Summit: Benchmarking Machine Learning Methods for Reaction Optimisation

Kobi C. Felton[*a], Jan G. Rittig[*b], and Prof. Alexei A. Lapkin[†a]

[a]Department of Chemical Engineering, University of Cambridge, Cambridge, UK

[b]RWTH Aachen University, Process Systems Engineering (AVT.SVT), Aachen 52074, Germany

November 3, 2020

Abstract

In the fine chemicals industry, reaction screening and optimisation are essential to development of new products. However, this screening can be extremely time and labor intensive, especially when intuition is used. Machine learning offers a solution through iterative suggestions of new experiments based on past experimental data, but knowing which machine learning strategy to apply in a particular case is still difficult. Here, we develop chemically-motivated virtual benchmarks for reaction optimisation and compare several strategies on these benchmarks. The benchmarks and strategies are encompassed in an open-source framework named Summit. The results of our tests show that Bayesian optimisation strategies perform very well across the types of problems faced in chemical reaction optimisation, while many strategies commonly used in reaction optimisation fail to find optimal solutions.

Keywords–Machine learning, Flow chemistry, Aromatic substitution, Cross-coupling

1 Introduction

The development of novel and efficient chemical processes is essential to meeting grand challenges in healthcare, energy and sustainability.[1,2] In the fine chemicals industry particularly, there is a great need for methods to simultaneously optimise reaction throughput while also minimising environmental waste.[3] A skilled chemist can often select optimal conditions for high yield after a few experiments when

[*]These authors contributed equally to this work.

[†]Corresponding author. Email: aal35@cam.ac.uk

there is detailed mechanistic understanding of a reaction. Then, chemical engineers can develop kinetic models to design pilot plant and large scale reactors for manufacturing.[4] However, time and budget constraints often prevent detailed kinetic studies to elucidate the full mechanistic model. Therefore, scientists have to rely on screening to find optimal conditions.

The challenge with screening is that there are a vast number of possible combinations of reagents, solvents, stoichiometries and temperatures for a reaction. One author estimated that there are more than 50 million potential conditions for most catalytic reactions, so brute force exploration of the parameter space is not possible.[5] Indeed, even the most advanced automated droplet flow chemistry systems can only complete thousands of reactions per day,[6] and the 24, 96 or 384 vial plates commonly found in the pharmaceutical industry only enable tens to hundreds of reactions per week.[7–9] Since experiments often use expensive reagents, it is important to maximise the information learned and exploited from each experiment.

Design of Experiments (DoE) techniques formalise and streamline the screening that a trained chemist would do naturally.[10] These methods study the individual and combined effects of each factor that could affect yield, selectivity or any other variable that needs to be optimised. Factorial DoE is a commonly used method which tests combinations of each factor at predetermined levels. Factorial DoE is widely used in industry due to the simple interpretability of these methods and availability of several commercial software packages for setting up designs and analysing results.[5,9,11] However, the number of experiments required for a full factorial DoE campagin grows exponentially with the number of factors. For example, a project examining temperature and the ratio of two reactants, each at two levels, would require four experiments, but expanding to test a library of 10 bases and 10 solvents would require 400 experiments ($N = 2^2 * 10 * 10 = 400$). While it is possible to conduct a small experimental design and then follow-up based on the results, the scientist has to make a decision about which variables are most important to screen first. Furthermore, factorial DoE tests continuous variables at predefined values, so optima in between these values might be missed.

To overcome the issues with factorial DoE algorithms, recent work has turned to machine learning (ML) strategies that use previous experimental data to suggest single experiments or small batches.[3,12,13] By learning from each experiment, these ML strategies claim to have greater efficiency than factorial methods, which plan all experiments in a batch. Additionally, these strategies can test values throughout a range for each continuous factor, enabling more precise optimisation. ML has been applied to a wide variety of reactions including oxidation,[14] Diels-Alder,[15] methylation,[16] Paal-Knorr,[17] cross-coupling,[18–21] asymmetric hydrogenation,[22] and C-H activation.[23]

Choosing the most efficient optimisation strategy for a particular chemical reaction remains a challenge. The number of machine learning methods that could be used for reaction optimisation has exploded in recent years. Often strategies have been applied on different case studies of varying diffi-

culty, and previous studies have shown that an algorithm which performs well on one reaction might be inefficient on another.[14,24] Furthermore, each strategy has limited capabilities and can only be applied to certain types of optimisation problems. For example, the SNOBFIT optimisation algorithm has been used by several authors for optimising yield by changing continuous variables (e.g., temperature and stoichiometry),[14,25,26] but it does not work with categorical variables such as solvents or bases. When ML strategies have been applied to selection of categorical variables, only a small number of categorical options have been used.[19] However, recent work has shown that descriptors of a solvent or catalyst can be be used to extrapolate performance from a small number of experiments to a large library.[22,27]

An additional challenge is that there are often trade-offs between optimisation objectives. For example, using harsher conditions (i.e., higher temperature or concentration) will increase conversion in many reactions but also cause side reactions that reduce selectivity. In a broader sense, using a particular catalyst might increase reaction yield but be too expensive to use at commercial scale. These are called multi-objective optimisation problems, and their solutions are Pareto optimal points.[3,28,29] Pareto optimal points are defined as a set of solutions where an improvement in one objective necessarily causes a worsening of the other. Recently, several approaches have been developed for multi-objective optimisation,[13,28–30] but a rigorous study has not been completed to compare them all on standardised benchmarks.

Herein, we present a comparison of a wide range of potential strategies for optimising chemical reactions on two chemically motivated *in silico* benchmarks (see Figure 1). A benchmark is a standard tool in the ML field that enables practitioners to compare the performance of different ML strategies. For example, a benchmark containing over one million everyday objects enabled comparisons of image recognition models on the same easily-accessible dataset.[31] Much of the rapid progress in the field of image classification over the last ten years can be attributed to this benchmark. Instead of images, our benchmarks are models of reactions that provide predictions of reaction outcomes (e.g., yield) given reaction conditions. Our benchmarks act like virtual laboratories in which researchers can test the efficiency of each strategy through virtual experiments. We chose benchmarks that had already been validated experimentally, so anyone could use them without needing to run laboratory experiments. Each benchmark represents a different aspect of the chemical development process. The benchmarks present the challenges of process development, first with a S_NAr reaction run in a flow chemistry reactor and, then, selecting the optimal catalyst and ligand for a Pd-catalysed cross-coupling reaction. We note that a contemporaneous study, which also enables benchmarking of different optimisation strategies, aims at a broader range of experimental disciplines in chemistry and material science.[32] Ours focuses specifically on chemical reactions, and we are the first to include extensive comparisons of different strategies.

The advantage of our approach is that it allows each strategy to be compared on a standard basis without the need for expensive laboratory experiments. In preparing this study, we ran over 50,000

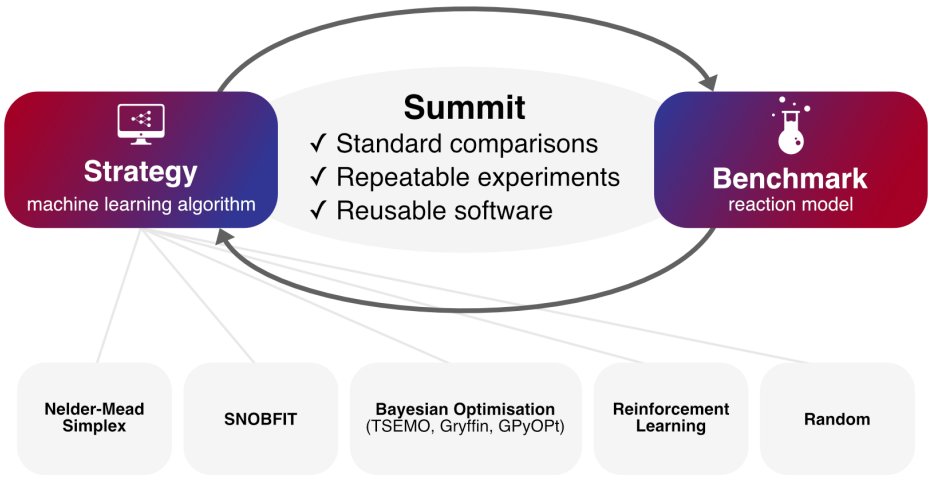


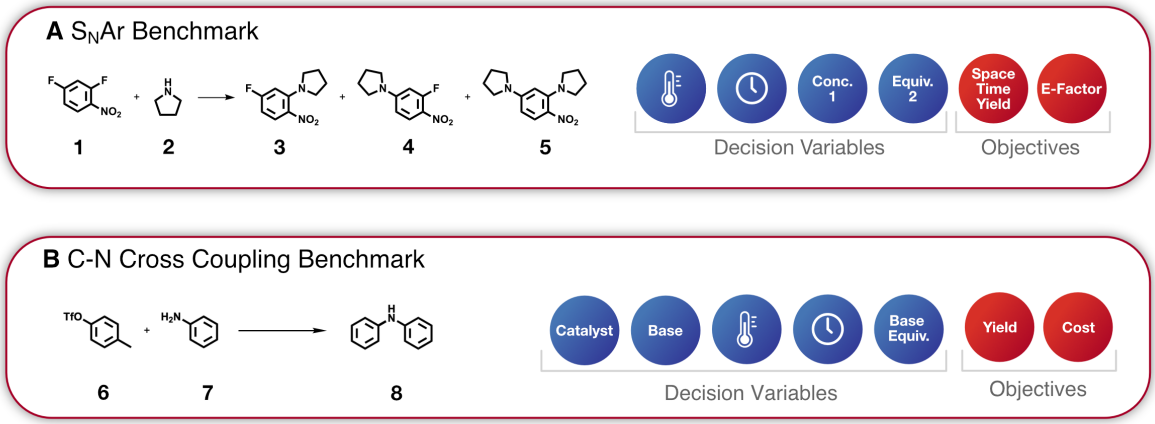
Figure 1: Overview of the approach used by Summit. Strategies are machine learning algorithms or baselines. Benchmarks are models of reactions that simulate physical experiments. Strategies and benchmarks are used together in an iterative approach to find optimal reaction conditions. This software framework enables standard comparisons of strategies on different repeatable benchmarks without the need for expensive experiments. Furthermore, the same framework can be applied confidently to real experimental case studies.

virtual reactions, which would have been cost-prohibitive in a laboratory. To facilitate others using our workflow, we release an open-source software package for optimising reactions called Summit. Summit contains the benchmarks developed in this work and a framework for implementing more in the future. Furthermore, it includes standard implementations of seven strategies and can easily be expanded to include more. We envision that researchers will be able to use Summit to test optimisation strategies on benchmarks and, then, use the same software for optimisation of their physical experiments. This will be particularly important for emerging work on closed loop optimisation, where experiments suggested by a strategy are executed by an automated experimental setup and fed back to the algorithm.[23,33–37]

2 Results and Discussion

2.1 Developing Benchmarks and Strategies for Reaction Optimisation

We propose two approaches to developing benchmarks in Summit. The first is to use mechanistic models that include reaction kinetics and potentially other phenomena such as mass transport. These models are typically differential equations that can be integrated to find reaction yield and selectivity.[19,38] In Summit, these equations are easily integrated using numerical integration software available in packages like SciPy,[39] and a standard interface saves researchers the time of re-implementing common functionality. Alternatively, when a mechanistic model is not available, it is possible to train a machine learning model on experimental data to predict yield or other reaction performance metrics given reaction conditions.[40] If data is available in the form of a CSV file, this can be achieved in Summit using as little as two lines of code. Summit is the first package to offer a simple workflow for developing both of these types of



Scheme 1: Benchmarks used in Summit for evaluating the performance of different machine learning strategies. A) Four dimensional optimisation of a S_NAr reaction between difluoronitrobenzene (**1**) and pyrrollidine (**2**).[41] The objective is to maximise space time yield and minimise E-factor. B) Optimisation of a C-N cross coupling between aryl triflate (**6**) and aniline (**7**).[42] The strategies must select one of three Buchwald catalysts and one of four organic bases for each experiment. Additionally, strategies must select the temperature, residence time, and base equivalents. The objectives are to maximise yield and minimise cost.

benchmarks.

As shown in Scheme 1A, the first benchmark is the optimisation of a nucleophilic aromatic substitution (S_NAr) reaction in a flow reactor. The benchmark is based on a kinetic model developed by Hone et al.,[41] for the reaction of difluoronitrobenzene **1** and pyrrolidine **2**, which produces one desired product and two side products. The reaction is carried out in a virtual flow reactor, so the natural objective is to maximise space time yield (STY), the mass of product formed per unit residence time. Additionally, we minimise the E-factor, which is defined as the ratio of product formed to waste produced.[1] The decision variables (i.e,. the reaction conditions) are the temperature (40-120 °C), residence time (0.5-2 minutes), equivalents of **2** (1.0-5.0) and concentration of difluoronitrobenzene (0.1-0.5M). Further details of the S_NAr benchmark can be found in the SI.

