Benchmarking Experiment Planning Algorithms for Autonomous Experimentation

Introduction

Optimization is one of the most important areas in computational sciences. In chemistry and materials science, optimization tasks are abundant and often challenging due to complex multi-dimensional relationships between parameters and output, subjection to noise on the input parameters and output measurements, and expensive-to-evaluate experimentation. Thus, various optimization algorithms have been used extensively among chemists and material scientists for their use in experiment planning. Interfacing these experiment planning algorithms with automated laboratory equipment to practically execute the experimentation, we obtain a laboratory setup capable of autonomous experimentation. [1, 2, 3] Recently, prototypes of autonomous science have designed and optimized advanced materials and functional molecules with increased sample efficiently compared to conventional experimentation. [4, 5, 6, 7, 8, 9]

Despite this progress, the selection of experiment planning algorithms for a novel optimization campaign has largely relied on heuristics and ease-of-use rather than empirical evidence from thorough benchmarks on related campaigns. Furthermore, there are many optimization algorithms available to a researcher, sometimes with various software implementations. Thus, it is unclear which strategy will yield best results for the optimization campaign at hand. To address the need for standardized benchmarks of optimization techniques, the Python package OLYMPUS was reported by The Matter Lab. [10] In the present effort, we attempt benchmark the planners present within OLYMPUS and discuss the behaviour of optimization trials by certain planners on various experimental surfaces.

Planners and Datasets

The `planners` module and `Planner` class in OLYMPUS implements a high-level API to a variety of experiment planning strategies. As of now, this class implements 18 different experiment planning strategies which can be categorized by the optimization approach that they utilize. The five categories considered here are grid-like, Bayesian, evolutionary, gradient-based, and heuristic. We provide a brief description of each approach below.

1. Grid-like strategies: These strategies suggest evaluation of points in the parameter space that are chosen either randomly (`RandomSearch`) or quasi-randomly (`LatinHyperCube` and `Sobol`) or from an evenly spaced grid spanning the parameter space (`Grid`). These strategies do not involve feedback from previous measurements and sample points that are determined beforehand.

2. Bayesian strategies: These strategies optimize the objective function using ML-based surrogate models and acquisition functions that are updated in light of new observations as the experiment proceeds. `Gpyopt` [11] uses a Gaussian process as its surrogate model, `Phoenics` [12] uses Bayesian neural networks and kernel density estimation to construct a weighted kernel average surrogate model and `Hyperopt` [13]uses Parzen estimators.

3. Evolutionary strategies: These planning strategies are inspired by biological processes like swarm movement (`ParticleSwarms`), biological evolution (`Cma` and `DifferentialEvolution`) and genetic processes (`Genetic`).

4. Gradient-based strategies: These algorithms utilize estimates of the derivative of the objective function to recommend subsequent parameter points. This category includes `SteepestDescent, Lbfgs, Slsqp` and `ConjugateGradient`.

5. Heuristic strategies: This category includes `Simplex`, an implementation of the Nelder-Mead algorithm, `BasinHopping`, a planner inspired by the energy landscape of atom clusters and `Snobfit`. [14] These strategies have very different optimization approaches and do not fit into other categories.

However, the planners mentioned below are not included in the benchmark results due to certain technical difficulties. We intend to fix these technical issues in subsequent work.

1. `Slsqp`: This planner does not allow a bound on its parameter space and thus the parameters obtained from this planner are physically impossible to achieve.
2. `Phoenics`: This planner could not be included in the benchmark results due to unavailability of an implementation compatible with Windows OS.
3. `ParticleSwarms`: Current distribution of planner does not sample initial points randomly.

The datasets included in OLYMPUS are experimentally derived and represent various optimization problems from chemistry and material science. The datasets vary in parameter space dimensionality from 3 to 6 and each have a scalar-valued objective value which is to be optimized. OLYMPUS includes the following datasets:

1. `alkox`: This dataset reports the oxidation of benzyl alcohol by a copper radical oxidase. The objective of the dataset is conversion of benzyl alcohol has to be maximized.
2. `photo_pce10`: This dataset reports the amount of photo-degradation of polymer blends of varying proportions of PCE10, P3HT, PCBM and oIDTBR which are used in organic solar cells as OPV donors. This photo-degradation has to be minimized, indicating a photostable blend.

