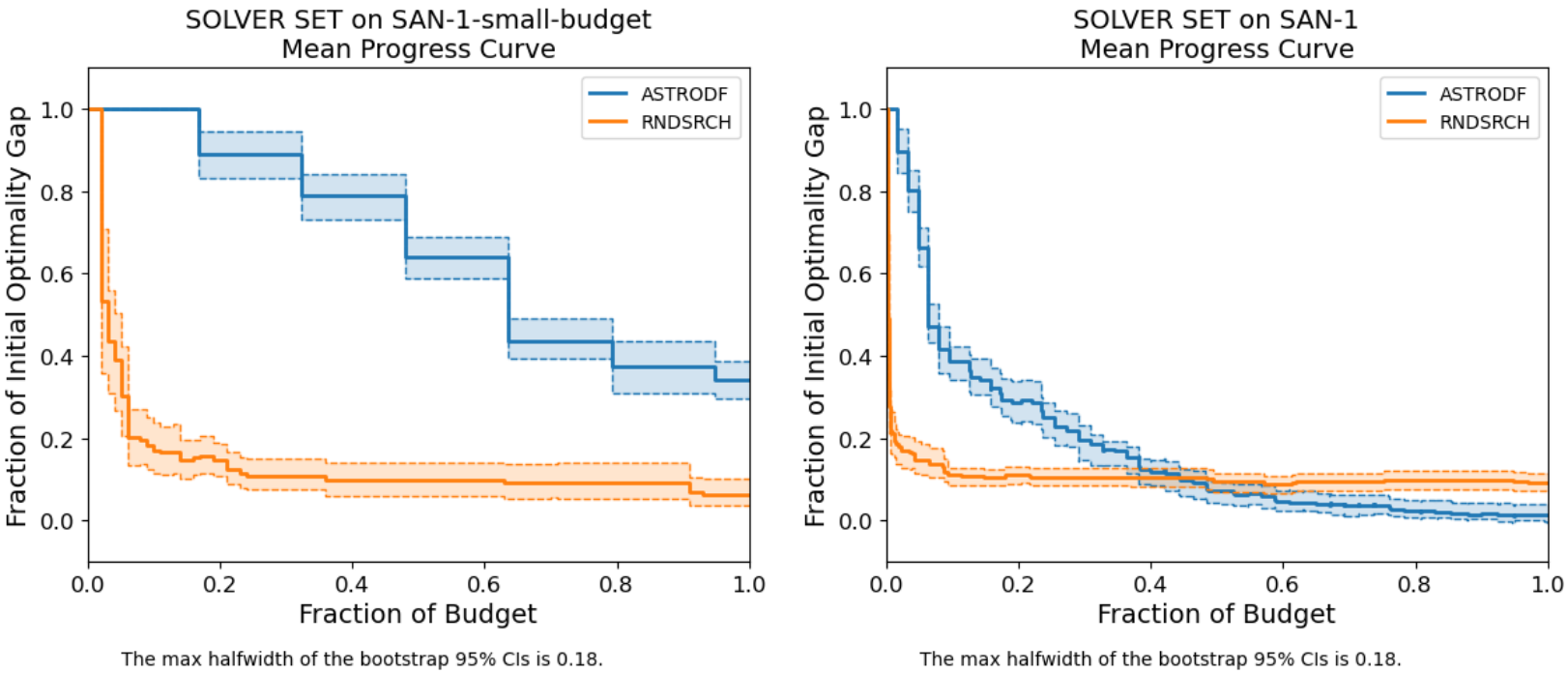
(a) Budget too small



(b) Budget too large

Figure: Unnormalized aggregated progress curves.

Bad budgets (another example)