As shown in Scheme 1B, our second benchmark is the optimisation of a Pd-catalysed C-N cross coupling reaction. The aim is to select the best of three catalysts (t-BuXPhos, t-BuBrettPhos, AlPhos) and four organic bases (TEA, TMG, BTMG, and DBU) for the reaction as well as the residence time (1-30 minutes), temperature (30-100 °C) and base equivalents (1.0-2.5). Since it is difficult to find fundamental kinetic models that span across multiple catalysts and reagents, we build a predictive model on data from 96 reactions published by Baumgartner et al.[42] As shown in Figure S7, the trend of the data is captured by the model, and the cross-validation results show that the model predicts yield on unseen reaction conditions with approximately 8% mean absolute error. We use the trained model as a virtual experiment that the optimisation strategies consult for yield at given conditions. In addition to optimising yield, we also minimise cost of the reagents, so this becomes a multi-objective optimisation.

We note that we set the bounds on the decision variables by observing the limits of the experimental data from the original sources for both benchmarks. When optimising a new reaction the lab, a chemist's prior knowledge and a small number of scoping experiments could be used to set these bounds judiciously. For example, to prevent precipitation in a flow reactor, a researcher could chose bounds based on a small number of solubility experiments.

There are a large number of optimisation strategies that could be compared in this work. However, we are most interested in strategies designed to achieve state of the art performance in a small number of iterations; this naturally excludes many evolutionary and gradient based optimisation techniques. We opt to implement in Summit seven different strategies that have been used for reaction optimisation previously but not compared in a standard fashion, as shown in Figure 1. Nelder-Mead simplex is a local optimisation strategy that builds simplexes, or multidimensional triangles, that are gradually shrunk to approach the optimum value.[43] SNOBFIT builds quadratic response surfaces around the experimental data and optimises these to find new experimental conditions.[44] Bayesian optimisation is a class of strategies that train a probabilistic model to predict the objectives given the experimental conditions; sample this probabilistic model; and optimise it *in silico* to predict the best subsequent conditions. We develop implementations of three promising Bayesian optimisation strategies: single objective Bayesian optimisation (SOBO),[45] Gryffin[40] and TSEMO.[28] Finally, we develop an implementation of the Deep Reaction Optimiser (DRO) proposed by Zhou et al., which uses a pretrained reinforcement learning agent to optimise reactions.[46] Further details of each strategy can be found in the SI. As a baseline, we also include a random strategy.

In addition to benchmarks and strategies, we implement the concept of domains and transforms in Summit. Domains specify the decision variables and objectives for a particular optimisation experiment (e.g., all benchmarks include a domain). Transforms enable a domain to be reconfigured to improve the performance of optimisation or enable adaptation of a domain to a particular strategy. For example, we implement transforms for converting multi-objective optimisation problems into single objective optimisation problems. These transforms are used in the S_NAr benchmark.

2.2 S_NAr optimisation highlights Bayesian strategies

For the S_NAr benchmark, there is a negative correlation between STY and E-factor, so it is possible to optimize both simultaneously. As shown in Figure 2a, as STY decreases, E-factor increases because increasing product throughput does not result in significantly greater environmental waste. Most of the waste comes from the flow of the solvent ethanol, which does not change significantly between reaction conditions.[47]

Out of the strategies implemented in Summit, only TSEMO is inherently able to handle multi-objective problems.[28] For the other strategies, we implement transforms that combine the objectives

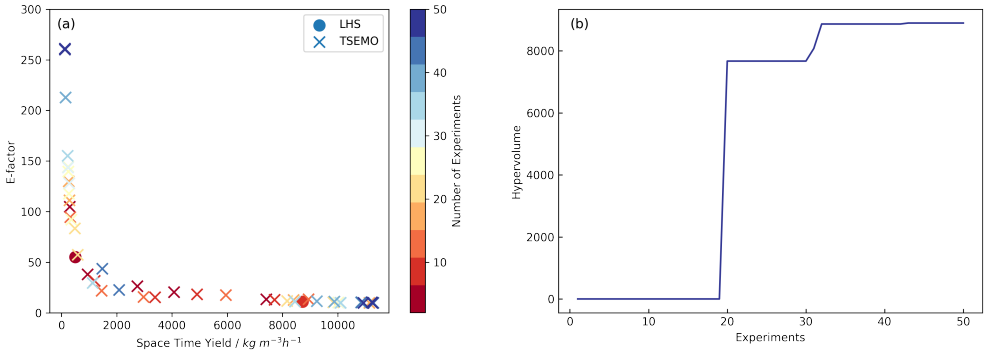


Figure 2: An exemplary optimisation of run of the S_NAr benchmark with the TSEMO. (a) The objective values of each experiment suggested by TSEMO. TSEMO explores the parameter space and exploits to find conditions that give high space-time yield (>10,000 ($kg\ m^{-3}h^{-1}$) and low E-factor (<10.0). (b) Hypervolume trajectory of the optimisation run. Hypervolume is a measure of the number of optimal trade-offs between space-time-yield (STY) and E-factor found by a given machine learning strategy; larger hypervolumes correspond with more optimal solutions.

into a single value that can be optimised. Specifically, we implement Chimera, a hierarchical scalarisation transform that works with any number of objectives,[29] and we include a method for users to specify a custom transform of a multi-objective problem to a single objective problem.[13,30]

An exemplary run of TSEMO on the S_NAr benchmark is shown in Figure 2. Figure 2a shows the objective values of each experiment suggested by TSEMO. The optimisation begins with two random experiments suggested by Latin Hypercube Sampling (LHS),[48] which are then followed by suggestions by TSEMO. Out of the first suggestions, only a few have optimal STY and E-factor values. With more data, TSEMO begins to select high concentrations of **1** (near 0.5 M), high equivalents of **2** (>3.5), and short residence times (30 seconds). However, a variety of temperatures give high STY and low E-factor. To illustrate the improvement with more experiments, we plot a measure of the quality of the pareto front called hypervolume. Hypervolume is a measure of the volume of a set of points in N-dimensions, and, in this case, we take the hypervolume of the total optimal points found by a strategy after a given number of iterations. Since a larger hypervolume will always correspond with a better set of optimal values,[49] we can see if the the optimisation strategy is finding conditions that result in better values for all objectives. As shown in Figure 2b, the hypervolume starts at zero indicating no optimal points are found. Subsequently, hypervolume rapidly increases with more iterations, indicating better values of STY and E-factor are achieved.

Figure 3 compares the performance of all strategies in Summit on the S_NAr benchmark. Each strategy was run for a maximum of 50 iterations (i.e., 50 virtual reactions) and repeated with twenty different random starts to understand the influence of randomness on the performance of the strategy. Additionally, Chimera and the custom multi-objective transform were tested, and the best combination by terminal hypervolume for each strategy is shown. Figure 3a plots the hypervolume after 50 iterations and

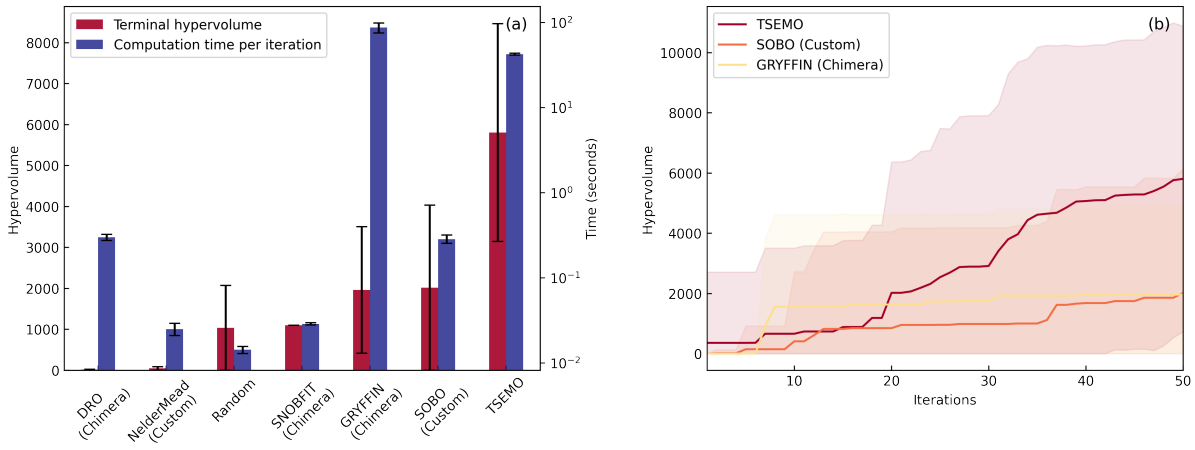


Figure 3: Comparison of several strategies on the S_NAr benchmark. (a) Trade-offs between improved performance and higher computation time. If used, a multi-objective transform is displayed in parenthesis. (b) Comparison of the hypervolume trajectories of three Bayesian optimisation strategies. The mean is plotted with the 95% confidence interval.

the average computation time per suggestion for each strategy. On average, the Bayesian optimisation strategies (GRYFFIN, SOBO, and TSEMO) perform best, finding the most optimal points in the allotted number of experiments. However, this performance comes at the price of three orders of magnitude greater computation cost. Furthermore, these tests were run on a high performance computing cluster with up to 32 threads dedicated for each strategy, and our experience is that the runtime increases by a factor of ten on consumer hardware. The longer computation time is likely acceptable in the case of TSEMO given the improvement in performance, but it might not be justified in the case of GRYFFIN or the SOBO strategies. Interestingly, Nelder-Mead Simplex, which has been used in several studies,[13,14,50–54] failed to find any optimal points, even with ten random restarts on each run; in fact, random sampling performed better. This is because Nelder-Mead is a local search strategy, and we found that the multi-objective scalarisation functions (e.g., Chimera) have many local optima (see Figure S4). The DRO also failed to find optimal points, and SNOBFIT only had slightly better performance than random sampling.

Figure 3b plots the average hypervolume of each of the Bayesian optimisation strategies over the course of 50 iterations. GRYFFIN quickly improves the quality of its suggestions with less than ten iterations, but TSEMO has better terminal performance. Both strategies have large confidence intervals with approximately 10% of repeats resulting in zero hypervolume. Still, their average performance is better than other strategies evaluated. The outstanding performance of TSEMO is likely linked to the fact that it trains individual surrogate models for each objective,[28] while the other strategies directly optimise the multi-objective transform value. As shown Figure S5 and Table S4, there were not consistent patterns in which multi-objective transform worked best.

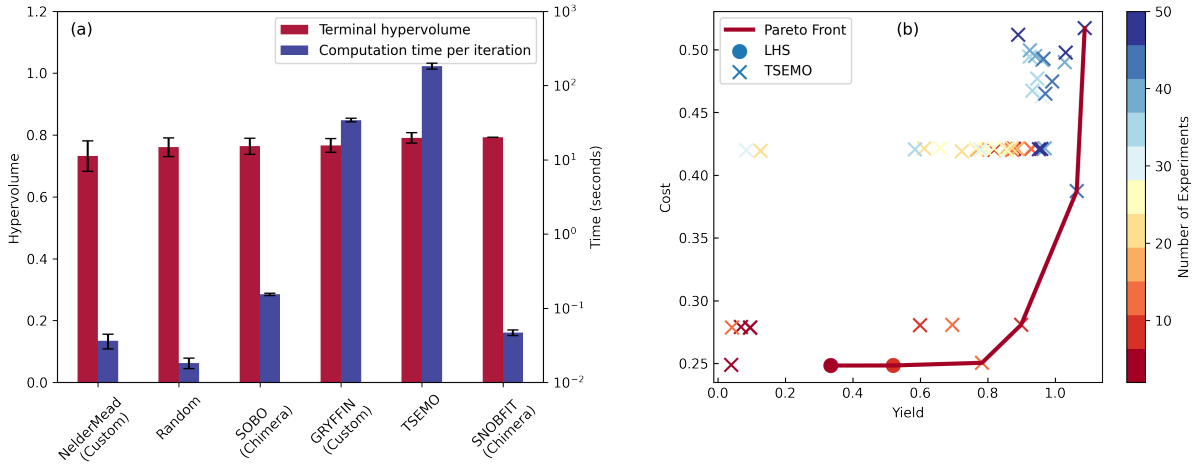


Figure 4: Comparison of several strategies on the C-N benchmark. (a) Performance of various optimisation strategies. If used, a multi-objective transform is displayed in parenthesis. (b) Comparison of the hypervolume trajectories of top performing strategies. The mean is plotted with the 95% confidence interval.

