






Cite this: *Chem. Sci.*, 2021, 12, 14792

All publication charges for this article have been paid for by the Royal Society of Chemistry

# Golem: an algorithm for robust experiment and process optimization†

Matteo Aldeghi, \*<sup>abc</sup> Florian Häse, <sup>abcd</sup> Riley J. Hickman, <sup>bc</sup> Isaac Tamblyn <sup>ae</sup> and Alán Aspuru-Guzik \*<sup>abcf</sup>

Numerous challenges in science and engineering can be framed as optimization tasks, including the maximization of reaction yields, the optimization of molecular and materials properties, and the fine-tuning of automated hardware protocols. Design of experiment and optimization algorithms are often adopted to solve these tasks efficiently. Increasingly, these experiment planning strategies are coupled with automated hardware to enable autonomous experimental platforms. The vast majority of the strategies used, however, do not consider robustness against the variability of experiment and process conditions. In fact, it is generally assumed that these parameters are exact and reproducible. Yet some experiments may have considerable noise associated with some of their conditions, and process parameters optimized under precise control may be applied in the future under variable operating conditions. In either scenario, the optimal solutions found might not be robust against input variability, affecting the reproducibility of results and returning suboptimal performance in practice. Here, we introduce Golem, an algorithm that is agnostic to the choice of experiment planning strategy and that enables robust experiment and process optimization. Golem identifies optimal solutions that are robust to input uncertainty, thus ensuring the reproducible performance of optimized experimental protocols and processes. It can be used to analyze the robustness of past experiments, or to guide experiment planning algorithms toward robust solutions on the fly. We assess the performance and domain of applicability of Golem through extensive benchmark studies and demonstrate its practical relevance by optimizing an analytical chemistry protocol under the presence of significant noise in its experimental conditions.

Received 17th March 2021  
Accepted 11th October 2021

DOI: 10.1039/d1sc01545a

rsc.li/chemical-science

## 1. Introduction

Optimization problems, in which one seeks a set of parameters that maximize or minimize an objective of interest, are ubiquitous across science and engineering. In chemistry, these parameters may be the experimental conditions that control the yield of the reaction, or those that determine the cost-efficiency of a manufacturing process (*e.g.*, temperature, time, solvent, catalyst).<sup>1,2</sup> The design of molecules and materials with specific properties is also a multi-parameter, multi-objective optimization problem, with their chemical composition ultimately

governing their properties.<sup>3–7</sup> These optimization tasks may, in principle, be performed autonomously. In fact, thanks to ever-growing automation, machine learning (ML)-driven experimentation has attracted considerable interest.<sup>8–14</sup> Self-driving laboratories are already accelerating the rate at which these problems can be solved by combining automated hardware with ML algorithms equipped with optimal decision-making capabilities.<sup>15–21</sup>

Recent efforts in algorithm development have focused on providing solutions to the requirements that arise from the practical application of self-driving laboratories. For instance, newly proposed algorithms include those with favorable computational scaling properties,<sup>22</sup> with the ability to optimize multiple objectives concurrently,<sup>23</sup> that are able to handle categorical variables (such as molecules) and integrate external information into the optimization process.<sup>24</sup> One practical requirement of self-driving laboratories that has received little attention in this context is that of robustness against variability of experimental conditions and process parameters.

During an optimization campaign, it is typically assumed that the experimental conditions are known and exactly reproducible. However, the hardware (*e.g.*, dispensers, thermostats)

<sup>a</sup>Vector Institute for Artificial Intelligence, Toronto, ON, Canada. E-mail: [matteo.aldeghi@vectorinstitute.ai](mailto:matteo.aldeghi@vectorinstitute.ai); [alan@aspuru.com](mailto:alan@aspuru.com)

<sup>b</sup>Chemical Physics Theory Group, Department of Chemistry, University of Toronto, Toronto, ON, Canada

<sup>c</sup>Department of Computer Science, University of Toronto, Toronto, ON, Canada

<sup>d</sup>Department of Chemistry and Chemical Biology, Harvard University, Cambridge, MA, USA

<sup>e</sup>National Research Council of Canada, Ottawa, ON, Canada

<sup>f</sup>Lebovic Fellow, Canadian Institute for Advanced Research, Toronto, ON, Canada

† Electronic supplementary information (ESI) available. See DOI: 10.1039/d1sc01545a

may impose limitations on the precision of the experimental procedure such that there is a stochastic error associated with some or all conditions. As a consequence, the optimal solution found might not be robust to perturbations of the inputs, affecting the reproducibility of the results and returning suboptimal performance in practice. Another scenario is when a process optimized under precise control is to be adopted in the future under looser operating conditions. For instance, in large-scale manufacturing, it might not be desirable (or possible) to impose tight operating ranges on the process parameters due to the cost of achieving high precision. This means that the tightly controlled input parameters used during optimization might not reflect the true, variable operating conditions that will be encountered in production.

In general, it is possible to identify two main types of input variability encountered in an experimental setting. The first is due to uncertainty in the experimental conditions that are controlled by the researchers, often referred to as the *control factors*, corresponding to the examples discussed above. It can be caused by the imprecision of the instrumentation, which may reflect a fundamental limitation or a design choice, and could affect the present or future executions of the experimental protocol. A second type of input variability that can affect the performance of the optimization is due to experimental conditions that the researcher does not directly control. This may be, for instance, the temperature or the humidity of the room in which the experiments are being carried out. While it might not always be possible or desirable to control these conditions, they might be known and monitored such that their impact on the experimental outcome can in principle be accounted for.<sup>25</sup> The work presented here focuses on the first type of variability, related to control factors, although the approach presented may be in principle extended and applied to environmental factors too.