(a) Budget = 1000

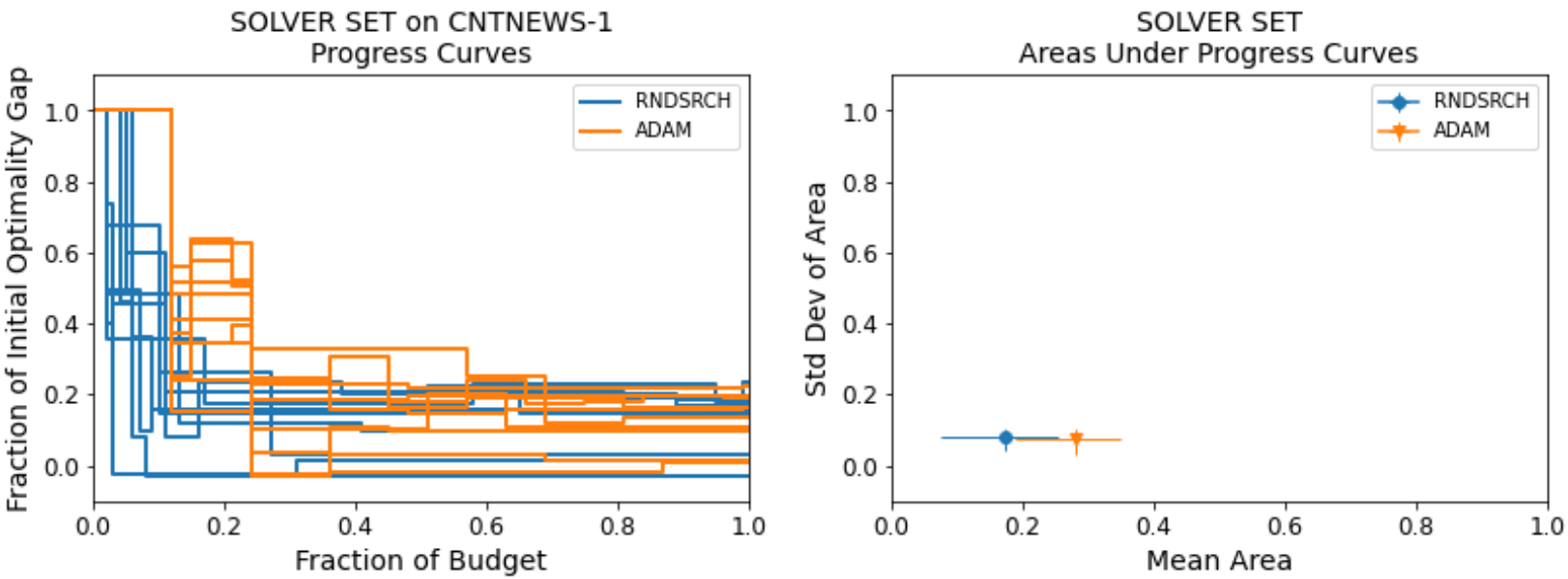
(b) Budget = 10,000

Figure: Normalized aggregated progress curves.

Multiple problems

Area under progress curves and scatter plots

- Summarize performance of many solvers on many problems
- Let A be the (random) area under a normalized progress curve from a single macroreplication. Plot (μ_A, σ_A) and their marginal CIs for each problem and solver



(a) aggregate progress curves

(b) scatter plots

Figure: Area under progress curves.