2.3 Optimisation of C-N Cross Coupling

Strategies for the C-N cross coupling benchmark need to be able to select continuous variables (temperature, base equivalents, residence time) and categorical variables (the base and catalyst). Only SOBO and GRYFFIN can inherently work with categorical variables. To overcome this challenge, we calculate a set of descriptors or properties for the catalyst and base that transform the categorical variables into continuous variables. Previous work has shown that descriptors such as melting and boiling point or those from thermodynamic programs can help the optimisation.[22,40] We use the first two of five σ-moments from computational fluid thermodynamics program COSMO-RS as these act as universal descriptors for any molecule.[55] We would prefer to use all five σ-moments, but we found that it is difficult to optimise in the resulting fourteen-dimensional input space. Using only the first two σ-moments for the catalyst and base makes it a seven-dimensional input space (three continuous variables and two categorical variables with two descriptors each). Note that DRO is not included because we empirically found that pre-training the policy on input spaces greater than six-dimensions was very slow. Since the C-N benchmark is also a multi-objective problem, we use the same transforms as in S_NAr benchmark.

As shown in Figure 4a, the performance across strategies is very similar for this benchmark, almost indistinguishable. For example, the results of the random selections are nearly equivalent to the machine learning strategies. We suggest this is because the effective parameter space is quite small. Once the optimal catalyst and base are selected, only three variables need to be tuned (reaction temperature, residence time and base equivalents). Furthermore, we see no noticeable difference between the strategies that use descriptors and those that do not use descriptors (SOBO and GRIFFYN).

The C-N benchmark presents trade-offs in the objectives. As shown in Figure S7, increasing yield often results in an increase in cost. This series of optimal trade-offs between yield and cost is called

Table 1: Pareto front data from one run of the C-N Benchmark using TSEMO. See the SI for more information about the C-N benchmark, including the catalysts and bases used.

Catalyst	**Base**	**Base Equivalents**	**Temperature (° C)**	$\tau(s)$	**Yield**	**Cost ($)**
tBuXPhos	TMG	1.32	89.86	117.73	0.34	0.25
tBuXPhos	TMG	2.38	64.35	1594.63	0.52	0.25
tBuXPhos	DBU	2.03	71.12	1040.59	0.78	0.25
tBuBrettPhos	DBU	2.28	87.16	107.66	0.90	0.28
tBuBrettPhos	BTMG	2.50	99.32	1081.24	1.06	0.39
AlPhos	BTMG	2.25	99.89	1763.66	1.09	0.52

a Pareto front. In Figure 4b, we plot the objective values for one optimisation of C-N benchmark using TSEMO. With more experiments, the Pareto front is better filled out and a greater proportion of selections have high yield. As shown in Table 1, the optimal catalyst complex is AlPhos, in line with results from Baumgartner et al.[42] However, it is clear that some conditions result in a prediction of yield being greater than 100%. Some of the original data has yield greater than 100%, apparently due to small errors in HPLC measurements.[42] Since the data-driven model has no concept of chemistry (i.e., that yield cannot be greater than 100%), it simply learns this pattern and extrapolates it. The lack of physical constraints is a known issue in data-driven models and could possibly be overcome through hybrid models that combine physical insights and learned approximations.[56,57]

3 Conclusions

In this work, we introduce Summit, a framework for optimisation of chemical reactions. We present two benchmarks for reaction optimisation and compare the performance of seven strategies with various combinations of multi-objective transforms on these benchmarks. Our results show that Bayesian optimisation performs most consistently, with TSEMO showing best-in-class performance on both benchmarks.

Our framework enables researchers to test new optimisation strategies without expensive experiments. We envision that new benchmarks could address outstanding challenges in reaction optimisation such as multi-step optimisation[37,58] and mixed objective domains (i.e., categorical and continuous objectives). Additionally, we foresee that new strategies can be tested against current and future benchmarks for performance assessment. Summit provides a simple method for implementing new benchmarks either as mechanistic models or by training data-driven models based on past experiments.

In the optimisation community, the idea of "No-free lunch" is often cited.[59] The principle is that an optimisation strategy that works well for one problem will inherently be bad for another. This implies that understanding the strengths and weaknesses of each optimisation strategy are important. Here, we have identified that Bayesian optimisation is an efficient way to overcome the challenges of reaction optimisation.

4 Acknowledgements

K.C.F. thanks the Marshall Scholarship and Cambridge Trust for PhD funding. He is also affiliated with the SynTech Centre for Doctoral Training in the Department of Chemistry, University of Cambridge. We thank Alexander Pomberger for useful feedback on this manuscript.

5 Entry for the Table of Contents

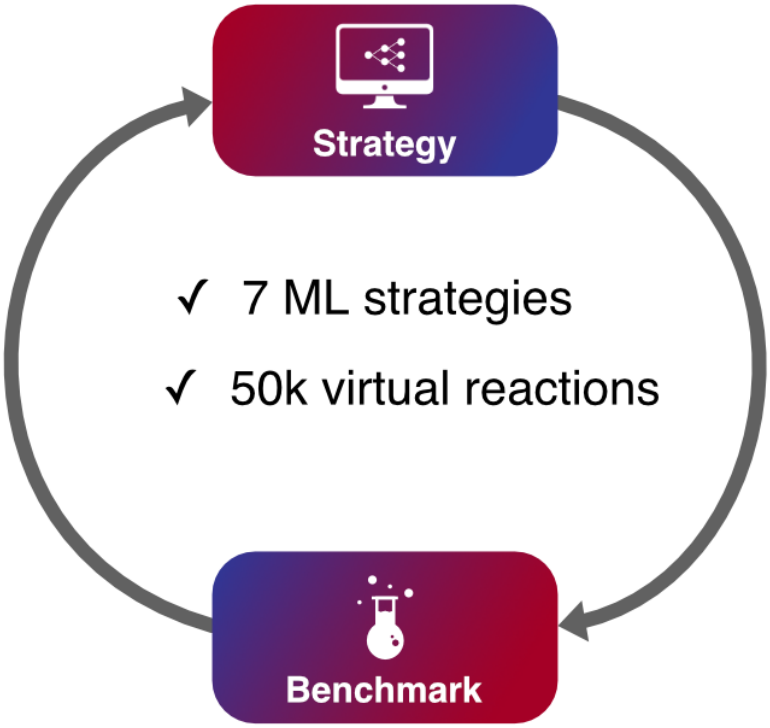


Figure 5: We introduce Summit, a framework for applying machine learning (ML) to rapid optimisation of reaction conditions. We use Summit to compare seven ML strategies on two new in silico benchmarks based on real chemical reactions—a nucleophilic aromatic substitution and a C-N cross-coupling. Using high performance computing, we execute over 50,000 virtual reactions and demonstrate that Bayesian optimisation is effective for reaction optimisation.

References

[1] R. A. Sheldon, *ACS Sustainable Chem. Eng.* **2018**, *6*, 32–48, DOI 10.1021/acssuschemeng.7b03505.

[2] L. Rogers, K. F. Jensen, *Green Chem.* **2019**, *21*, 3481–3498, DOI 10.1039/c9gc00773c.

[3] A. M. Schweidtmann, A. D. Clayton, N. Holmes, E. Bradford, R. A. Bourne, A. A. Lapkin, *Chem. Eng. J.* **2018**, *352*, 277–282, DOI 10.1016/J.CEJ.2018.07.031.

[4] G. Roberts, *Chemical Reactions and Chemical Reactors*, John Wiley and Sons, **2008**.

[5] P. M. Murray, S. N. G. Tyler, J. D. Moseley, *Org. Process Res. Dev.* **2013**, *17*, 40–46, DOI 10.1021/op300275p.

[6] D. Perera, J. W. Tucker, S. Brahmbhatt, C. J. Helal, A. Chong, W. Farrell, P. Richardson, N. W. Sach, *Science* **2018**, *359*, 429–434, DOI 10.1126/science.aap9112.

[7] A. Buitrago Santanilla, E. L. Regalado, T. Pereira, M. Shevlin, K. Bateman, L.-C. Campeau, J. Schneeweis, S. Berritt, Z.-C. Shi, P. Nantermet, Y. Liu, R. Helmy, C. J. Welch, P. Vachal, I. W. Davies, T. Cernak, S. D. Dreher, *Science* **2015**, *347*, 49–53, DOI 10.1126/science.1259203.

[8] M. Shevlin, *ACS Med. Chem. Lett.* **2017**, *8*, 601–607, DOI 10.1021/acsmedchemlett.7b00165.

[9] S. M. Mennen, C. Alhambra, C. L. Allen, M. Barberis, S. Berritt, T. A. Brandt, A. D. Campbell, J. Castañón, A. H. Cherney, M. Christensen, D. B. Damon, J. Eugenio De Diego, S. García-Cerrada, P. García-Losada, R. Haro, J. Janey, D. C. Leitch, L. Li, F. Liu, P. C. Lobben, D. W. Macmillan, J. Magano, E. McInturff, S. Monfette, R. J. Post, D. Schultz, B. J. Sitter, J. M. Stevens, I. I. Strambeanu, J. Twilton, K. Wang, M. A. Zajac, *Org. Process Res. Dev.* **2019**, *23*, 1213–1242, DOI 10.1021/acs.oprd.9b00140.

[10] R. Carlson, J. E. Carlson, *Design and Optimization in Organic Synthesis*, 2nd ed., Elsevier B.V., Amsterdam, **2005**.

[11] D. F. Emiabata-Smith, D. L. Crookes, M. R. Owen, *Org. Process Res. Dev.* **1999**, *3*, 281–288, DOI 10.1021/op990016d.

[12] B. J. Reizman, K. F. Jensen, *Acc. Chem. Res.* **2016**, *49*, 1786–1796, DOI 10.1021/acs.accounts.6b00261.

[13] D. E. Fitzpatrick, C. Battilocchio, S. V. Ley, *Org. Process Res. Dev.* **2016**, *20*, 386–394, DOI 10.1021/acs.oprd.5b00313.

[14] J. P. McMullen, K. F. Jensen, *Org. Process Res. Dev.* **2010**, *14*, 1169–1176, DOI 10.1021/op100123e.

[15] J. P. McMullen, K. F. Jensen, *Org. Process Res. Dev.* **2011**, *15*, 398–407, DOI 10.1021/op100300p.

[16] R. A. Bourne, R. A. Skilton, A. J. Parrott, D. J. Irvine, M. Poliakoff, *Org. Process Res. Dev.* **2011**, *15*, 932–938, DOI 10.1021/op200109t.

[17] J. S. Moore, K. F. Jensen, *Org. Process Res. Dev.* **2012**, *16*, 1409–1415, DOI 10.1021/op300099x.

[18] B. J. Reizman, Y.-M. Wang, S. L. Buchwald, K. F. Jensen, *React. Chem. Eng.* **2016**, *1*, 658–666, DOI 10.1039/c6re00153j.

[19] L. M. Baumgartner, C. W. Coley, B. J. Reizman, K. W. Gao, K. F. Jensen, *React. Chem. Eng.* **2018**, *3*, 301–311, DOI 10.1039/C8RE00032H.

[20] H. W. Hsieh, C. W. Coley, L. M. Baumgartner, K. F. Jensen, R. I. Robinson, *Org. Process Res. Dev.* **2018**, *22*, 542–550, DOI 10.1021/acs.oprd.8b00018.

[21] D. Reker, G. Bernardes, T. Rodrigues, *ChemRxiv* **2018**, DOI 10.26434/chemrxiv.7291205.v1.

[22] Y. Amar, A. M. Schweidtmann, P. Deutsch, L. Cao, A. Lapkin, *Chem. Sci.* **2019**, DOI 10.1039/C9SC01844A.

[23] A. Echtermeyer, Y. Amar, J. Zakrzewski, A. Lapkin, *Beilstein J. Org. Chem.* **2017**, *13*, 150–163, DOI 10.3762/bjoc.13.18.

[24] F. Häse, L. M. Roch, C. Kreisbeck, A. Aspuru-Guzik, *ACS Cent. Sci.* **2018**, *4*, 1134–1145, DOI 10.1021/acscentsci.8b00307.

[25] S. Krishnadasan, R. J. C. Brown, A. J. DeMello, J. C. DeMello, *Lab Chip* **2007**, *7*, 1434, DOI 10.1039/b711412e.

[26] N. Holmes, G. R. Akien, R. J. D. Savage, C. Stanetty, I. R. Baxendale, A. J. Blacker, B. A. Taylor, R. L. Woodward, R. E. Meadows, R. A. Bourne, *React. Chem. Eng.* **2016**, *1*, 96–100, DOI 10.1039/C5RE00083A.

[27] J. P. Reid, M. S. Sigman, *Nature* **2019**, *571*, 343–348, DOI 10.1038/s41586-019-1384-z.

[28] E. Bradford, A. M. Schweidtmann, A. Lapkin, *J. Glob. Optim.* **2018**, *71*, 407–438, DOI 10.1007/s10898-018-0609-2.

[29] F. Häse, L. M. Roch, A. Aspuru-Guzik, *Chem. Sci.* **2018**, *9*, 7642–7655, DOI 10.1039/c8sc02239a.