3. `photo_wf3`: This dataset reports the amount of photo-degradation of polymer blends of varying proportions of WF3, P3HT, PCBM and oIDTBR which are used in organic solar cells as OPV donors. This photo-degradation has to be minimized, indicating a photostable blend.
4. `snar`: This dataset reports the e-factor (environmental factor) for nucleophilic aromatic substitution (SnAr reaction) conducted in a flow reactor. E-factor is the ratio of amount of waste produced to the amount of product and it has to be minimized.
5. `benzylation`: This dataset reports the yield of undesired tertiary amine product in an N-benzylation reaction. The yield of the impurity has to be maximized.
6. `colors_n9`: This dataset reports the difference between normalized green-like RGB [0.16 0.56 0.28] and colors obtained by mixing varied amounts of red, blue and green colored dyes. This difference is to be minimized.
7. `colors_bob`: This dataset reports the difference between normalized green-like RGB [0.16 0.56 0.28] and colors obtained by mixing varied amounts of red, orange, yellow, blue and green colored dyes. This difference is to be minimized.
8. `suzuki`: This dataset reports the reaction yield of Palladium-catalysed cross coupling between 2-Bromophenyltetrazole and Aryl boronate. The reaction yield is to be maximized.
9. `fullerenes`: This dataset reports the mole fraction of o-xylenol adducts of Buckminster fullerenes. Since these adducts are the desired product of the reaction, the mole fraction is to be maximized.
10. `hplc`: This dataset reports the peak response area of a high-performance liquid chromatography system with varied parameters. This response area is to be maximized.

The `alkox` dataset, however, could not be included in the benchmark results due to conflicting parameter space and dataset values in the implementation of the dataset. This issue will be resolved in future versions of OLYMPUS.

The experimental surfaces of these datasets were emulated using the `Emulator` class in OLYMPUS. These emulators are constructed using Bayesian Neural Networks (BNNs) that can be used to generate a distribution of the possible output values for sets of input parameters. BNN emulators can then be used to query any point within the experiment parameter space, providing a virtual measurement. This probabilistic approach towards the calculation of target values is responsible for noise in the output values which is intended to model measurement error frequently encountered during practical experimentation.

Simulations

In order to benchmark the optimization performance of each planner across multiple datasets, we required repeated optimization executions to average out random initializations and stochastic elements of the strategies. Thus, we ran each planner on each dataset 100 independent times for a duration of 100 iterations. At each iteration, planners were asked to suggest a set of parameters. This set of queried parameters was then passed to the BNN emulator, which produced a virtual measurement. The measurement was then supplied to the planner to update its

internal state. This “ask-tell” cycle was then repeated until the predefined budget of 100 measurements was reached.

After 100 such iterations, one independent run was said to be completed. The `campaign.params` and `campaign.values` objects are stored after each independent run. These objects contain the list of parameters obtained from the planners and output values from the emulator respectively. Next, the planner was re-instantiated, and the subsequent run commenced. The following code, for example, was used for running planner `RandomSearch` on dataset `benzylation`:

```
from olympus import Dataset, Planner, Emulator, Campaign

#loading an olympus dataset
dataset = Dataset(kind='benzylation')

#loading the corresponding emulator
emulator = Emulator(model='BayesNeuralNet', dataset='benzylation')
results = {}
for i in range(100):

    results[f'Run {i+1}'] = {}
    #Campaign objects are used to save information of the optimization task
    campaign = Campaign()

    #loading a planner from olympus
    planner = Planner(kind='RandomSearch')

    for iteration in range(100):

        #Input the optimization history to the planner
        planner.tell(campaign.observations)

        #Ask the planner for a new set of parameters
        params = planner.ask()

        #Query the new parameters on the emulators
        value = emulator.run(params.to_array(), return_paramvector=False)

        #Add the new parameters and value to optimization history
        campaign.add_observation(params, value)

    #saving the optimization history in a dict variable
    results[f'Run {i+1}']['Params'] = campaign.params
    results[f'Run {i+1}']['Values'] = campaign.values
```

Metrics

Before the calculation of metrics, the extrema of the experimental surfaces were estimated. For some of the datasets like `suzuki`, the theoretical extrema values were known and chosen (yield of reaction lies in [0, 100]). The others were estimated by Sobol sampling. Sobol sequences are low discrepancy quasi-random sequences frequently used as techniques for random sampling. A sobol sequence of 10^6 points is dispersed throughout the experiment parameter space and these points are queried for measurements using the emulator. The resulting extreme values (minima and maxima) are then used as theoretical optima of the surfaces, making computation of the following optimization performance metrics possible.