Area scatter plots

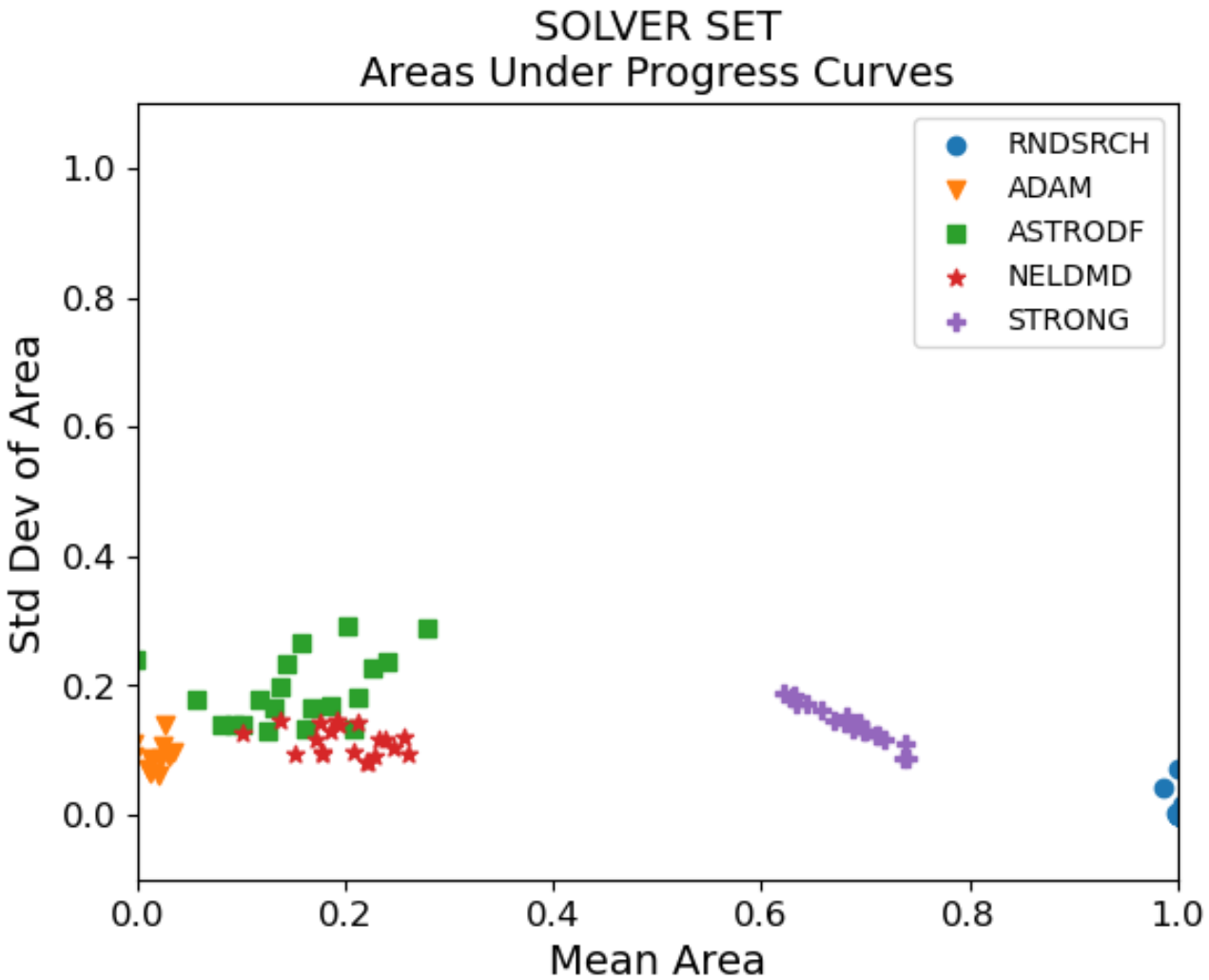


Figure: Comparing 5 solvers on 20 problems.

Solvability profiles

Main purpose is to explore time-dependent performance of multiple solvers on multiple problems. First consider *one* problem and *one* solver

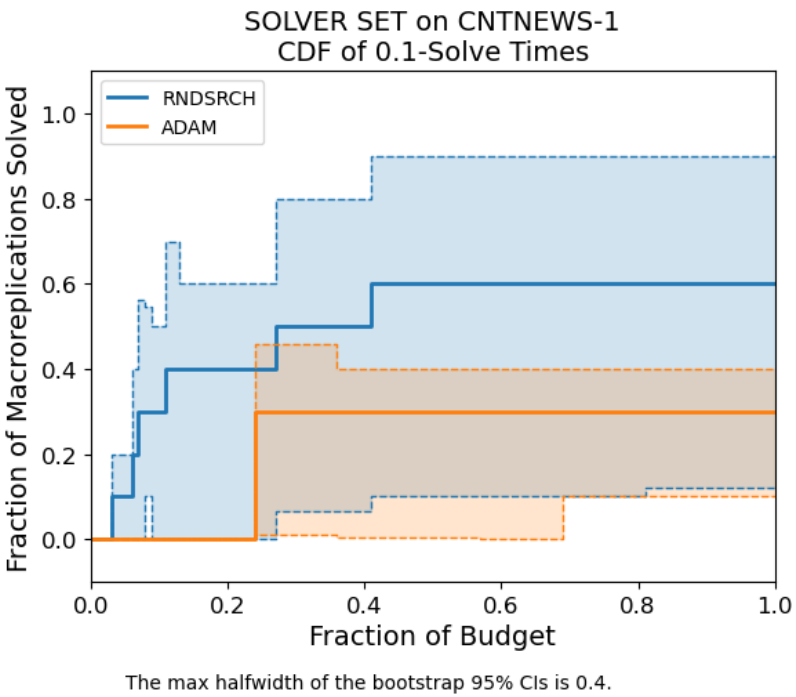
- Look at progress curves a different way. How long to reduce initial optimality gap to 10% of initial value?

$$\tau = \text{budget } t \text{ when first get within 10\%}$$

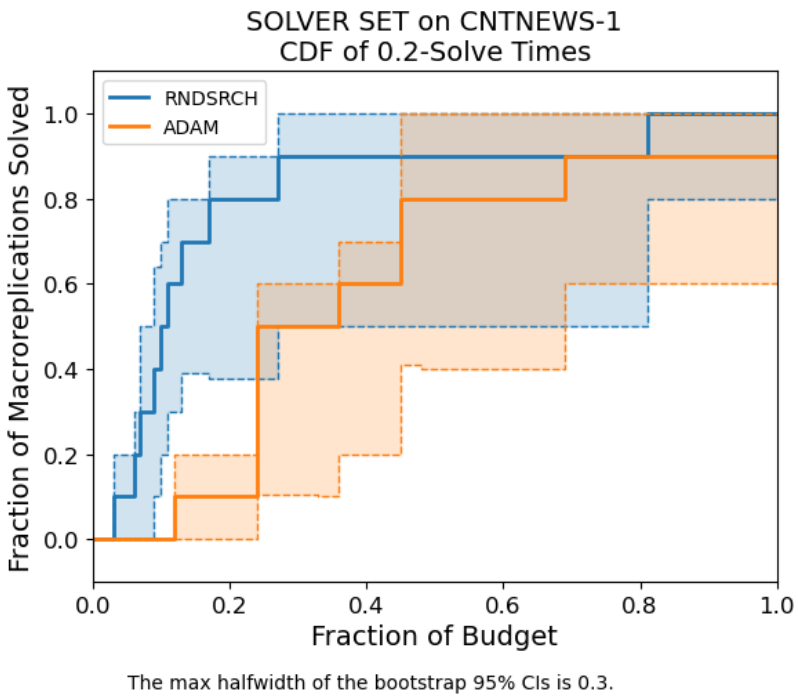
- τ is *random* and can $= \infty$ with positive probability
- We call τ the 10% solve time
- Could look at cdf and/or quantiles of τ ...