[30] R. W. Epps, M. S. Bowen, A. A. Volk, K. Abdel-Latif, S. Han, K. G. Reyes, A. Amassian, M. Abolhasani, *Adv. Mater.* **2020**, 2001626, DOI 10.1002/adma.202001626.

[31] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, L. Fei-Fei, *International Journal of Computer Vision* **2015**, *115*, 211–252, DOI 10.1007/s11263-015-0816-y.

[32] F. Häse, M. Aldeghi, R. J. Hickman, L. M. Roch, M. Christensen, E. Liles, J. E. Hein, A. Aspuru-Guzik, *arXiv preprint* **2020**, arXiv: 2010.04153.

[33] M. I. Jeraal, N. Holmes, G. R. Akien, R. A. Bourne, *Tetrahedron* **2018**, *74*, 3158–3164, DOI 10.1016/J.TET.2018.02.061.

[34] N. Holmes, G. R. Akien, A. J. Blacker, R. L. Woodward, R. E. Meadows, R. A. Bourne, *React. Chem. Eng.* **2016**, *1*, 366–371, DOI 10.1039/C6RE00059B.

[35] N. Cherkasov, Y. Bai, A. J. Expósito, E. V. Rebrov, *React. Chem. Eng.* **2018**, *3*, 769–780, DOI 10.1039/C8RE00046H.

[36] A. D. Clayton, J. A. Manson, C. J. Taylor, T. W. Chamberlain, B. A. Taylor, G. Clemens, R. A. Bourne, *React. Chem. Eng.* **2019**, *4*, 1545–1554, DOI 10.1039/C9RE00209J.

[37] A. D. Clayton, A. M. Schweidtmann, G. Clemens, J. A. Manson, C. J. Taylor, C. G. Niño, T. W. Chamberlain, N. Kapur, A. J. Blacker, A. A. Lapkin, R. A. Bourne, *Chem. Eng. J.* **2020**, *384*, 123340, DOI 10.1016/j.cej.2019.123340.

[38] B. J. Reizman, PhD thesis, MIT, **2015**.

[39] P. Virtanen, R. Gommers, T. E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, E. Burovski, P. Peterson, W. Weckesser, J. Bright, S. J. van der Walt, M. Brett, J. Wilson, K. Jarrod Millman, N. Mayorov, A. R. J. Nelson, E. Jones, R. Kern, E. Larson, C. Carey, İ. Polat, Y. Feng, E. W. Moore, J. Vand erPlas, D. Laxalde, J. Perktold, R. Cimrman, I. Henriksen, E. A. Quintero, C. R. Harris, A. M. Archibald, A. H. Ribeiro, F. Pedregosa, P. van Mulbregt, S. 1. 0. Contributors, *Nat. Methods* **2020**, *17*, 261–272, DOI https://doi.org/10.1038/s41592-019-0686-2.

[40] F. Häse, L. M. Roch, A. Aspuru-Guzik, *arXiv preprint* **2020**, arXiv: 2003.12127.

[41] C. A. Hone, N. Holmes, G. R. Akien, R. A. Bourne, F. L. Muller, *React. Chem. Eng.* **2017**, *2*, 103–108, DOI 10.1039/C6RE00109B.

[42] L. M. Baumgartner, J. M. Dennis, N. A. White, S. L. Buchwald, K. F. Jensen, *Org. Process Res. Dev.* **2019**, *23*, 1594–1601, DOI 10.1021/acs.oprd.9b00236.

[43] J. A. Nelder, R. Mead, *The Computer J.* **1965**, *7*, 308–313, DOI 10.1093/comjnl/7.4.308.

[44] W. Huyer, A. Neumaier, *ACM Transactions on Mathematical Software* **2008**, *35*, 1–25, DOI 10.1145/1377612.1377613.

[45] The GPyOpt authors, GPyOpt: A Bayesian Optimization framework in python, http://github.com/SheffieldML/GPyOpt, **2016**.

[46] Z. Zhou, X. Li, R. N. Zare, *ACS Cent. Sci.* **2017**, *3*, 1337–1344, DOI 10.1021/acscentsci.7b00492.

[47] M. I. Jeraal, S. Sung, A. A. Lapkin, *Chemistry Methods* **Under Review**.

[48] R. J. Beckman, W. J. Conover, M. D. McKay, *Technometrics* **1979**, *21*, 239–245.

[49] E. Zitzler, L. Thiele, M. Laumanns, C. M. Fonseca, V. G. da Fonseca, *IEEE Trans. Evol. Comput.* **2003**, *7*, 117–132, DOI 10.1109/TEVC.2003.810758.

[50] J. P. McMullen, M. T. Stone, S. L. Buchwald, K. F. Jensen, *Angew. Chem. Int. Ed. Engl.* **2010**, *49*, 7076–7080, DOI 10.1002/anie.201002590.

[51] A. J. Parrott, R. A. Bourne, G. R. Akien, D. J. Irvine, M. Poliakoff, *Angew. Chem. Int. Ed. Engl.* **2011**, *50*, 3788–3792, DOI 10.1002/anie.201100412.

[52] V. Sans, L. Porwol, V. Dragone, L. Cronin, *Chem. Sci.* **2015**, *6*, 1258–1264, DOI 10.1039/C4SC03075C.

[53] D. Cortés-Borda, K. V. Kutonova, C. Jamet, M. E. Trusova, F. Zammattio, C. Truchet, M. Rodriguez-Zubiri, F.-X. Felpin, *Org. Process Res. Dev.* **2016**, *20*, 1979–1987, DOI 10.1021/acs.oprd.6b00310.

[54] K. Poscharny, D. C. Fabry, S. Heddrich, E. Sugiono, M. A. Liauw, M. Rueping, *Tetrahedron* **2018**, *74*, 3171–3175, DOI 10.1016/j.tet.2018.04.019.

[55] A. M. Zissimos, M. H. Abraham, A. Klamt, F. Eckert, J. Wood, *J. Chem. Inf. Comput. Sci.* **2002**, *42*, 1320–1331, DOI 10.1021/ci025530o.

[56] M. L. Thompson, M. A. Kramer, *AIChE Journal* **1994**, *40*, 1328–1340, DOI 10.1002/aic.690400806.

[57] C. Tsay, M. Baldea, *Ind. Eng. Chem. Res.* **2019**, *58*, 16696–16708, DOI 10.1021/acs.iecr.9b02282.

[58] C. W. Coley, D. A. Thomas, J. A. M. Lummiss, J. N. Jaworski, C. P. Breen, V. Schultz, T. Hart, J. S. Fishman, L. Rogers, H. Gao, R. W. Hicklin, P. P. Plehiers, J. Byington, J. S. Piotti, W. H. Green, A. J. Hart, T. F. Jamison, K. F. Jensen, *Science* **2019**, *365*, eaax1566, DOI 10.1126/science.aax1566.

[59] D. Wolpert, W. Macready, *IEEE Transactions on Evolutionary Computation* **1997**, *1*, 67–82.

Contents

1 Summit Overview

Summit is the first python package dedicated to reaction optimisation. Summit is available as open-source software on **Github** (https://github.com/sustainable-processes/summit).

Summit was designed with the principles of simplicity and flexibility in mind. Previously, most machine learning strategies for reaction optimisation have been implemented in Matlab.[1–9] Matlab offers the benefits of being relatively easy to learn and flexible enough to enable fast development of new ideas. However, we see strong opportunities in using python due its prevalence in data science and the availability of machine learning packages such as Tensorflow[10] and PyTorch.[11] By leveraging these packages, we can keep the code relatively simple. Summit has two main components:

- **Strategies** contain the logic for machine learning algorithms. Each strategy has a *suggest_ experiments* method, which takes in a dataset with any previous experimental data and generates conditions for a new set of experiments. Since Summit is written according to object oriented methodologies, new strategies can easily inherit and override the functionality of existing strategies.

- **Benchmarks** are simulations of reactions. We include a base class called *Experiment* which is inherited by benchmarks. One noteworthy functionality is the *ExperimentalEmulator* which creates a benchmark from data by training a machine learning algorithm (specifically a Bayesian Neural Network). The overall process involves specifying the decision variables and objectives; importing experimental data from a CSV file and using one line of code to train the model. This process is quick and easy. We also note that the *Experiment* class could be used to connect Summit with automated laboratory equipment or even remote laboratories; this would enable closed-loop and self- optimisation on real case studies.

In Figure S1, we demonstrate how these components can be combined to run a closed-loop optimisation in Summit. In the first line of code, one of the included benchmarks is instantiated. Then, we instantiate one of the strategies, TSEMO, passing in the domain. Domains describe the decision variables and objectives of the optimisation. Finally, we pass the strategy and benchmark to the constructor for *Runner*, which will run closed loop optimisation automatically and report back the results. The advantage of Summit is that this process can be executed in only three lines of code, making it easy for even non-programmers to get started. Further examples and tutorials can be found in the additional supporting information with the documentation of the Summit code or our online documentation (https://gosummit.readthedocs.io/).

We note that Summit has several key utilities:

- **Domains** describe the decision variables and objectives of the optimisation. There are two types of variables. Continuous variables have upper and lower bounds. Categorical variables represent discrete choices such as solvents or bases and can either be represented directly or include descriptors (e.g., melting point, boiling point). Additionally, domains can have constraints on the input space, though not all strategies can use these constraints.

- **Transforms** change the domain of a Benchmark or Experiment to work with a particular strategy or improve its performance. See Section 4 for more details.

- **DataSets** are containers for holding data and its associated metadata. These DataSets make transforms simple. By augmenting the spreadsheet-like functionality of Pandas DataFrames[12] with the ability to distinguish between metadata and data columns, Summit enables transforms which only modify or act on the actual data while keeping additional metadata for further analysis (e.g., internal hyperparameters of optimisation strategies).

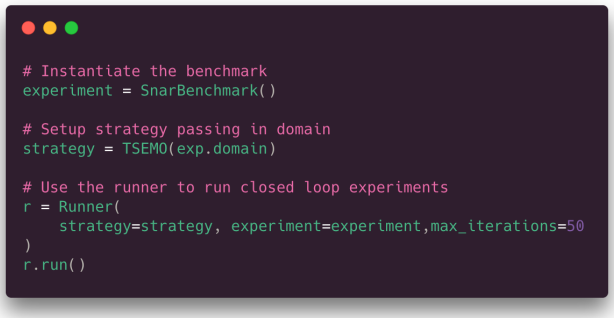


Figure S1: Example of how to set up a virtual reaction benchmark using Summit and optimize the benchmark *in silico* using TSEMO.

2 Computational Resources

All tests in this paper were completed on the University of Cambridge Service for Data-Driven Discovery (CSD3), with the exception of the NelderMead strategies. Each test was run on a single CPU with 32 threads. Example scripts for running the tests on a similar cluster or single servers are available on **Github**.

3 Strategies

The strategies available in Summit can be classified along three dimensions: the search space; the type of decision variables they can handle; and the number of objectives that can be optimised. Table S1, summarizes the capabilities of each type of strategy. Below, we give a brief summary of each strategy and any modifications we made to the traditional implementations.

3.1 Nelder-Mead Simplex

The Nelder-Mead Simplex (NMS)[13] is a local optimisation algorithm. NMS optimises an objective function using multi-dimensional polyhedra. For an optimisation problem with n_{DoF} degrees of freedom, the NMS spans a polyhedra with $n_{DoF}+1$ vertices within the feasible domain. Each vertex represents a set of conditions with the corresponding objective function value. During the optimisation process, vertices

Table S1: Capabilities of optimisation strategies: modified Nelder-Mead Simplex (NMS),[13] SNOBFIT,[14] Bayesian optimisation (BO), and Reinforcement Learning (RL).