Simple Regret at *x* iterations (R_x): This metric is simply the difference between the cumulative best value found by the planners after *x* iterations and the theoretical optimums of the emulated surface. This metric can be used to compare the planners on the basis of best values (closest to the theoretical optimum) found after a certain number of iterations.

$$R_x = |(best\ value\ after\ x\ iterations) - (dataset\ optimum)|\ .$$

For the simulations described above, simple regret was calculated after 25, 50 and 100 iterations and averaged over all 100 independent runs. We also discuss percentage regret values which are used to normalize the values of this metric and compare R_x values of planners.

Cumulative Regret (R_C): This metric calculates the total regret incurred over the multiple runs of the planner. At every iteration, the difference between the best value and the theoretical optimum is calculated and added to the cumulative regret value at that iteration.

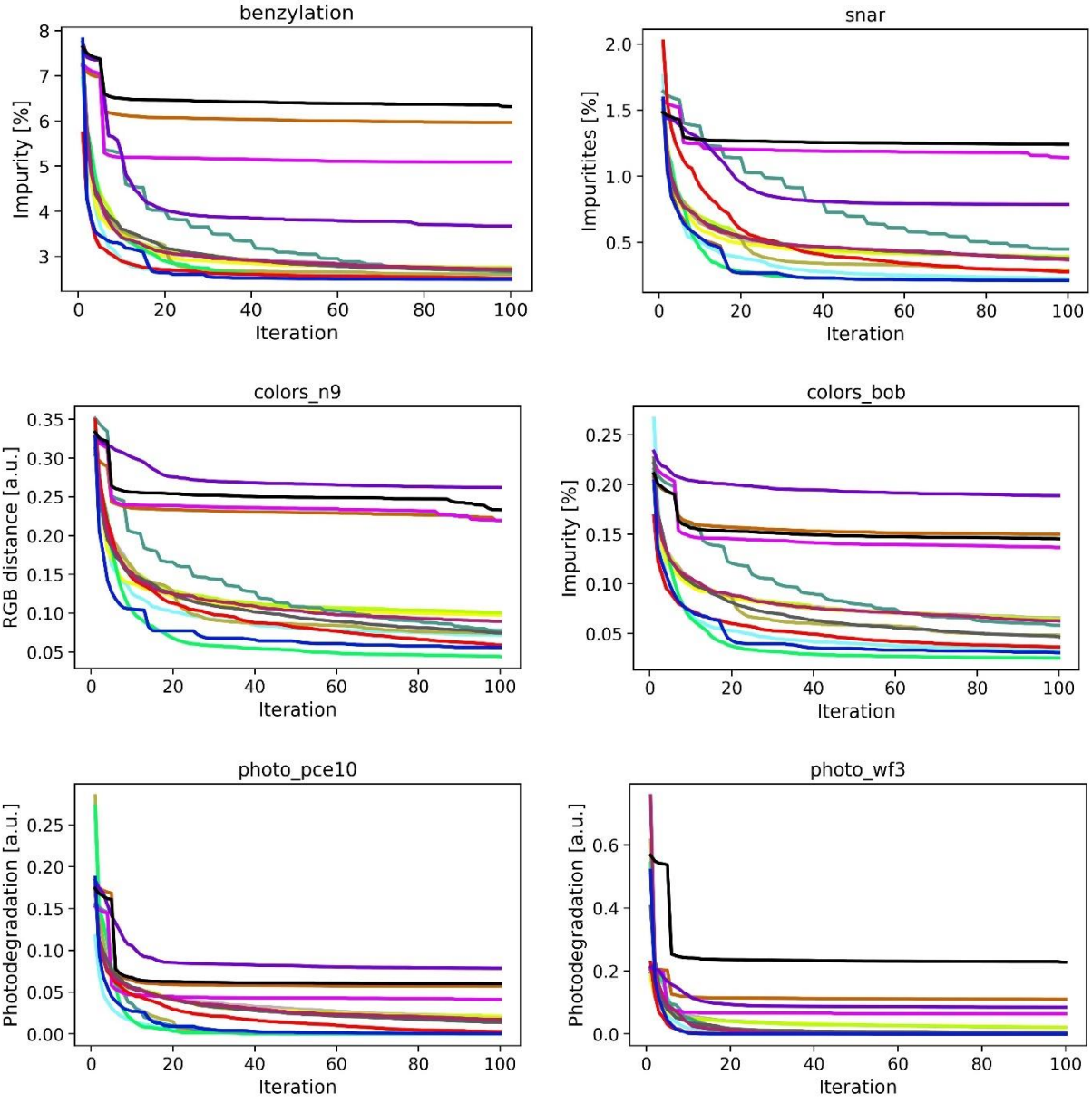
$$R_c = \sum_{x=1}^{100} R_x\ .$$

For the simulations previously described, this is done for 100 iterations and averaged over 100 independent runs.

Results

The results of the simulations were used to benchmark the planners on their optimization capability. The metrics mentioned above were calculated for the results from the simulations and the values obtained were evaluated to identify trends in the performances of the planners. Figure 1 shows average optimization traces of the planners in optimizing the experimental surfaces.

We start our discussion with the simple regret values calculated for each planner at different numbers of iterations. This metric helps evaluate the planners based on best values that it found. The lowest possible value of regret is obviously 0 (meaning the planner successfully found the global minima of the surface) and the highest possible value is the difference between the extremes of the experiment surface. These bounds are used only to infer relative performance of the planners.