Solve time

One problem, Multiple solvers



(a) $\alpha = 0.1$



(b) $\alpha = 0.2$

Figure: α-solve time CDF.

Solvability profiles

Multiple problems

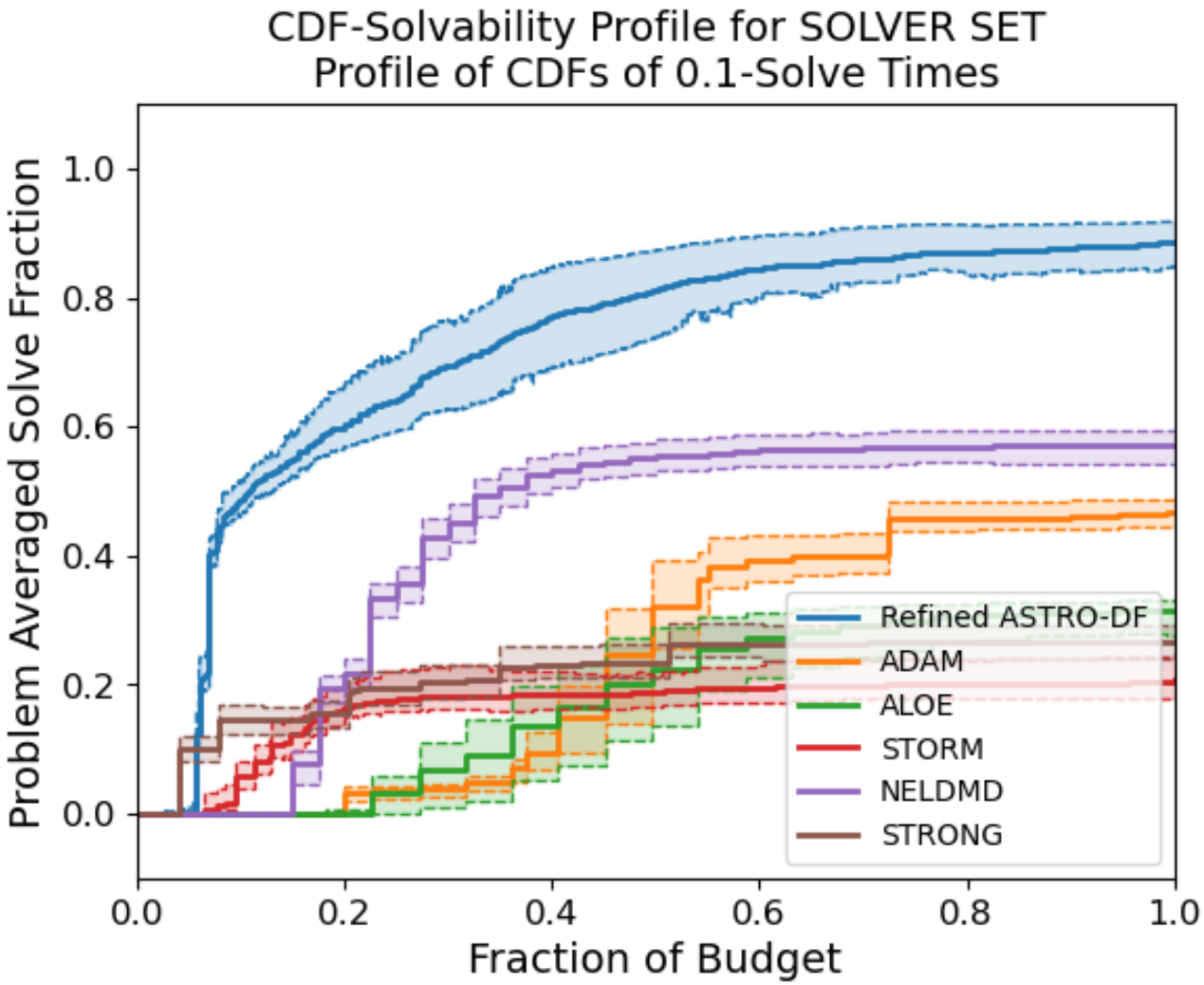
Fix a solver

- Let τ^p be the solve time for problem p
- CDF solvability profile: As a function of t,

$$\rho(t) = \frac{1}{|\mathcal{P}|} \sum_{p \in \mathcal{P}} \Pr(\tau^p \leq t)$$