Algorithm	Search space	Variables	Objectives	Applications
NMS	local	continuous	single	[2, 7, 15–19]
SNOBFIT	global	continuous	single	[1, 2, 20–24]
BO	global	continuous, categorical	single, multi	[25–27]
RL	global	continuous	single	[28]

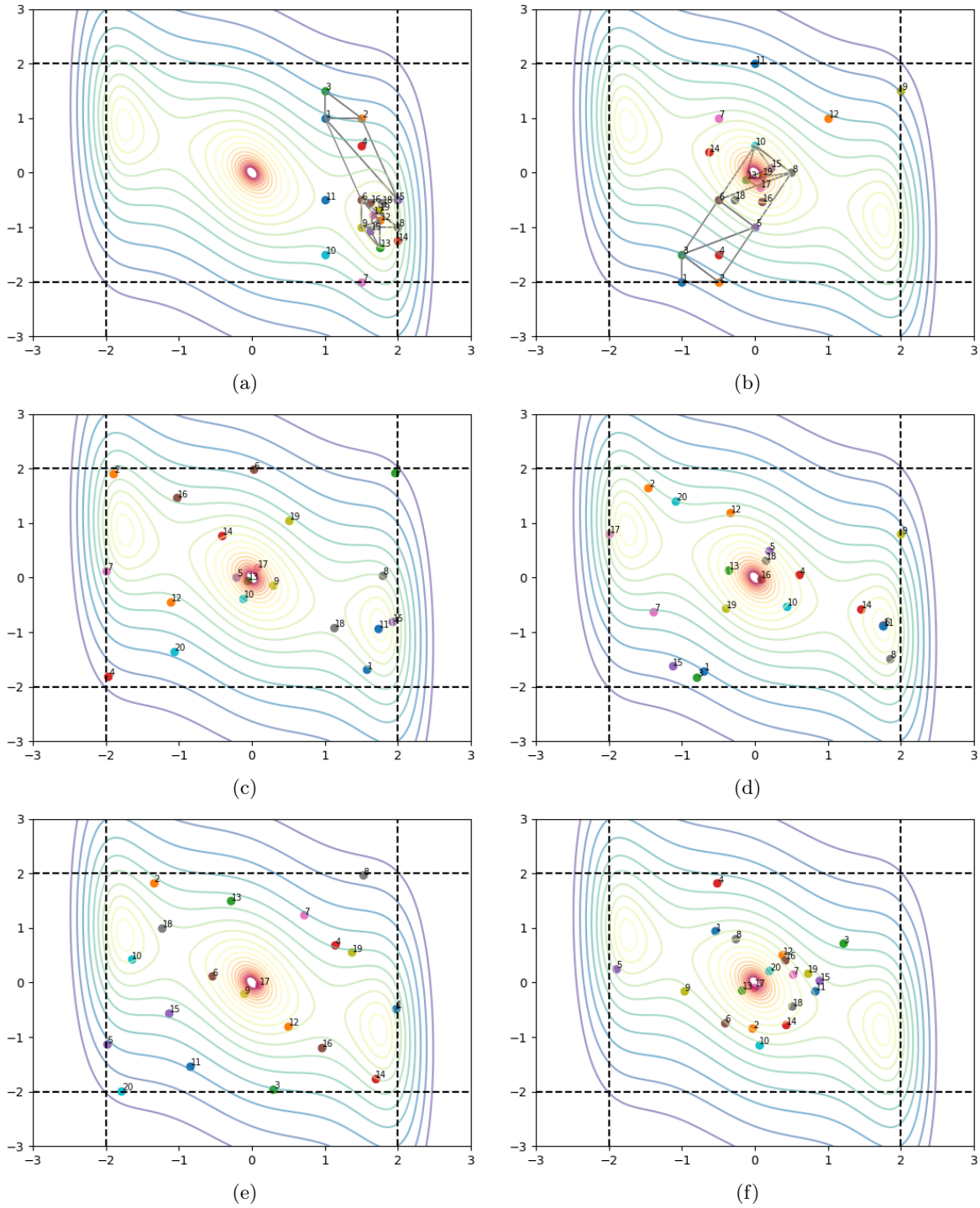


Figure S2: Applying different strategies for optimizing the Three-Hump Camel function, a standard minimitsation problem with two input parameters $x_1, x_2 \in [-2, 2]$, two local optima, and one global optimum at (0, 0); each algorithm is stopped after 20 function evaluations: (a) NMS with start point (1,1) gets stuck in local optimum; (b) NMS with start point (-1,-2) finds global optimum; (c) SNOBFIT; (d) single-objective BO; (e) Gryffin; (f) DRO.

with the worst objective function values are replaced iteratively by applying geometrical operations to the polyhedra: reflection, expansion, contraction, and shrinking.

There are several limitations to NMS. First, a different number of experiments are requested in advance of each the distinct geometrical operation. Therefore, it is impossible to request a fixed number of experiments from NMS. Second, since NMS only considers a local region of the problem domain, it can converge to local optima. This illustrated by Figure S2a where NMS converges to a local optima because the initial point is close to a local optimum that is not the global optimum. In contrast, when the initial point is near the global optima (Figure S2b), NMS quickly finds the global optimum.

We implement the NMS similar to the python SciPy package[29] with two modifications suggested by by Cortes-Borda et al.[18] First, we include a modification to account for bounded decision variables. When a point suggested by NMS is outside the bounds of the decision variables, we remove the dimension that is violating a constraint and carry out NMS in the reduced space. The optimal point from this reduced space is applied to the full problem. Second, if the optimisation problem includes linear constraints, we penalize new vertices suggested by NMS that violate at least one constraint. The geometric formation hyperparameters of the NMS we use are similar to the scipy package: $\rho = 1$ (refletion), $\chi = 2$ (expansion), $\psi = 0.5$ (contraction), $\sigma = 0.5$ (shrink).

3.2 SNOBFIT

SNOBFIT or Stable Noisy optimisation by Branch and Fit is a derivative-free optimisation algorithm.[14] It was developed to optimise expensive to evaluate objective functions globally. Therefore, SNOBFIT combines exploitation and exploration of a problem domain within lower and upper bounds of the decision variables. Figure S2c illustrates SNOBFIT applied to a two dimensional problem.

After an initial SNOBFIT call with a randomized space-filling design, exploitation is realized by fitting local quadratic models of the objective function for each point that has been evaluated. Then, SNOBFIT suggests new points by optimising these quadratic models. In addition to exploitation, points in unexplored regions of the problem domain are suggested. We treat constraints as hidden constraints.[14] That is, if a point suggested by SNOBFIT violates at least one constraint, we return NaN as objective function value.

We noticed that SNOBFIT sometimes suggest more experiments than requested by the user. This is most common if the number of experiments is low. We believe this could be caused by the exploratory part of SNOBFIT.

We set up hyperparameters similar to the default settings of SQSnobFit: probability a point of class 4 is generated = 0.5; resolution vector "dx" = 10^{-5}, i.e., two points are considered to be different if they differ by at least a factor of "dx" with respect to the difference of the upper and lower bounds in at least one coordinate i; unknown uncertainty of function evaluation = "sqrt(numpy.spacing(1))".

3.3 Bayesian Optimisation

Bayesian optimisation (BO) is a sequential model-based approach utilizing surrogate models to define beliefs about the true objective function.[30] These surrogate models are updated every time experiments are conducted. To suggest a new experiment, so-called acquisition functions that represent the utility of candidate points for the next evaluation are drawn and optimised.[30] This also enables the BO strategies to suggest multiple points for the next evaluation sequence. Our strategies include both single-objective and multi-objective BO.

3.3.1 GPyOpt

For single-objective BO we use GPyOpt, a python BO package developed by the Machine Learning Group of the University of Sheffield.[31] In our benchmarks, we apply the following hyperparameter settings to the Bayesian Optimisation method of GPyopt ("GPyOpt.methods.Bayesianoptimisation"): "model_type" = "GP" (Gaussian Process), "kernel" = "Matern52", "acquisition_type" = "EI" (Expected Improvement), "acquisition_optimiser" = "lfbgs", "normalize_Y" = true, "evaluator_type" = "random", ARD = true, "exact_feval" = false. For a detailed description of these hyperparameters, please refer to the GPyOpt package documentation.[31] Figure S2d illustrates single-objective BO applied to a two dimensional problem.

3.3.2 TSEMO

For multi-objective BO we use TSEMO, Thompson-Sampling for efficient multi-objective, proposed by Bradford et al.[8] It works by training a Gaussian Process (GP) to predict each objective given reaction conditions. Since GPs output a distribution instead of a point estimate, the GP itself cannot be optimised. TSEMO takes the approach of spectral sampling a deterministic function from the GP and optimizing that using NSGA-II.[32] From the feasible points suggested by NSGA-II, the one(s) with maximum hypervolume improvement are selected.

We found several key steps for proper implementation. For the training of the GPs (i.e., hyperparameter optimisation), at least 100 optimisation restarts are needed to consistently find the optimal log likelihood, especially with small data. It is also necessary to constrain the hyperparameters and add priors so they will not train to extremely large or small values early on. Spectral sampling will randomly fail due to problems with the singular value decomposition (SVD). Therefore, we had to implement "retries" in the code to restart the spectral sampling in these cases. Also, 1500 spectral sampling points were needed to get good results with reasonable computational performance. For comparison, the original code even used 4000 spectral points but implemented several portions of the code in C, since the SVD is the most computationally intensive part of TSEMO. Finally, it is important to have a good implementation of NSGA-II; we used pymoo.[33]

3.3.3 Gryffin

Gryffin is a BO approach concerning the challenge of categorical input variables in optimisation and was recently presented by Häse et al.[34] Gryffin offers to support the optimisation algorithm by including domain knowledge in the form of assigning descriptors to the possible values of the categorical input variables (categorical options). Figure S2e illustrates Gryffin applied to a two dimensional problem.

Since descriptors provide information about similarity or diversity of the categorical options, Gryffin utilizes this additional information by redefining the distance between the different categorical options depending on the their respective associated descriptor values. So, instead of using the frequently employed one-hot encoding of categorical options with similar distance between all options (e.g., in GPyOpt[31]), Gryffin defines the distance between categorical options to be the euclidean distance of the real-valued descriptors uniquely assigned to each option. In this way, a relationship of different categorical options is quantified and becomes measurable, providing additional knowledge to the optimisation algorithm.

Gryffin has two modes: a naive, a static, and a dynamic approach.[34] Naive Gryffin ignores descriptors; static Gryffin includes the descriptors "as given"; dynamic Gryffin also includes descriptors and furthermore infers a transformation of the initial descriptors by a one-layer neural network during the optimisation process. To simultaneously optimise categorical and continuous input variables, Gryffin internally integrates the Phoenics framework.[35]

In our benchmarks, we use Gryffin with the following settings which are similar to the default configuration of Gryffin: "auto_desc_gen" = True (dynamic Gryffin), "batches" = 1, "parallel" = true, "boosted" = true, "sampler" = "uniform", "softness" = 0.001, "continuous_optimiser" = "adam", "categorical_optimiser" = "naive", " sampling_strategies" = "4". We use this number of sampling strategies as we found it to work well with Gryffin. Additionally, we found that using precreated descriptors often led to dimensional explosion and caused the strategy to occupy large amounts of RAM.

3.4 Reinforcement Learning

Reinforcement learning agents learn to make optimal decisions given information about a problem.[36] These agents consist of a policy, which takes in a state (e.g., previously tried reaction conditions and corresponding yields) and suggests an action (e.g., a new set of reaction conditions). The optimality of an action or a set of actions is quantified through a reward. Zhou et al. built the bridge from RL to optimizing chemical reactions: an algorithm that (decision-maker) sequentially suggests experiments to carry out (actions) resulting in the highest yield (reward).[28]

Thereby, they proposed the Deep Reaction optimiser (DRO), a recurrent neural network architecture with a RL-motivated loss function that is proportional to the yield obtained from each experiment suggested and carried out. It was assumed that the response surfaces for chemical reactions are continuous and can be approximated by Gaussian processes (GPs). Thus, the DRO was pretrained on GPs in order

to simulate optimising the response surfaces of chemical reactions. These pretrained models were then inferred to yield optimisation of actual chemical reactions. Figure S2f illustrates DRO applied to a two dimensional problem.

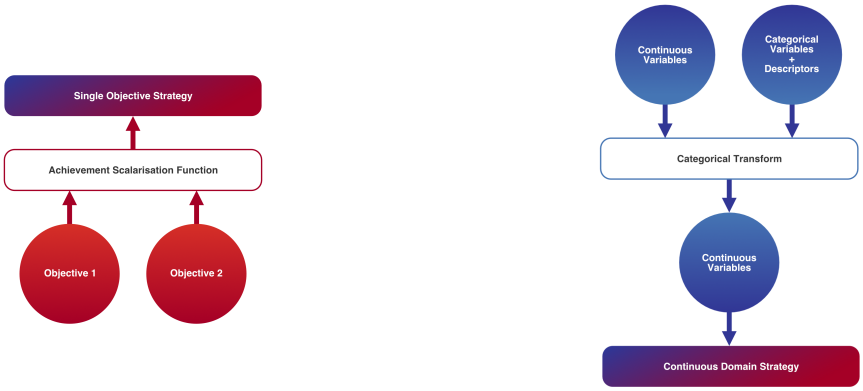
We use the same hyperparameters for pretraining the RL models as Zhou et al. except for the type of the loss function, because we found inference models pretrained with the original setting for the "loss_type" = "oi" with "reaction_type" = "gmm" to be unstable especially when the final reaction problem domain, i.e., the problem the model is inferred to, was constrained by bounds on variables. The complete list of the hyperparameters we use is as follows: "batch_size" = 128, "hidden_size" = 80, "num_layers" = 2, "batch_norm" = False, "num_layers" = 2, "batch_norm" = False, "reuse" = false, "num_epochs" = 50000, "evaluation_period" = 100, "evaluation_epochs" = 20, "reaction_type" = "gmm", "norm_cov" = 0.3, "constraints" = false, "num_params" = 3, "instrument_error" = 0.01, "num_steps" = 50, "unroll_length" = 50, "learning_rate" = 0.001, "optimiser" = "Adam", "loss_type" = "naive", "discount_factor" = 0.97, "opt_direction" = "min", "policy" = "srnn", "trainable_init" = true. Please refer to the paper by Zhou et al. for more details on these hyperparameters. According to suggestions by the authors, we increase the model size, i.e., "hidden_size", "unroll_length", and "num_epochs" of the pretraining for problem domains with more than three input variables (here "num_params"), since the problem becomes exponentially harder as the number of input variables increases. Therefore, significantly higher computational capacities are required, leading to a longer pre-training time.