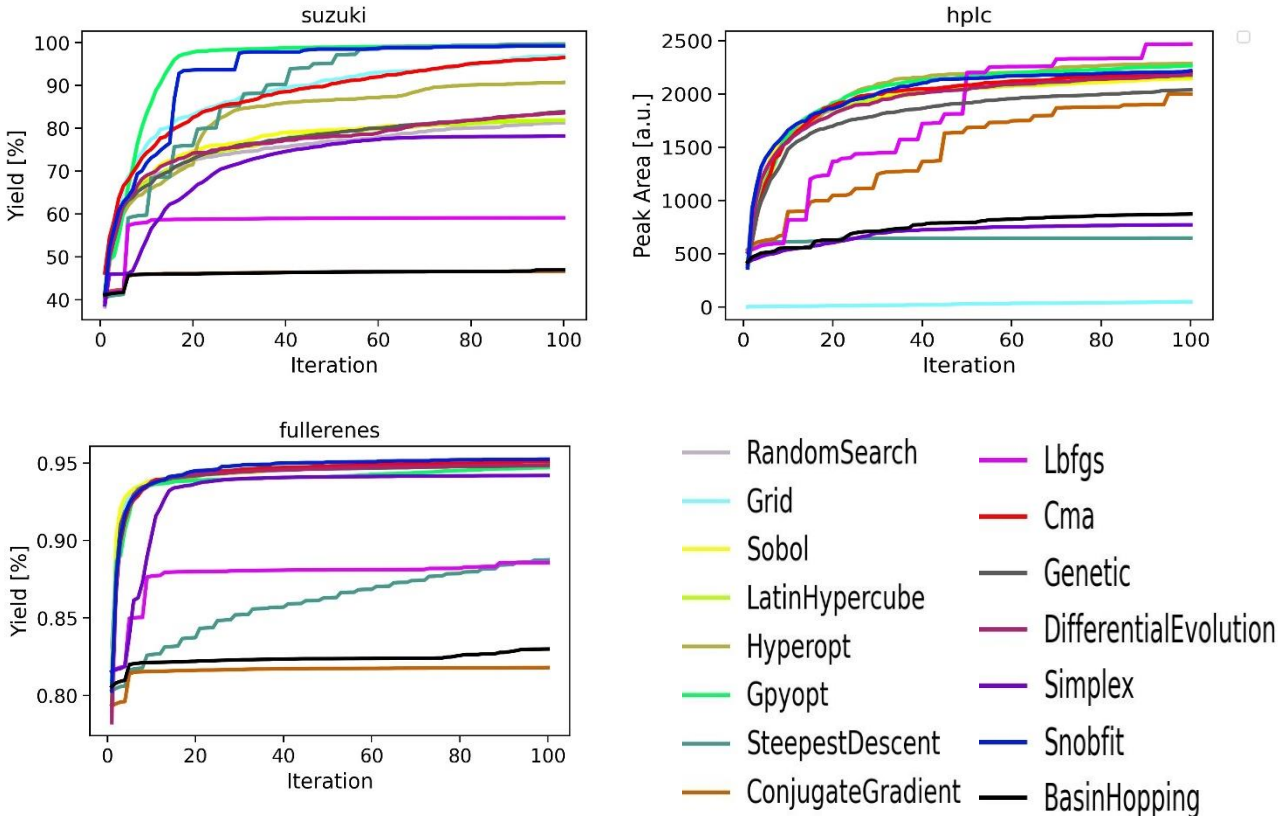


Figure 1: Optimization traces for each planner studied on each emulated dataset. Traces show the most optimal target values identified by each planner (vertical axis) at each iteration (horizontal axis). The traces show average performance over the 100 independently seeded runs.

Regret at 25 iterations (R_{25}):

The regret values are evaluated relatively early in the optimization runs to benchmark the planner performance for instances where experimentation is very expensive and the optimization campaign is allotted a small budget. Table 1 gives the relative rankings of the planners based on R_{25} values.

Since this regret only involves the first 25% of the optimization runs, the discrepancy in range of regret values over different datasets is significant. For example, the difference between the smallest and largest regret value is approximately 65% on dataset `suzuki` and 32% on dataset `colors_n9`. But for datasets `photo_pce10` and `photo_wf3` this difference is approximately 5% and 3% respectively. These observations are consistent with Figure 1.

Some discrepancy is also observed in the relative rankings of the planners. However, even this early in the optimization, the relative performance of the planners is similar to what is observed in later stages. Therefore, here we mainly mention the trends that help identify the extreme ends

of the planner spectrum (that is, the planners with the worst and best regret values). A more detailed analysis of these performance will be near the end of the optimization runs.

For most of the datasets, `Gpyopt, Grid` and `Snobfit` consistently have the highest relative rankings. Comparison between planners of the same category is also of interest to us here. Planner `Hyperopt` shows regret values very similar to `Gpyopt` on most datasets except for datasets `suzuki` and `colors_bob` (where regret values of `Hyperopt` are 10 times and 6 times respectively that of `Gpyopt`). The Grid-based planners `RandomSearch, LatinHypercube` and `Sobol` show mediocre but identical R_{25} values and relative rankings. On the other hand, planners `Lbfgs, BasinHopping, Simplex` and `ConjugateGradient` show very poor performance over most of the datasets.