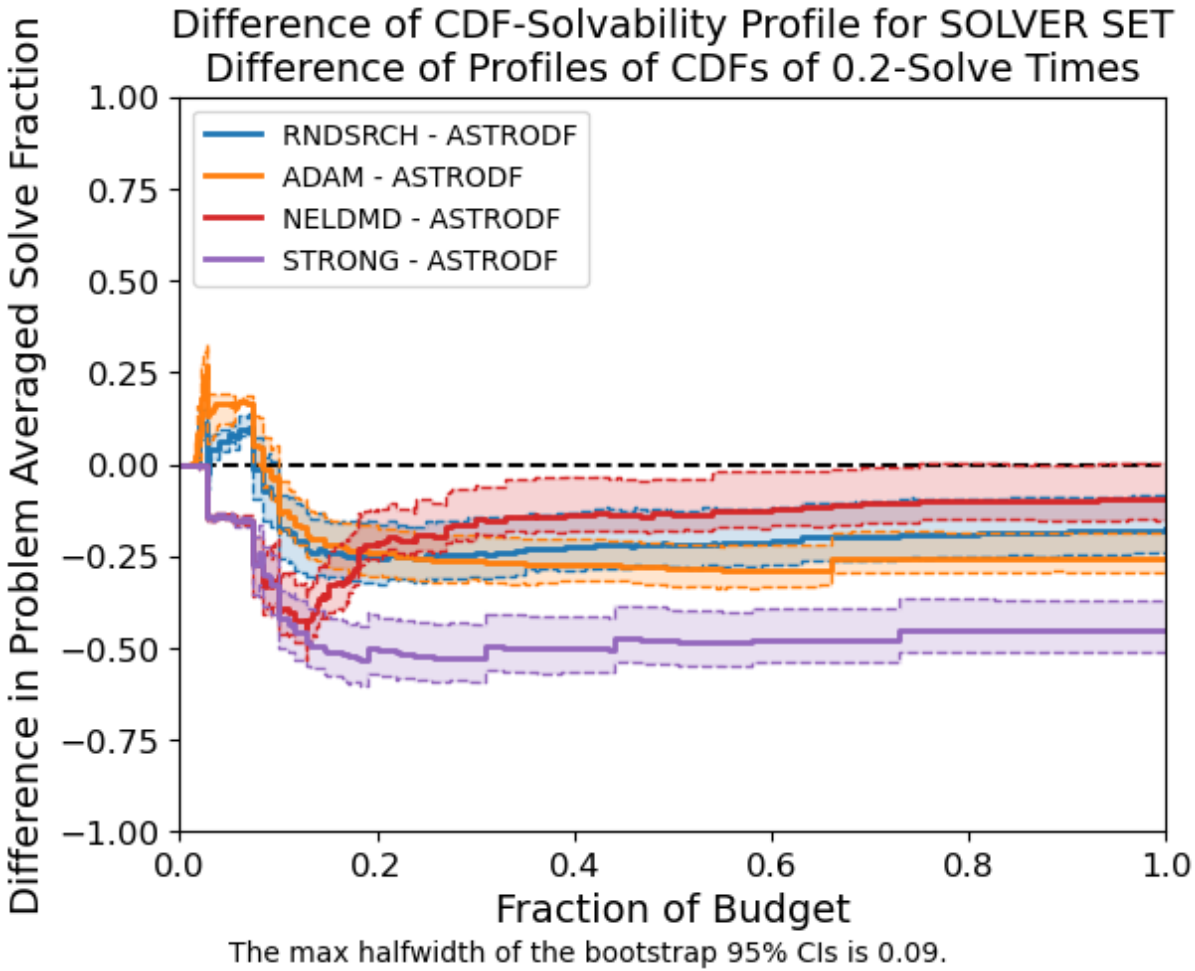
  Average, over problems, of probability solve by time t
- Quantile solvability profiles gives the fraction of problems that are likely (a quantile) solved by time t, as function of t. (Skipping those, today.)

Examples


CDF-Solvability Profile for SOLVER SET
Profile of CDFs of 0.1-Solve Times
Problem Averaged Solve Fraction
Fraction of Budget
0.0
0.2
0.4
0.6
0.8
1.0
Refined ASTRO-DF
ADAM
ALOE
STORM
NELDMD
STRONG

Difference profiles

If you are focused on one solver in particular, compute differences in solvability profiles relative to that solver