4 Transforms

4.1 Multiobjective Transforms

As shown in Figure S3a, multiobjective transforms aim to convert multiobjective optimisation problems into single objective optimisation problems. This makes a wider range of strategies amenable to multiobjective problems. These transforms are often called achievement scalarisation functions (ASFs).

Chimera is a hiearchical ASF developed by Hase et al.[34,37] Hiearchical means that the objectives are weighted, so the ASF will prioritise one objective over another. The main hyperparameters are the tolerances, which are values for each objective in $[0, 1]$ that set how much each objective is weighted. A smaller tolerance indicates that the objective will be weighted more, while a larger tolerance indicates that the objective will be weighted less. The original paper does not give a clear mechanism for setting the tolerances, and the tolerances in the examples are tuned to the specific benchmarks used in the paper. To reflect what might be seen in practice, where the best optimal tolerances are not known in advance, we choose to round values for the tolerances and test four different combinations shown in Table S2. In this way, we aim to see if there was significant difference between different tolerance settings. In Figure S4, we show Chimera with different tolereances on the S_NAr benchmark.



(a) (b)

Figure S3: (a) Schematic of achievement scalarisation functions which transform multiobjective optimisation problems into single objective problems. (b) Schematic of the categorical transform, which use descriptors to represent a categorical domain in continuous space.

Table S2: Tolerance combinations for Chimera used in benchmark tests.

Objective 1	Objective 2
0.5	0.5
0.5	1.0
1.0	0.5
1.0	1.0

We also implement a transform that can do any basic arithmetic transformation of the objective values to give a single value. This could, for example, take a weighted sum of the objectives or even apply nonlinear functions (e.g., log or exponential). This is particularly useful when the user has prior knowledge about the bounds of the objective space.

4.2 Categorical Transform

The categorical transform is used to adapt domains with categorical variables to work with strategies that only work with continuous variables. As shown in Figure S3, the idea is to represent the categorical variables using their descriptors. For example, a domain containing a two continuous variables and a categorical variable with five descriptors would be transformed to a domain with seven continuous variables. The optimisation strategy could then optimise in this seven dimensional continuous space. When a desired value was obtained the inverse transform would be invoked which finds the closest categorical value in descriptor space by Euclidean distance.

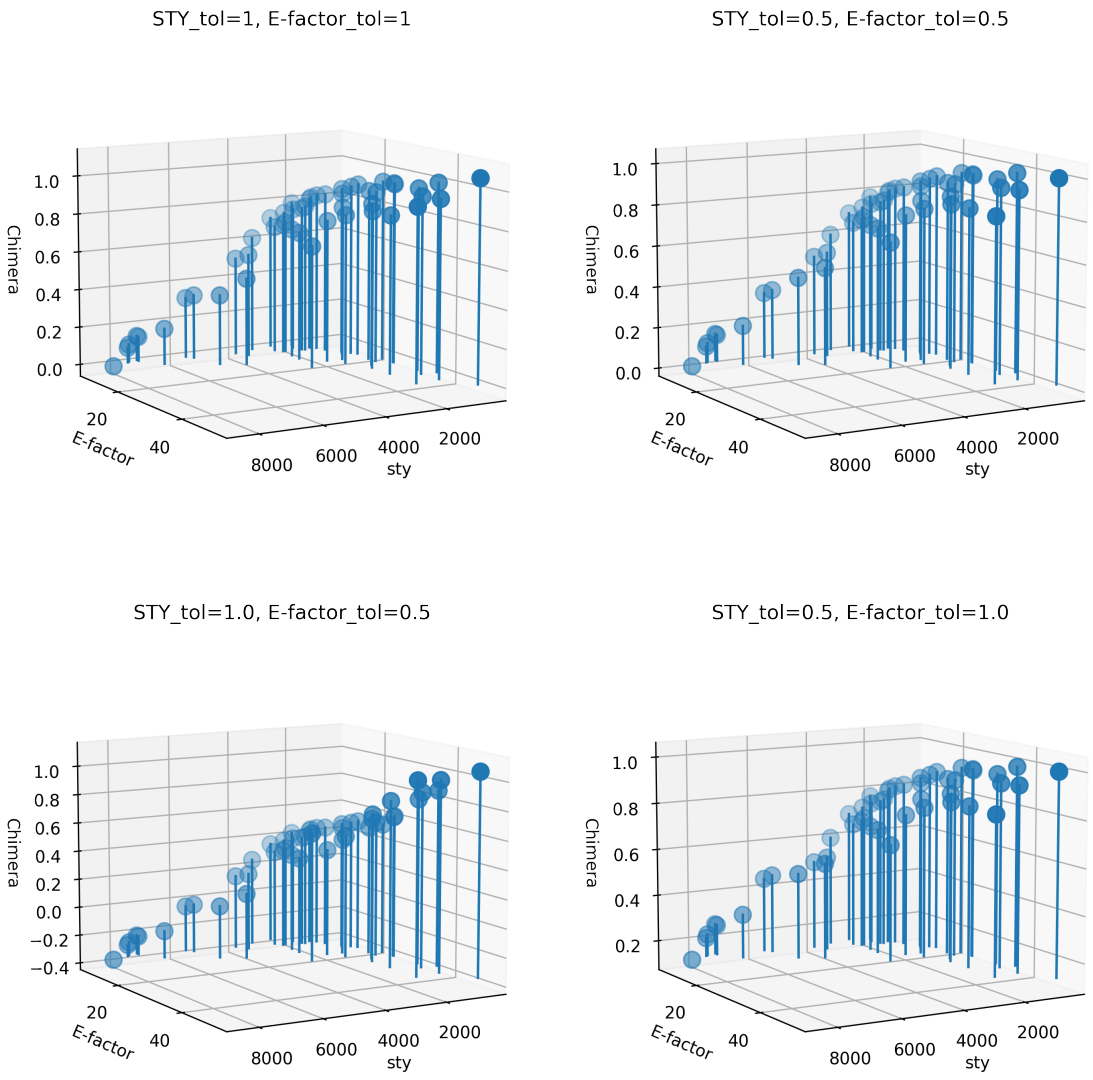


Figure S4: Comparison of the Chimera ASF on the SnAr Benchmark with different tolerances.

5 Benchmarks

5.1 S_NAr Benchmark

The S_NAr benchmark is a reaction between 2,4-difluoronitrobenzene and pyrrolidine to afford the desired products **3** with **4** and **5** as undesired side-products.[38] Here, we aim to optimise the reaction in a virtual self-optimising flow setup, as shown in Figure S5a. Formally, the optimisation problem can be stated as simultaneously maximising space time yield (STY) and minimizing the E-factor (E). As noted in Equation 1, this is achieved by adjusting the residence time τ, inlet concentration of **1** $C_{1,i}$, equivalents of pyrrolidine n_2, and reactor temperature, T.

$$\begin{aligned} &\min_X (-STY, E) \\ &\text{where } \; X = [\tau, C_{1,i}, n_2, T] \end{aligned} \tag{1}$$

In the main text, we use hypervolume as a measure of performance on this benchmark. To calculate hypervolume, a reference is needed. We use the Nadir point as a reference, which is estimated by running the NSGA-II optimisation algorithm[32] on the S_NAr benchmark for 100 generations with a population size of 100. As shown in Figure S5b, the Pareto front extends over a wide range of space time yields (2500-12 000) and a narrow range of E-factors (9.5-10.75). The Nadir point is estimated as (-2957, 10.7), given that the STY is negated to account for hypervolume assuming minimisation.

We now show the equations used to calculate the objective values for given a set of conditions. Space time yield (STY) is defined as the mass of 2,4-difluoronitrobenzene leaving the reactor per residence time (τ), and E-factor (E) is defined as the ratio of mass of waste to mass of product:

$$STY = \frac{M_3 C_3}{\tau} \tag{2}$$

$$\begin{aligned} E &= \frac{m_{waste}}{\tau V} \\ &= \frac{Q_{tot}\rho_{eth} + \frac{1}{1000}\sum_{n=1,i\neq 3}^{5} M_n C_n Q_{tot}}{\frac{1}{1000} M_3 C_3 Q_{tot}} \end{aligned} \tag{3}$$

where M_n is the molar weight in g/mol of species n; Q_{eth} and ρ_{eth} are the volumetric flowrate the density of ethanol respectively; and Q_{tot} is the total volumetric flowrate. The volume of the reactor is assumed to be 5 mL. The inlet concentration of **2** is calculated:

$$C_{2,i} = N_2 C_{1,i} \tag{4}$$

is the equivalents of pyrrolidine. Since the total flowrate Q_{tot} is $\frac{V}{\tau}$, then flowrates of the pumps from

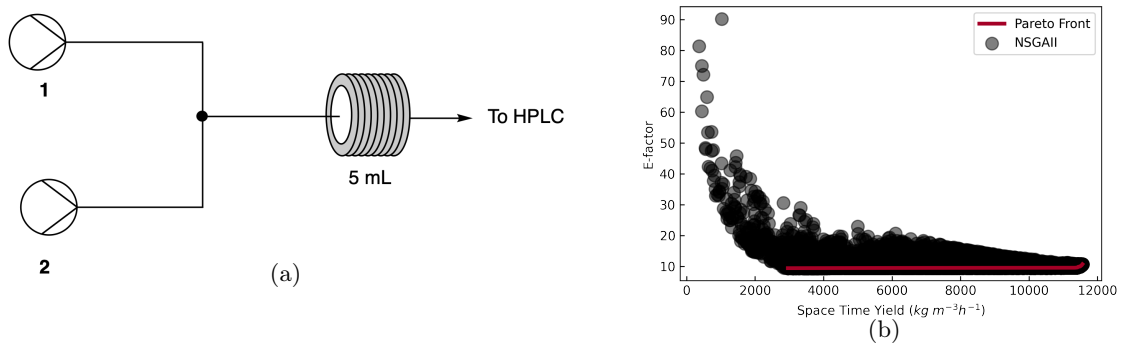


Figure S5: (a) Schematic of the virtual flow chemistry reactor used in the benchmark. (b) Optimisation landscape of S_NAr benchmark as determined by running NSGA-II for 100 generations with a population size of 100.

reservoir n are calculated:

$$Q_1 = \frac{C_{1,i}}{C_{1,0}} Q_{tot} \tag{5}$$

$$Q_2 = \frac{C_{2,i}}{C_{2,0}} Q_{tot} \tag{6}$$

where $C_{1,0} = 1M$ and $C_{2,0} = 2M$ are the reservoir concentrations of **1** and **2** respectively. To calculate the outlet concentrations of the products, the standard equation for a constant density plug flow reactor is used:

$$d\tau = \frac{dC_n}{-r_n} \tag{7}$$

where C_n and r_n are the concentration and reaction rate of species n, respectively. Hone et al. determined the reaction rates to be first order with kinetic constants k_l (see Table S3) in each species:[38]

$$r_1 = -(k_a + k_b)C_1C_2 \tag{8}$$

$$r_2 = -(k_a + k_b)C_1C_2 - k_cC_2C_3 - k_dC_2C_4 \tag{9}$$

$$r_3 = k_aC_1C_2 - k_cC_2C_3 \tag{10}$$

$$r_4 = k_aC_1C_2 - k_dC_2C_4 \tag{11}$$

$$r_5 = k_cC_2C_3 + k_dC_2C_4 \tag{12}$$

The kinetic constants k_l for each reaction are calculated using the Arrhenius equation:

$$k_l = k_{l,ref} \exp\left[-\frac{E_{a,l}}{R}\left(\frac{1}{T} - \frac{1}{T_{ref}}\right)\right] \tag{13}$$

where $k_{l,ref}$ is the constant at the reference temperature T_{ref} of 90°C and $E_{a,l}$ is the activation energy. The five differential equations described by Equation 7 or Equations 8-11 are integrated simultaneously

Table S3: Kinetic parameters used for the S_NAr benchmark based on work by Hone et al.[38]

	$k_{l,ref}$ (10^{-2}mol^{-1}dm^{3}s^{-1})	E_a (kJ mol^{-1})
$k_{a,ref}$	57.9	33.3
$k_{b,ref}$	2.70	35.3
$k_{c,ref}$	0.865	38.9
$k_{d,ref}$	1.63	44.8

over the residence time to find the final concentrations given the inlet concentrations of each species $C_{n,i}$. These concentrations are then supplied to Equations 2 and 3 to calculate the space time yield and E-factor.