Table 1 gives the rankings of all the planners on the datasets and is color coded to give an idea of the performance of the planner over multiple datasets.

Regret at 50 iterations (R_{50}):

This time the regret values are evaluated exactly halfway through the optimization runs. `Snobfit, Gpyopt` and `Grid` continue to have very low regret values for most datasets. On datasets other than `photo_pce10` and `photo_wf3`, the regret values of planner `Hyperopt` however, are significantly larger than that of `Gpyopt`. The Grid-based planners continue to show similar mediocre performance throughout the datasets.

`BasinHopping, ConjugateGradient, Lbfgs` and `Simplex` continue to show extremely poor performance compared to other planners. Planner `SteepestDescent` shows extremely inconsistent performance over different datasets. It shows relatively low regret values for `photo_pce10` and `photo_wf3` datasets and is among the worst for all other datasets.

At this halfway point we notice some regret values that are inconsistent with our statements before. Two examples are from the dataset `hplc`. Planner `Lbfgs` (which shows consistently poor performance on most datasets) has the smallest regret value. Planner `Grid` (having very low regret values on most datasets) has the highest regret value on this dataset. Another example is the mediocre performance of planner `Gpyopt` on dataset `fullerenes` and very high performance of `ConjugateGradient` on dataset `snar`.

Regret at 100 iterations (R_{100}):

It may be noted that for some datasets, absolute regret values of most planners are approximately the same at the end of 100 iterations. For the dataset `benzylation`, 12 out of 16 planners have regret values within 1% of each other. For datasets `photo_pce10` and `photo_wf3`, this is true for 12 and 8 out of 16 planners respectively. Which is why the comparison of the planners on this metric is done mostly relative to each other.

Planners `Gpyopt, Grid, Snobfit` and `Cma` consistently perform better than most of the planners over all datasets. For the `photo_pce10` and `photo_wf3` datasets, planner `Hyperopt` is seen to be on par with `Gpyopt` while for other datasets, it has regret values slightly higher than the other three.