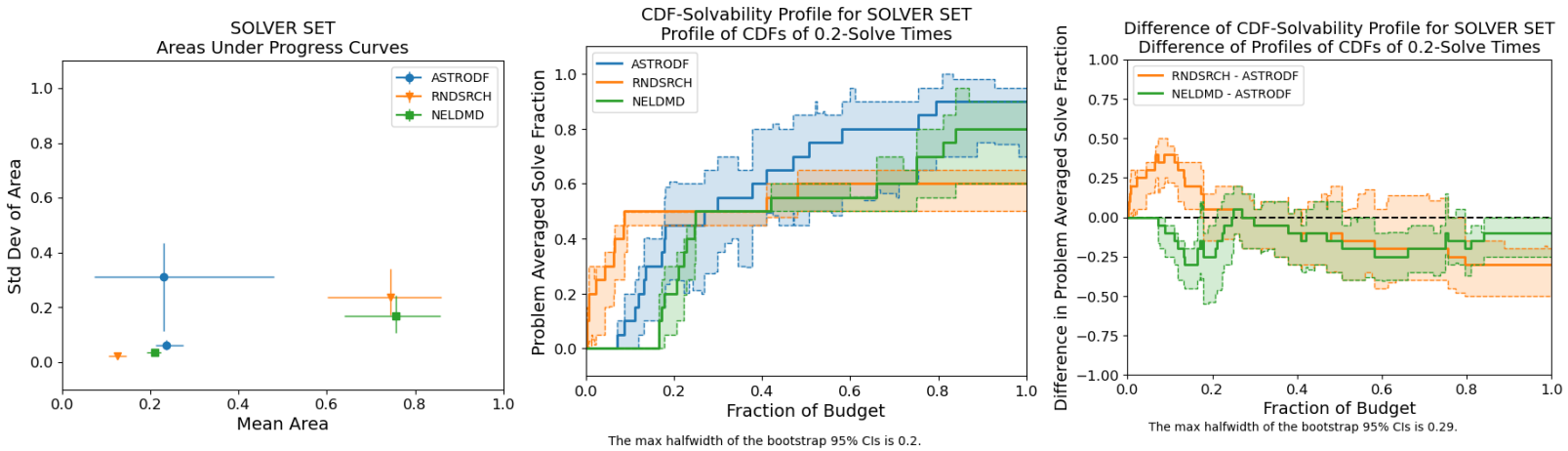


Your turn

1. Create a new problem-solver group using problems
   - Max Revenue for Continuous Iron Ore, and
   - Min Mean Largest Path for Stochastic Activity Network

   and solvers
   - ASTRO-DF,
   - Random Search, and
   - Nelder-Mead.
2. Run and postprocess the results. It'll take a few minutes; you can monitor progress in the Terminal panel.
3. Generate the plots you want.

Two problems, Three solvers



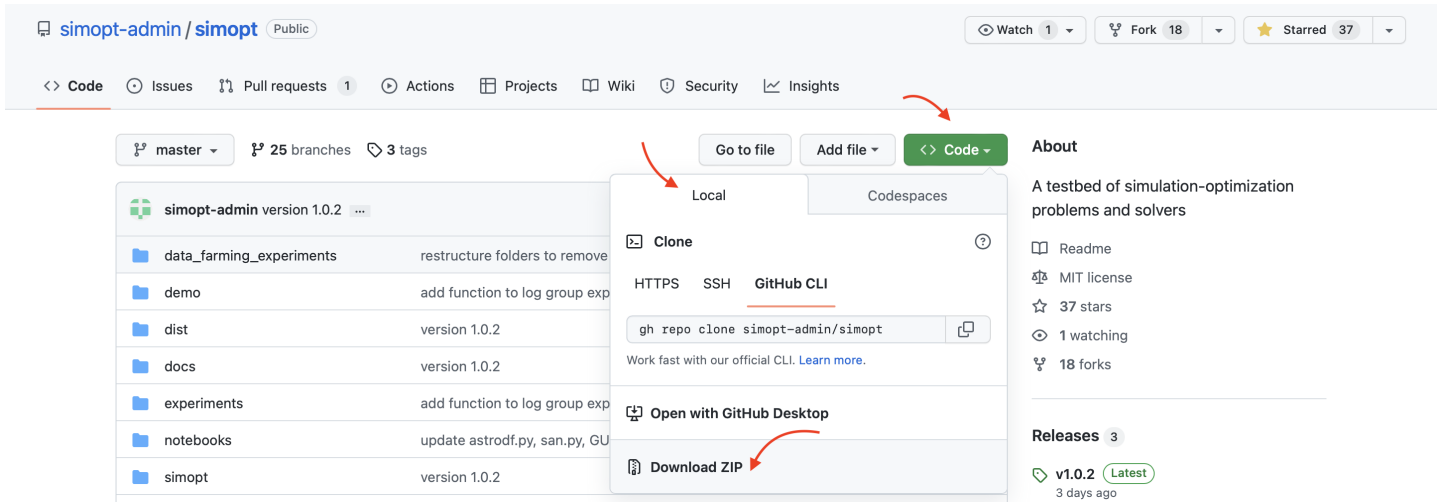
(a) Area scatter plots (b) Solvability profiles (c) Difference profiles

Figure: Comparing 3 solvers on 2 problems.

Writing your own models, problems, and solvers

Reminder (you may have already done this)

1. In your browser, navigate to `https://github.com/simopt-admin/simopt`.
2. Click on "Download ZIP" as shown.



3. Unzip the folder `simopt-master` and open it in VS Code using "File > Open Folder".
4. In the terminal, run
   - `ipython kernel install --user --name=venv`
   - `jupyter notebook workshop/workshop.ipynb`.

Running a data farming experiment

Data farming and SimOpt

Data farming: draw inferences about a stochastic system by "growing" a dataset from an experimental design.