Figure S6 shows the estimated Pareto fronts for the best run of each unique combination of strategy, transform, and number of initial experiments. The best run is determined by the terminal hypervolume. The reference for the hypervolume calculations is (0, 1). TSEMO finds the best combinations of high space-time-yield and E-factor. Note that for the Custom multi-objective transform, the following objective was minimised: $-STY/1000 + E/100$. Table S4 gives the complete results.

Table S4: Results of tests of strategies and transforms available in Summit on the SnAr benchmark. Each strategy and transform combination was run with twenty repeats and results are shown with the standard deviation. tol. stands for the tolerance used in Chimera where applicable, time is is the average time per iteration for a new suggestion from the strategy.

Strategy	**Transform**	**STY tol.**	**E-factor tol.**	**Hypervolume**	**Time (s)**	**Repeats**
DRO	Chimera	0.5	0.5	10.0±29.0	0.0±0.0	20
			1.0	2.0±7.0	0.0±0.0	20
		1.0	0.5	0.0±2.0	0.0±0.0	20
			1.0	7.0±29.0	0.0±0.0	20
	Custom	-	-	0.0±0.0	0.0±0.0	20
GRYFFIN	Chimera	0.5	0.5	669.0±1132.0	79.0±11.0	20
			1.0	1449.0±2243.0	78.0±11.0	20
		1.0	0.5	1715.0±1766.0	106.0±19.0	20
			1.0	1959.0±1545.0	87.0±12.0	20
	Custom	-	-	528.0±1048.0	89.0±10.0	20
NelderMead	Chimera	0.5	0.5	0.0±0.0	0.0±0.0	20
			1.0	0.0±0.0	0.0±0.0	20
		1.0	0.5	0.0±0.0	0.0±0.0	20
			1.0	0.0±0.0	0.0±0.0	20
	Custom	-	-	43.0±108.0	0.0±0.0	20
Random	Transform	-	-	1032.0±1315.0	0.0±0.0	20
SNOBFIT	Chimera	0.5	0.5	1095.0±0.0	0.0±0.0	20
			1.0	1095.0±0.0	0.0±0.0	20
		1.0	0.5	1095.0±0.0	0.0±0.0	20
			1.0	1095.0±0.0	0.0±0.0	20
	Custom	-	-	0.0±0.0	0.0±0.0	20
SOBO	Chimera	0.5	0.5	634.0±1049.0	0.0±0.0	20
			1.0	593.0±876.0	0.0±0.0	20
		1.0	0.5	1414.0±1599.0	0.0±0.0	20
			1.0	786.0±1325.0	0.0±0.0	20
	Custom	-	-	2013.0±2155.0	0.0±0.0	20
TSEMO	Transform	-	-	5803.0±2659.0	42.0±1.0	20

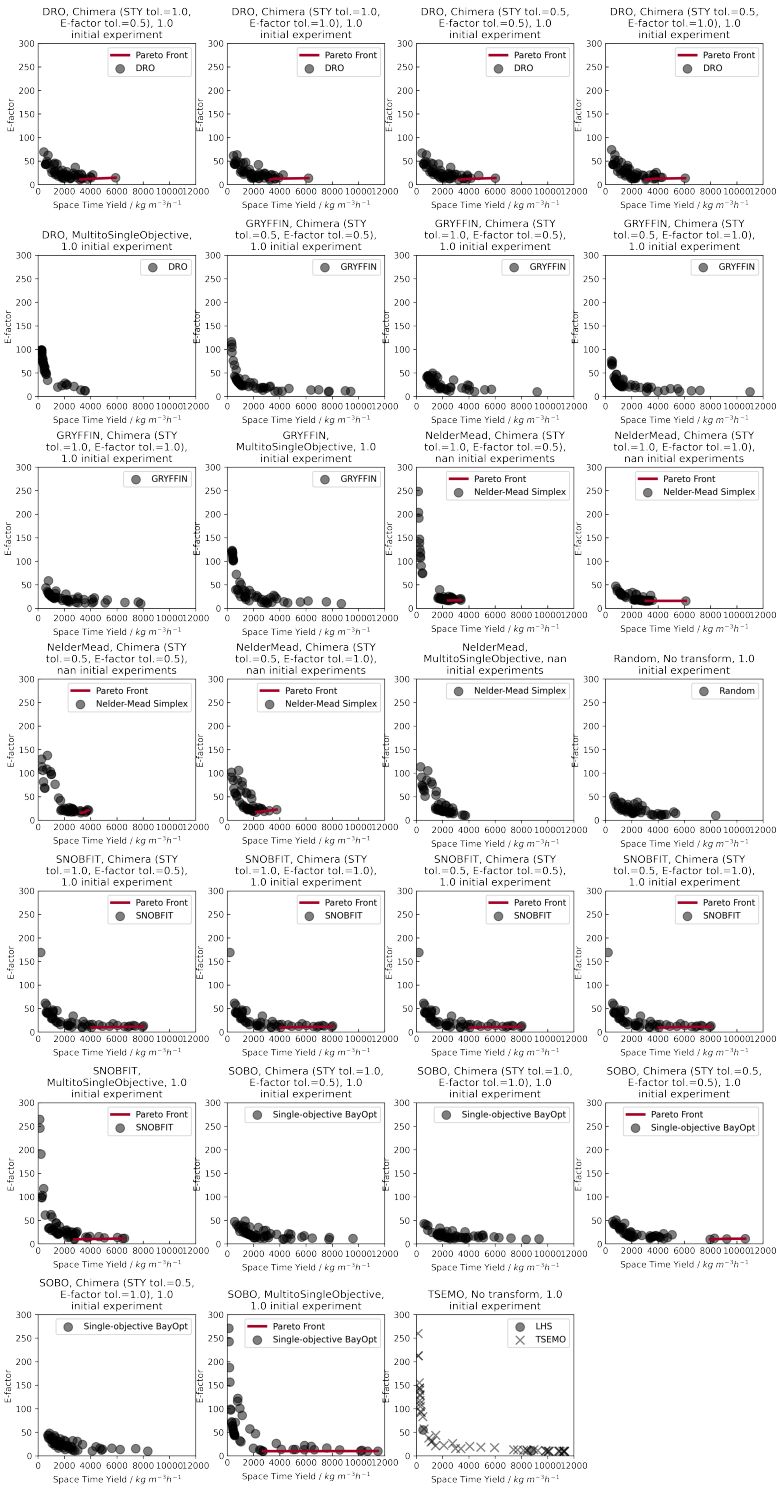


Figure S6: Pareto fronts for the best run (by terminal hypervolume) of each unique combination of strategy, transform, and number of initial experiments for the SnAr benchmark.

5.2 C-N Benchmark

The C-N benchmark represents a Pd-catalyzed cross coupling between aryl triflate and aniline.[39] This is a five dimensional optimisation of temperature, residence time, base equivalents, catalyst and base. The catalyst choices are t-BuXPhos, t-BuBrettPhos, AlPhos. The bases are triethylamine (TEA); 1,1,3,3-tetramethylguanidine (TMG); 2-tert-butyl-1,1,3,3-tetramethylguanidine (BTMG); and 1,8-Diazabicyclo[5.4.0]undec-7-ene (DBU). The categorical variables (catalyst and base) contain descriptors that are pre-calculated using COSMOQuick computational fluid thermodynamics program.[40] The descriptor are the σ-moments, which are interpretable and general descriptors for any molecule.[41] We use the first two σ-moments which correspond with area and polarizability respectively.

The yield is determined using a Bayesian Neural Network (BNN) that is trained on the experimental data from Baumgartner et al.[39] The BNN is implemented in PyTorch[11] using the BLITZ library.[42] The total dataset contains 96 experiments, where 86 are used for training and 10 for test. The model is trained using 10-fold cross validation. As shown in Figure S7, a mean absolute error of 0.08 is achieved on the test set.

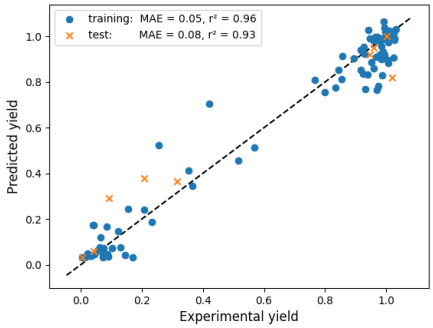


Figure S7: Parity plot for C-N benchmark using a predictive model trained on 86 data points.

The cost is determined using data from chemical suppliers, as shown in Table S5 and Table S6. For the bases, we use data supplied by Baumgartner et al.,[39] and all other data is determined by us.

Table S5: Chemicals used in C-N Benchmark.

Abbreviation	MW (g/mol)	Density (g/mL)	CAS Number	Supplier
t-BuXPhos	686.69	-	1142811-12-8	Sigma-Alddrich (Merck)
t-BuBrettPhos	854.43	-	1536473-72-9	Sigma-Alddrich (Merck)
AlPhos	815.06	-	2097600-15-0	Sigma-Alddrich (Merck)
p-Tolyl TMS	240.2	-	29540-83-8	TCI Chemicals
Anniline	93.13	1.022	62-53-3	Sigma-Alddrich (Merck)
TEA	101.19	0.726	121-44-8	Millipore-Sigma
TMG	115.18	0.918	80-70-6	BetaPharma
BTMG	171.28	0.85	29166-72-1	BetaPharma
DBU	152.24	1.018	6674-22-2	ChemShuttle
MTBD	153.22	1.067	84030-20-6	Enamine BB
BTTP	312.43	1.022	161118-67-8	Millipore-Sigma
P2Et	339.4	1.02	165535-45-5	Millipore-Sigma

Table S6: Prices of compounds used in C-N benchmark.

Abbreviation	Mass Available (g)	Volume Availability (mL)	Price	Price /mmol
t-BuXPhos	1	-	$137.00	$94.08
t-BuBrettPhos	5	-	$1,070.00	$182.85
AlPhos	1	-	$729.00	$594.18
p-Tolyl TMS	5	-	$123.00	$5.91
Anniline	-	1000	$109.00	$0.01
TEA	20000	143492.4	$1,830.00	$0.01
TMG	21786.49	173641.3	$184.00	$0.00
BTMG	117.65	583.8	$706.00	$1.21
DBU	4911.59	32842.9	$893.00	$0.03
MTBD	9.37	65.3	$511.00	$7.83
BTTP	25	81.8	$572.00	$6.99
P2Et	5	15	$601.00	$40.00

Figure S8 displays the pareto fronts of the best performing run from each combination of strategy and transform, while Table S7 enumerates the results of the tests of the C-N benchmark. Note that for the Custom multi-objective transform, the following expression was minimized: $-yield + cost$.

Table S7: Results of tests of strategies and transforms available in Summit on the CN benchmark. Each strategy and transform combination was run with twenty repeats and results are shown with the standard deviation. tol. stands for the tolerance used in Chimera where applicable, time is is the average time per iteration for a new suggestion from the strategy.