The performance of the planner `SteepestDescent` is seen to be very inconsistent here: having best regret values for dataset `suzuki`, worst for `fullerenes` and `snar` and mediocre performance on `benzylation` dataset.

Planners `DifferentialEvolution, Genetic, RandomSearch, Sobol` and `LatinHypercube` show mediocre performance for most datasets. Also, regret values of planners `RandomSearch, Sobol` and `LatinHypercube` lie very close to each other for all datasets - less than 0.5% difference is seen.

The planners `ConjugateGradient, Simplex, Lbfgs` and `BasinHopping` perform consistently the worst of all planners. For some datasets, these planners have regret values over three times that of other planners.

The rankings of the planners based on R_{100} can be found in Table 3.

Cumulative Regret (R_C):

This metric is calculated as the sum of the regret values over all iterations of the planner and thus gives an overall analysis of the optimization trial by the planner. And obviously, the trends seen in the R_c values reflect the exact trends in the various regret values we have already shown.

Planners `Gpyopt, Snobfit, Cma` and `Grid` show the lowest R_c values for most of the datasets which is consistent with the R_{25}, R_{50} and R_{100} values we discussed earlier. On the worse side of planner spectrum, planners `ConjugateGradient, Simplex, Lbfgs` and `BasinHopping` are seen to have higher R_c values compared to most other planners. All other planners show mediocre R_c values as suggested above.

It can also be noted that planners `Gpyopt` and `Hyperopt` show very different performance despite belonging to the same category of planners. However, the Grid-based planners (`RandomSearch, LatinHypercube` and `Sobol`) continue to have very similar R_c values.

The rankings of the planners can be found in table 4.

Planners \ Datasets	benzylation	hplc	snar	colors_n9	colors_bob	suzuki	fullerenes	photo_pce10	photo_wf3
RandomSearch	9	6	7	9	10	11	7	11	11
Grid	2	15	4	3	3	3	2	3	3
Sobol	6	5	6	8	7	7	5	9	10
LatinHypercube	8	7	9	10	9	10	9	7	9
Hyperopt	4	1	5	4	5	5	4	2	6
Gpyopt	5	2	1	1	1	1	10	1	4
Snobfit	1	4	3	2	2	2	1	5	1
Genetic	10	9	10	6	6	9	8	8	8
Cma	3	3	11	5	4	4	3	6	2
BasinHopping	15	12	15	14	13	15	14	14	15
SteepestDescent	11	14	13	11	11	3	13	4	5
ConjugateGradient	14	11	2	12	14	14	15	13	14
Lbfgs	13	10	14	13	12	13	12	12	12
Simplex	12	13	12	15	15	12	11	15	13
DifferentialEvolution	7	8	8	9	8	8	6	10	7

Table 1: Rankings of the planners based on R_{25} values for each dataset. Planners with ranks 1-4 are colored in green, those with rank 5-8 are in blue, those with ranks 9-12 are in yellow and ranks 13-15 in red.

Planners \ Datasets	benzylation	hplc	snar	colors_n9	colors_bob	suzuki	fullerenes	photo_pce10	photo_wf3
RandomSearch	10	6	8	9	10	11	9	11	9
Grid	3	15	4	3	3	4	2	3	7
Sobol	6	7	7	8	7	7	7	10	11
LatinHypercube	8	8	9	10	8	8	5	9	10
Hyperopt	5	2	5	5	5	6	4	2	4
Gpyopt	2	3	3	1	1	1	10	1	2
Snobfit	1	4	1	2	2	2	1	5	1
Genetic	7	10	10	6	6	9	8	7	8
Cma	4	5	6	4	4	5	3	6	5
BasinHopping	15	12	15	14	13	15	14	14	15
SteepestDescent	11	14	12	11	11	3	13	4	6
ConjugateGradient	14	11	2	12	14	14	15	13	14
Lbfgs	13	1	14	13	12	13	12	12	12
Simplex	12	13	13	15	15	12	11	15	13
DifferentialEvolution	9	9	11	7	9	10	6	8	3

Table 2: Rankings of the planners based on R_{50} values for each dataset. Planners with ranks 1-4 are colored in green, those with rank 5-8 are in blue, those with ranks 9-12 are in yellow and ranks 13-15 in red.