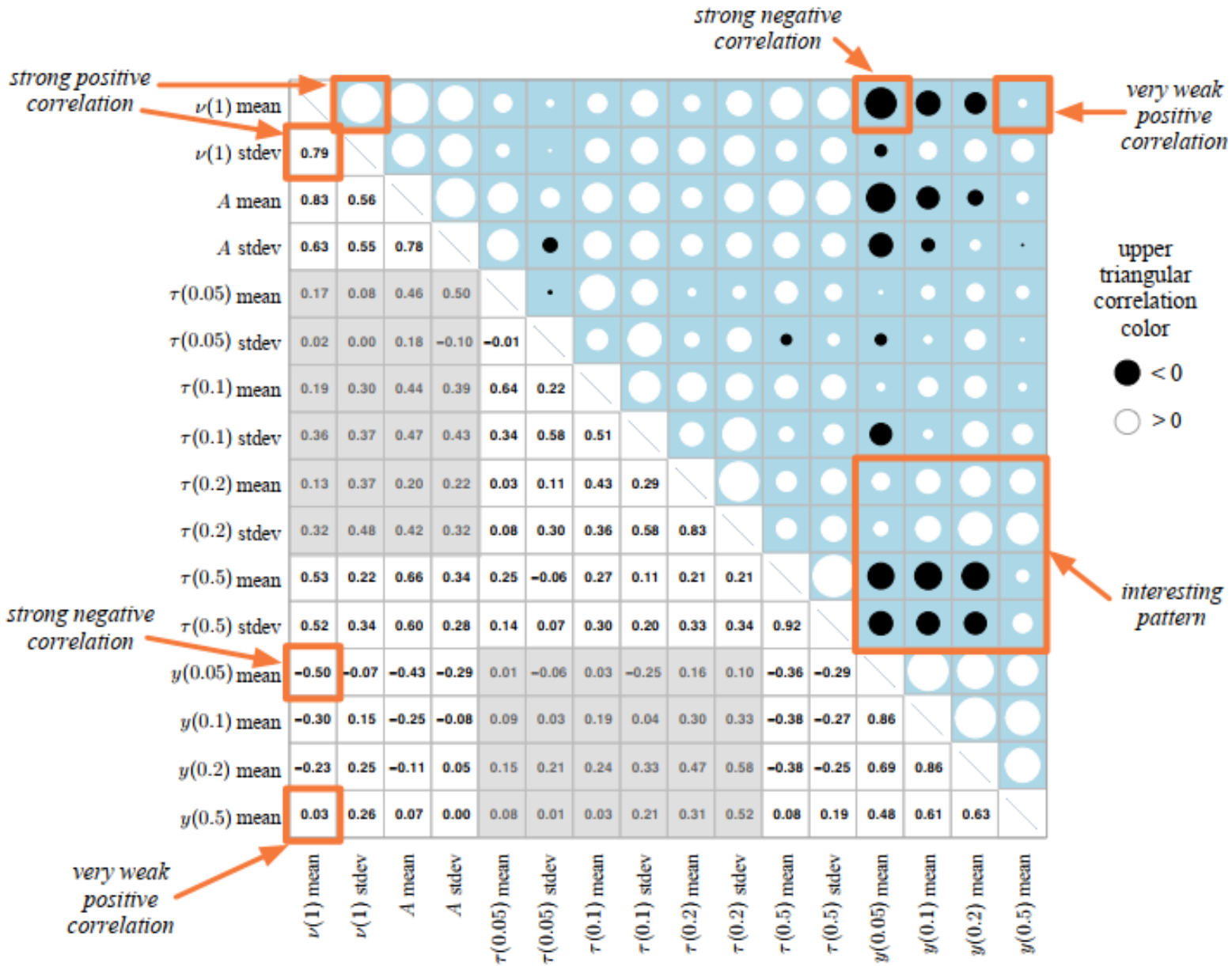
- A data farming experiment in SimOpt has two steps:
  1. Create large designs over problem and/or solver *factors*.
  2. Run experiments on the resulting problem-solver pairs.
- The resulting dataset can be analyzed with statistical software to:
  - use case 1: data farm a problem given a solver to
    - study solver robustness/sensitivity to problem variants.
  - use case 2: data farm a solver given a problem (set) to
    - tune solver hyperparameters for a (class of) problem(s).
- SimOpt 2.0 beta version has a rudimentary version of this data-farming capability (coming up soon!).

An example of data farming a solver

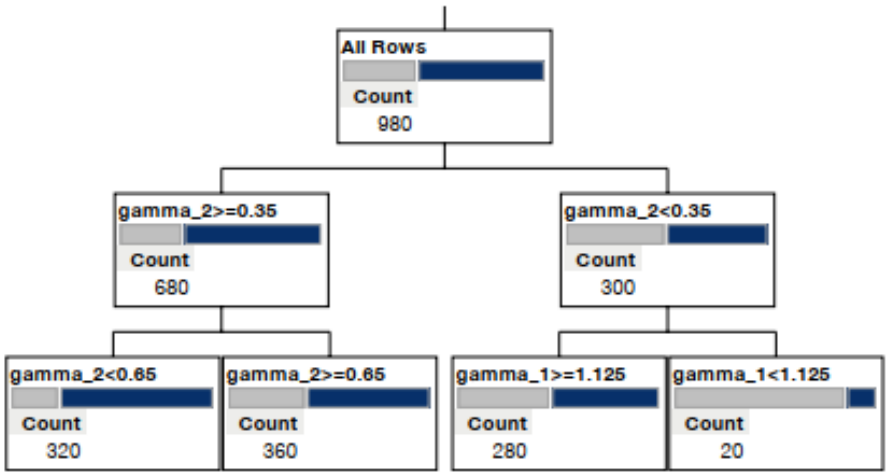
Table: Summary of factor influences on ASTRO-DF performance metrics. Classifications are categorized as very very strong (VVS), very strong (VS), strong (S), medium (M), weak (W) and very weak (VW).
Observation: γ_1 (step size expansion factor) has a strong impact on solving the inventory problem, while it has less impact for the SAN problem.