Strategy	Transform	Yield tol.	Cost tol.	Hypervolume	Time (s)	Repeats
GRYFFIN	Chimera	0.5	0.5	0.74±0.0	15.13±0.52	20
			1.0	0.74±0.01	16.57±4.38	20
		1.0	0.5	0.74±0.0	13.64±0.61	20
			1.0	0.74±0.0	14.35±1.25	20
	Custom	-	-	0.77±0.02	34.21±1.95	20
NelderMead	Chimera	0.5	0.5	0.73±0.05	0.03±0.0	20
			1.0	0.72±0.05	0.04±0.01	20
		1.0	0.5	0.72±0.05	0.04±0.01	20
			1.0	0.72±0.05	0.03±0.01	20
	Custom	-	-	0.73±0.05	0.04±0.01	20
Random	Transform	-	-	0.76±0.03	0.02±0.0	20
SNOBFIT	Chimera	0.5	0.5	0.79±0.0	0.05±0.0	20
			1.0	0.79±0.01	0.06±0.03	20
		1.0	0.5	0.79±0.0	0.05±0.0	20
			1.0	0.79±0.0	0.04±0.0	20
	Custom	-	-	0.75±0.0	0.06±0.0	20
SOBO	Chimera	0.5	0.5	0.76±0.03	0.14±0.01	20
			1.0	0.76±0.02	0.15±0.01	20
		1.0	0.5	0.76±0.03	0.15±0.01	20
			1.0	0.76±0.03	0.14±0.01	20
	Custom	-	-	0.71±0.03	0.24±0.02	20
TSEMO	Transform	-	-	0.79±0.02	182.96±16.11	20

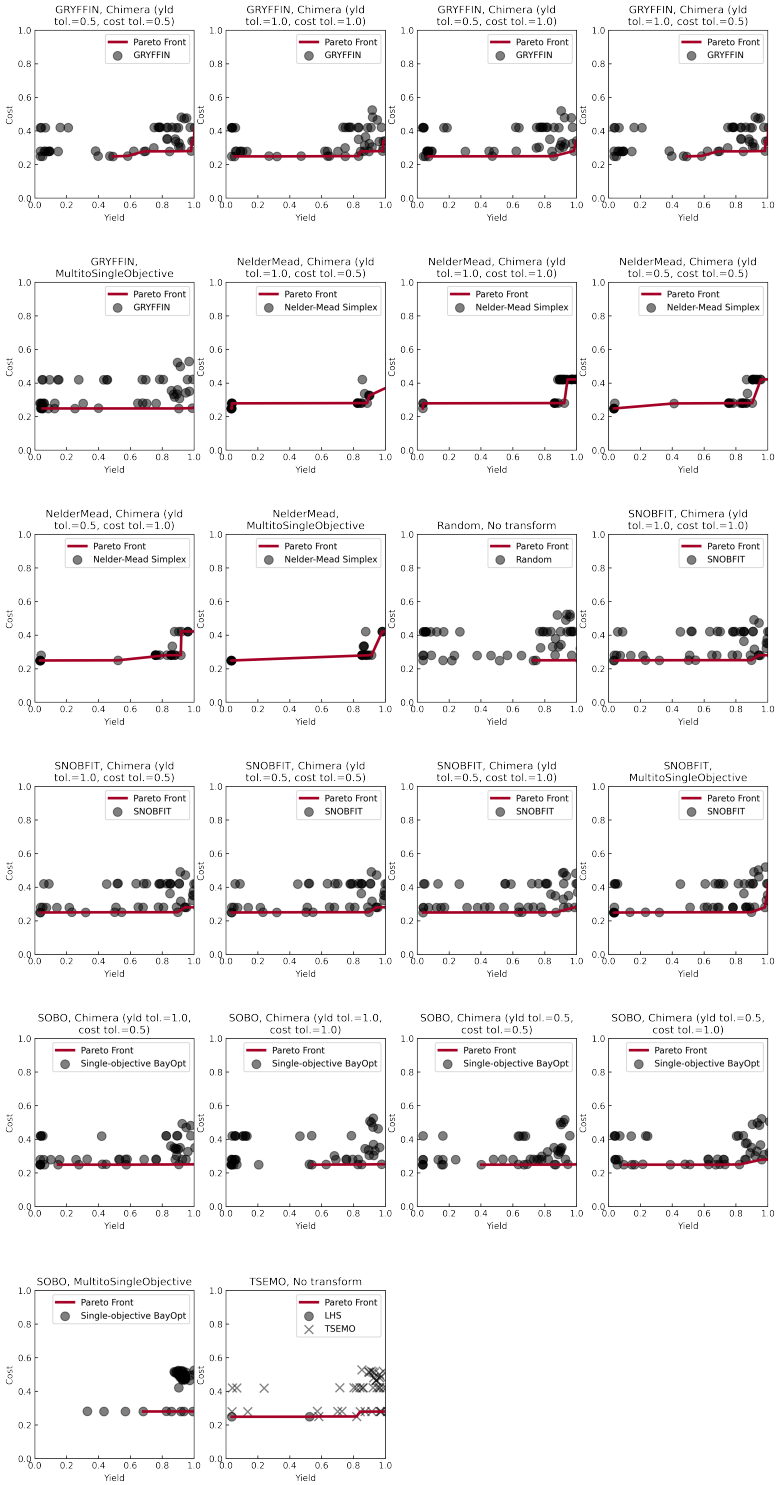


Figure S8: Pareto fronts for the best run (by terminal hypervolume) of each unique combination of strategy, transform, and number of initial experiments for the CN benchmark.

References

[1] S. Krishnadasan, R. J. C. Brown, A. J. DeMello, J. C. DeMello, *Lab Chip* **2007**, *7*, 1434, DOI 10.1039/b711412e.

[2] J. P. McMullen, K. F. Jensen, *Org. Process Res. Dev.* **2010**, *14*, 1169–1176, DOI 10.1021/op100123e.

[3] J. P. McMullen, K. F. Jensen, *Org. Process Res. Dev.* **2011**, *15*, 398–407, DOI 10.1021/op100300p.

[4] R. A. Bourne, R. A. Skilton, A. J. Parrott, D. J. Irvine, M. Poliakoff, *Org. Process Res. Dev.* **2011**, *15*, 932–938, DOI 10.1021/op200109t.

[5] J. S. Moore, K. F. Jensen, *Org. Process Res. Dev.* **2012**, *16*, 1409–1415, DOI 10.1021/op300099x.

[6] B. J. Reizman, K. F. Jensen, *Acc. Chem. Res.* **2016**, *49*, 1786–1796, DOI 10.1021/acs.accounts.6b00261.

[7] D. E. Fitzpatrick, C. Battilocchio, S. V. Ley, *Org. Process Res. Dev.* **2016**, *20*, 386–394, DOI 10.1021/acs.oprd.5b00313.

[8] E. Bradford, A. M. Schweidtmann, A. Lapkin, *J. Glob. Optim.* **2018**, *71*, 407–438, DOI 10.1007/s10898-018-0609-2.

[9] L. M. Baumgartner, C. W. Coley, B. J. Reizman, K. W. Gao, K. F. Jensen, *React. Chem. Eng.* **2018**, *3*, 301–311, DOI 10.1039/C8RE00032H.

[10] Martın Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Y. Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, Xiaoqiang Zheng, TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems, Software available from tensorflow.org, **2015**.

[11] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, S. Chintala in *Advances in Neural Information Processing Systems 32*, (Eds.: H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, R. Garnett), Curran Associates, Inc., **2019**, pp. 8024–8035.

[12] W. McKinney in Proceedings of the 9th Python in Science Conference, (Eds.: S. van der Walt, J. Millman), **2010**, pp. 51–56.

[13] J. A. Nelder, R. Mead, *The Computer J.* **1965**, *7*, 308–313, DOI 10.1093/comjnl/7.4.308.

[14] W. Huyer, A. Neumaier, *ACM Transactions on Mathematical Software* **2008**, *35*, 1–25, DOI 10.1145/1377612.1377613.

[15] J. P. McMullen, M. T. Stone, S. L. Buchwald, K. F. Jensen, *Angew. Chem. Int. Ed. Engl.* **2010**, *49*, 7076–7080, DOI 10.1002/anie.201002590.

[16] A. J. Parrott, R. A. Bourne, G. R. Akien, D. J. Irvine, M. Poliakoff, *Angew. Chem. Int. Ed. Engl.* **2011**, *50*, 3788–3792, DOI 10.1002/anie.201100412.

[17] V. Sans, L. Porwol, V. Dragone, L. Cronin, *Chem. Sci.* **2015**, *6*, 1258–1264, DOI 10.1039/C4SC03075C.

[18] D. Cortés-Borda, K. V. Kutonova, C. Jamet, M. E. Trusova, F. Zammattio, C. Truchet, M. Rodriguez-Zubiri, F.-X. Felpin, *Org. Process Res. Dev.* **2016**, *20*, 1979–1987, DOI 10.1021/acs.oprd.6b00310.

[19] K. Poscharny, D. C. Fabry, S. Heddrich, E. Sugiono, M. A. Liauw, M. Rueping, *Tetrahedron* **2018**, *74*, 3171–3175, DOI 10.1016/j.tet.2018.04.019.

[20] N. Holmes, G. R. Akien, R. J. D. Savage, C. Stanetty, I. R. Baxendale, A. J. Blacker, B. A. Taylor, R. L. Woodward, R. E. Meadows, R. A. Bourne, *React. Chem. Eng.* **2016**, *1*, 96–100, DOI 10.1039/C5RE00083A.

[21] N. Holmes, G. R. Akien, A. J. Blacker, R. L. Woodward, R. E. Meadows, R. A. Bourne, *React. Chem. Eng.* **2016**, *1*, 366–371, DOI 10.1039/C6RE00059B.

[22] A.-C. Bédard, A. Adamo, K. C. Aroh, M. G. Russell, A. A. Bedermann, J. Torosian, B. Yue, K. F. Jensen, T. F. Jamison, *Science* **2018**, *361*, 1220–1225, DOI 10.1126/science.aat0650.

[23] N. Cherkasov, Y. Bai, A. J. Expósito, E. V. Rebrov, *React. Chem. Eng.* **2018**, *3*, 769–780, DOI 10.1039/C8RE00046H.

[24] M. I. Jeraal, N. Holmes, G. R. Akien, R. A. Bourne, *Tetrahedron* **2018**, *74*, 3158–3164, DOI 10.1016/J.TET.2018.02.061.

[25] A. M. Schweidtmann, A. D. Clayton, N. Holmes, E. Bradford, R. A. Bourne, A. A. Lapkin, *Chem. Eng. J.* **2018**, *352*, 277–282, DOI 10.1016/J.CEJ.2018.07.031.

[26] Y. Amar, A. M. Schweidtmann, P. Deutsch, L. Cao, A. Lapkin, *Chem. Sci.* **2019**, DOI 10.1039/C9SC01844A.

[27] A. D. Clayton, A. M. Schweidtmann, G. Clemens, J. A. Manson, C. J. Taylor, C. G. Niño, T. W. Chamberlain, N. Kapur, A. J. Blacker, A. A. Lapkin, R. A. Bourne, *Chem. Eng. J.* **2020**, *384*, 123340, DOI 10.1016/j.cej.2019.123340.

[28] Z. Zhou, X. Li, R. N. Zare, *ACS Cent. Sci.* **2017**, *3*, 1337–1344, DOI 10.1021/acscentsci.7b00492.

[29] P. Virtanen, R. Gommers, T. E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, E. Burovski, P. Peterson, W. Weckesser, J. Bright, S. J. van der Walt, M. Brett, J. Wilson, K. Jarrod Millman, N. Mayorov, A. R. J. Nelson, E. Jones, R. Kern, E. Larson, C. Carey, İ. Polat, Y. Feng, E. W. Moore, J. Vand erPlas, D. Laxalde, J. Perktold, R. Cimrman, I. Henriksen, E. A. Quintero, C. R. Harris, A. M. Archibald, A. H. Ribeiro, F. Pedregosa, P. van Mulbregt, S. 1. 0. Contributors, *Nat. Methods* **2020**, *17*, 261–272, DOI `https://doi.org/10.1038/s41592-019-0686-2`.

[30] B. Shahriari, K. Swersky, Z. Wang, R. P. Adams, N. de Freitas, *Proceedings of the IEEE* **2016**, *104*, 148–175.

[31] The GPyOpt authors, GPyOpt: A Bayesian Optimization framework in python, `http://github.com/SheffieldML/GPyOpt`, **2016**.

[32] K. Deb, A. Pratap, S. Agarwal, T. Meyarivan, *IEEE Trans. Evol. Comput.* **2002**, *6*, 182–197, DOI `10.1109/4235.996017`.

[33] J. Blank, K. Deb, *IEEE Access* **2020**, *8*, 89497–89509, DOI `10.1109/ACCESS.2020.2990567`.

[34] F. Häse, L. M. Roch, A. Aspuru-Guzik, *arXiv preprint* **2020**, arXiv: 2003.12127.

[35] F. Häse, L. M. Roch, C. Kreisbeck, A. Aspuru-Guzik, *ACS Cent. Sci.* **2018**, *4*, 1134–1145, DOI `10.1021/acscentsci.8b00307`.

[36] R. S. Sutton, A. G. Barto, *Reinforcement learning: An introduction*, MIT press, **2018**.

[37] F. Häse, L. M. Roch, A. Aspuru-Guzik, *Chem. Sci.* **2018**, *9*, 7642–7655, DOI `10.1039/c8sc02239a`.

[38] C. A. Hone, N. Holmes, G. R. Akien, R. A. Bourne, F. L. Muller, *React. Chem. Eng.* **2017**, *2*, 103–108, DOI `10.1039/C6RE00109B`.

[39] L. M. Baumgartner, J. M. Dennis, N. A. White, S. L. Buchwald, K. F. Jensen, *Org. Process Res. Dev.* **2019**, *23*, 1594–1601, DOI `10.1021/acs.oprd.9b00236`.

[40] C. Loschen, A. Klamt, *Ind. Eng. Chem. Res.* **2012**, *51*, 14303–14308, DOI `10.1021/ie3023675`.

[41] A. M. Zissimos, M. H. Abraham, A. Klamt, F. Eckert, J. Wood, *J. Chem. Inf. Comput. Sci.* **2002**, *42*, 1320–1331, DOI `10.1021/ci025530o`.

[42] P. Esposito, BLITZ - Bayesian Layers in Torch Zoo (a Bayesian Deep Learning library for Torch), `https://github.com/piEsposito/blitz-bayesian-deep-learning/`, **2020**.