Planners \ Datasets	benzylation	hplc	snar	colors_n9	colors_bob	suzuki	fullerenes	photo_pce10	photo_wf3
RandomSearch	10	8	9	10	11	11	9	11	10
Grid	4	15	4	4	3	4	2	3	7
Sobol	8	9	11	9	10	9	8	10	11
LatinHypercube	11	7	10	11	9	10	6	9	9
Hyperopt	5	2	6	5	6	6	4	1	1
Gpyopt	2	3	2	1	1	2	10	1	1
Snobfit	1	4	1	2	2	3	1	5	1
Genetic	7	10	8	6	5	7	7	7	8
Cma	3	5	5	3	4	5	3	6	1
BasinHopping	15	12	15	14	13	14	14	14	15
SteepestDescent	6	14	12	7	7	1	12	4	6
ConjugateGradient	14	11	3	12	14	15	15	13	14
Lbfgs	13	1	14	13	12	13	13	12	12
Simplex	12	13	13	15	15	12	11	15	13
DifferentialEvolution	9	6	7	8	8	8	5	8	1

Table 3: Rankings of the planners based on R_{100} values for each dataset. Planners with ranks 1-4 are colored in green, those with rank 5-8 are in blue, those with ranks 9-12 are in yellow and ranks 13-15 in red.

Planners \ Datasets	benzylation	hplc	snar	colors_n9	colors_bob	suzuki	fullerenes	photo_pce10	photo_wf3
RandomSearch	9	5	7	9	10	11	7	11	11
Grid	3	15	4	3	3	4	2	1	2
Sobol	6	6	6	8	7	7	5	10	9
LatinHypercube	10	8	11	10	9	10	6	9	10
Hyperopt	5	1	5	5	5	6	4	4	6
Gpyopt	4	2	2	1	1	1	10	3	5
Snobfit	2	3	1	2	2	2	1	2	3
Genetic	8	9	9	6	6	8	9	7	8
Cma	1	4	8	4	4	5	3	6	1
BasinHopping	15	12	15	14	13	15	14	14	15
SteepestDescent	11	14	12	11	11	3	13	5	4
ConjugateGradient	14	11	3	12	14	14	15	13	14
Lbfgs	13	10	14	13	12	13	12	12	12
Simplex	12	13	13	15	15	12	11	15	13
DifferentialEvolution	7	7	10	7	8	9	8	8	7

Table 4: Rankings of the planners based on R_C values for each dataset. Planners with ranks 1-4 are colored in green, those with rank 5-8 are in blue, those with ranks 9-12 are in yellow and ranks 13-15 in red.

Conclusion

It can be clearly stated now that there exists a significant difference in the optimization capabilities of the planners benchmarked. Over most datasets, planners `Gpyopt, Snobfit, Cma` and `Grid` consistently exhibit better optimization performance than the other planners, and are therefore highly recommended for usage in optimization problems where the objective is to find optimize and expensive-to-evaluate, black-box, noisy experimental setup. For the same optimization task, planners categories such as Gradient-based planners, `BasinHopping` and Nelder-Mead `Simplex` algorithm should be avoided. The remaining planners are found to have mediocre performances. However, the difference in the optimization performance among the planners may not be significant depending entirely on the experimental surface. Some correlation in the performance of planners of the same category is noted as well. Bayesian optimization strategies show exceptional performance on most surfaces. Grid-based strategies (except planner `Grid`) show mediocre but very similar metric values and optimization curves. Most Gradient-based planners are seen to show extremely poor performance for most datasets.

Acknowledgements

I would like to begin by thanking my research advisors Riley Hickman, Matteo Aldeghi and Prof. Alán Aspuru-Guzik for all the support and guidance before and throughout the project. I also express my gratitude towards the Laidlaw Scholars Foundation, funded by Lord Laidlaw, welcoming me into this wonderful community of scholars and future leaders.