	(s, S) Inventory Problem			Stochastic Activity Network		
Response	γ_1	γ_2	R^2	γ_1	γ_2	R^2
normalized terminal objective function mean	VVS	VS	0.86	M	VVS	0.78
normalized terminal objective function stdev	VVS	VS	0.77	W	VVS	0.90
area-under-the-progress-curve mean	VVS	S	0.91	VS	VVS	0.93
area-under-the-progress-curve stdev	VVS	W	0.72		VVS	0.95
0.1-solve time mean	VVS	VS	0.91	VS	VVS	0.90
0.1-solve time stdev	VVS	S	0.88	S	VS	0.79
0.5-solve time mean	VVS		0.74	VVS	VS	0.92
0.5-solve time stdev	VVS	W	0.60	VS	VS	0.80

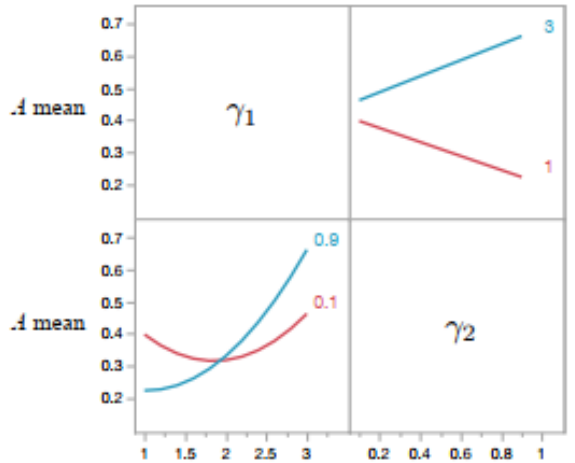
More plots with "farmed" data...



More plots with "farmed" data...



(a) Partial partition tree for a generalist, showing the top three layers of the 10-split partition tree for the 0.1-solvability, $y(0.10)$, for both problems using the raw datafile. The light and dark bars in each leaf indicate the proportions of non-solved and solved runs, respectively.



(b) Interaction profiles for a specialist, based on stepwise regression metamodel for A mean for the SSCONT problem. There is a quadratic effect for γ_1, a linear effect for γ_2, and a strong interaction.

Data farm a simulation model

Even without a problem or a solver, we can use data farming in SimOpt for models to understand how the model outputs change with respect to

1. design variables, for constructing a metamodel, or
2. input parameters, for input uncertainty quantification.

Check out the Data Farming workshop this afternoon to learn more about how to create designs.

Run a data farming experiment on the newsvendor model

- Open the `Continuous Newsvendor_design.csv` file from the data_farming_experiments folder.
- We will run this data farming experiment with GUI:
  1. In terminal, move to the simopt directory, e.g., `cd ./Downloads/simopt-master`
  2. Run `python -m simopt.GUI`
  3. Click "Model Data Farming (beta)".
  4. Click "Load Design CSV" and select the file "Continuous Newsvendor_design.csv" from the "data_farming_experiments" folder.
  5. Choose the number of replications and CRN setting and click "Run All".
  6. Choose the outputs file name and execute.
  7. Open the outputs file.

Wrapping up

Future Plans for SimOpt

- Accelerating code, including parallelization
- Random problem instances
- Extending mrg32k3a
- More data farming capabilities, e.g., hyper-parameter tuning
- Increasing problem/solver diversity, especially simheuristics like genetic algs
- Automatic differentiation
- Trace-driven simulation
- Calibration, empirical risk minimization, distributionally robust optimization

Thanks for attending today!

Please stay involved! Some ideas:

1. Use SimOpt to test/improve your own solver, and share your solver code with us
2. Use SimOpt to tackle your own problem, and share your problem code with us
3. Tackle some of the items on the SimOpt to-do list (see the README on GitHub and please work on your own fork)
4. Suggest improvements, bug fixes, ideas for new problems, ideas for new solvers

References

http://simopt.org

Diagnostic tools for evaluating and comparing simulation-optimization algorithms

David J. Eckman, Shane G. Henderson, and Sara Shashaani

INFORMS Journal on Computing **35** (2) 350–367, 2023.

SimOpt: A testbed for simulation-optimization experiments

David J. Eckman, Shane G. Henderson, and Sara Shashaani

INFORMS Journal on Computing **35** (2) 495–508, 2023.

Data Farming the Parameters of Simulation-Optimization Solvers

Sara Shashaani, David J. Eckman, and Susan M. Sanchez

Under Review. Online: `https://shorturl.at/hkKY6`.