References

1. Häse, F., Roch, L. M., & Aspuru-Guzik, A. (2019, March 6). *Next-generation experimentation with self-driving laboratories*. Trends in Chemistry. Retrieved September 24, 2021, from https://www.sciencedirect.com/science/article/abs/pii/S258959741930019X?via%3Dihub.
2. Langner, S., Häse, F., Perea, J. D., Stubhan, T., Hauch, J., Roch, L. M., Heumueller, T., Aspuru-Guzik, A., & Brabec, C. J. (2020, February 12). *Beyond ternary OPV: HIGH-THROUGHPUT experimentation AND Self-Driving laboratories Optimize Multicomponent Systems*. Wiley Online Library. Retrieved September 24, 2021, from https://onlinelibrary.wiley.com/doi/pdf/10.1002/adma.201907801.
3. Roch, L. M., Häse, F., Kreisbeck, C., Tamayo-Mendoza, T., Yunker, L. P. E., Hein, J. E., & Aspuru-Guzik, A. (n.d.). *Chemos: An orchestration software to democratize*

autonomous discovery. PLOS ONE. Retrieved September 24, 2021, from https://journals.plos.org/plosone/article?id=10.1371%2Fjournal.pone.0229862.

4. Roch, L. M., Häse, F., Kreisbeck, C., Tamayo-Mendoza, T., Yunker, L. P. E., Hein, J. E., & Aspuru-Guzik, A. (n.d.). *Chemos: An orchestration software to democratize autonomous discovery*. PLOS ONE. Retrieved September 24, 2021, from https://journals.plos.org/plosone/article?id=10.1371%2Fjournal.pone.0229862.
5. Roch, L. M., Häse, F., Kreisbeck, C., Tamayo-Mendoza, T., Yunker, L. P. E., Hein, J. E., & Aspuru-Guzik, A. (n.d.). *Chemos: An orchestration software to democratize autonomous discovery*. PLOS ONE. Retrieved September 24, 2021, from https://journals.plos.org/plosone/article?id=10.1371%2Fjournal.pone.0229862.
6. Roch, L. M., Häse, F., Kreisbeck, C., Tamayo-Mendoza, T., Yunker, L. P. E., Hein, J. E., & Aspuru-Guzik, A. (n.d.). *Chemos: An orchestration software to democratize autonomous discovery*. PLOS ONE. Retrieved September 24, 2021, from https://journals.plos.org/plosone/article?id=10.1371%2Fjournal.pone.0229862.
7. Nikolaev, P., Hooper, D., Webber, F., Rao, R., Decker, K., Krein, M., Poleski, J., Barto, R. and Maruyama, B. *Autonomy in materials research: a case study in carbon nanotube growth*. Retrieved September 24, 2021, from https://www.nature.com/articles/npjcompumats201631
8. Nikolaev, P., Hooper, D., Webber, F., Rao, R., Decker, K., Krein, M., Poleski, J., Barto, R. and Maruyama, B. *Autonomy in materials research: a case study in carbon nanotube growth*. Retrieved September 24, 2021, from https://www.science.org/doi/10.1126/sciadv.aaz8867
9. Zhi Li, Mansoor Ani Najeeb, Liana Alves, Alyssa Z. Sherman, Venkateswaran Shekar, Peter Cruz Parrilla *Robot-Accelerated Pervoskite Investigation and Discovery.* Retrieved September 24, 2021, from https://pubs.acs.org/doi/10.1021/acs.chemmater.0c01153
10. Häse, F., Aldeghi, M., Hickman, R., J., Roch, L., M., Christensen, M., Liles, E., Hein, J. E., Aspuru-Guzik10, A.,. *Olympus: a benchmarking framework for noisy optimization and experiment planning.* Retrieved September 24, 2021, from https://iopscience.iop.org/article/10.1088/2632-2153/abedc8.
11. SheffieldML. (n.d.). *Sheffieldml/gpyopt: Gaussian process optimization using gpy*. GitHub. Retrieved September 24, 2021, from https://github.com/SheffieldML/GPyOpt
12. Alán Aspuru-Guzik, Florian, Loïc M., Christoph Kreisbeck. *Phoenics: A bayesian optimizer for chemistry*. ACS Publications. Retrieved September 24, 2021, from https://pubs.acs.org/doi/10.1021/acscentsci.8b00307.
13. Bergstra, J., Yamins, D., Cox, D., D. *Hyperopt: A Python Library for Optimizing the Hyperparameters of a Machine Learning Algorithm.* Retrieved September 24, 2021 from https://conference.scipy.org/proceedings/scipy2013/pdfs/bergstra_hyperopt.pdf
14. Huyer, W., Neumaier, A. *SNOBFIT – Stable Noisy Optimization by Branch and Fit.* Retrieved September 24, 2021 from https://www.mat.univie.ac.at/~neum/ms/snobfit.pdf