Here, we introduce Golem, a probabilistic approach that identifies optimal solutions that are robust to input uncertainty, thus ensuring the reproducible performance of optimized experiments and processes. Golem accounts for sources of uncertainty and may be applied to reweight the merits of previous experiments, or integrated into popular optimization algorithms to directly guide the optimization toward robust solutions. In fact, the approach is agnostic to the choice of experiment planning strategy and can be used in conjunction with both design of experiment and optimization algorithms. To achieve this, Golem explicitly models experimental uncertainty with suitable probability distributions that refine the merits of the collected measurements. This allows one to define an objective function that maximizes the average performance under variable conditions, while optionally also penalizing the expected variance of the results.

The article is organized as follows. First, we review some background information and previous work on robust optimization (Section II). Second, we introduce the core ideas behind the Golem algorithm (Section III). We then present the analytical benchmark functions used to test Golem together with different optimization approaches (Section IV), as well as the results of these benchmark studies (Section V). Finally, we show

how Golem may be used in practice, taking the calibration of a high-performance liquid chromatography (HPLC) protocol as an example application (Section VI).

## II. Background and related work

Formally, an optimization task requires finding the set of conditions  $x$  (i.e., the *parameters*, or *control factors*) that yield the most desirable outcome for  $f(x)$ . If the most desirable outcome is the one that minimizes  $f(x)$ , then the solution of the optimization problem is

$$x^* = \arg \min_{x \in \mathcal{X}} f(x), \quad (1)$$

where  $\mathcal{X}$  is the domain of the optimization defining the range of experimental conditions that are feasible or that one is willing to consider. The objective function value  $f(x)$  determines the *merit* of a specific set of parameters  $x$ . This merit may reflect the yield of a reaction, the cost-efficiency of a manufacturing process, or a property of interest for a molecule or material. Note that the objective function  $f(x)$  is *a priori* unknown, but can be probed *via* experiment. Only a finite number  $K$  of samples  $\mathcal{D}_K = \{x, f(x)\}_{k=1}^K$  are typically collected during an optimization campaign, due to the cost and time of performing the experiments. A *surrogate* model of  $f(x)$  can be constructed based on  $\mathcal{D}_K$ . This model is typically a statistical or machine learning (ML) model that captures linear and non-linear relationships between the input conditions  $x$  and the objective function values  $f(x)$ .

An optimization campaign thus typically proceeds by iteratively testing sets of parameters  $x$ , as defined *via* a design of experiment or as suggested by an experiment planning algorithm.<sup>26–28</sup> Common design of experiment approaches rely on random or systematic searches of parameter combinations. Other experiment planning algorithms include sequential model-based approaches, such as Bayesian optimization,<sup>29,30</sup> and heuristic approaches like evolutionary and genetic algorithms.<sup>31–33</sup> Experiment planning algorithms are now of particular interest in the context of self-driving laboratories for chemistry and materials science,<sup>18,19,22,34,35</sup> which aim to autonomously and efficiently optimize the properties of molecules and materials.

### A. Robust optimization

The goal of *robust* optimization is to identify solutions to an optimization problem that are robust to variation or sources of uncertainty in the conditions under which the experiments are or will be performed.<sup>36</sup> Robustness may be sought for different reasons. For instance, the true location in parameter space of the query points being evaluated might be uncertain if experiments are carried out with imprecise instruments. In another scenario, a process might be developed in a tightly controlled experimental setting, however, it is expected that future execution of the same protocol will not. In such cases, a solution that is insensitive to the variability of the experimental conditions is desirable.



Several unique approaches have been developed for this purpose, originating with the robust design methodology of Taguchi, later refined by Box and others.<sup>36,37</sup> Currently, the most common approaches rely on either a *deterministic* or *probabilistic* treatment of input parameter uncertainty. Note that, by *robust optimization*, and with chemistry applications in mind, we broadly refer to any approach aiming at solutions that mitigate the effects of the variability of experimental conditions. In the literature, the same term is sometimes used to specifically refer to what we are here referring to as *deterministic* approaches.<sup>36,38</sup> At the same time, the term *stochastic optimization*<sup>39,40</sup> is often used to refer to approaches that here we describe as *probabilistic*. We also note that, while being separate fields, many similarities with robust control theory are present.<sup>41</sup> The lack of a unified nomenclature is the result of robust optimization problems arising in different fields of science and engineering, from operations research to robotics, finance, and medicine, each with their own sets of unique challenges. While a detailed review of all robust optimization approaches developed to date is out of the scope of this brief introductory section, we refer the interested reader to more comprehensive appraisals by Beyer,<sup>36</sup> Bertsimas,<sup>38</sup> and Powell.<sup>40</sup> In the interest of conciseness, we also do not discuss approaches based on fuzzy sets<sup>42,43</sup> and those based on the minimization of risk measures.<sup>44,45</sup>

Deterministic approaches define robustness with respect to an uncertainty set.<sup>46,47</sup> Given the objective function  $f(x)$ , the robust counterpart  $g(x)$  is defined as

$$g(x) \equiv \sup_{z \in \mathcal{U}(x, \delta)} f(z), \quad (2)$$

where  $\mathcal{U}$  is an area of parameter space in the neighborhood of  $x$ , the size of which is determined by  $\delta$ .  $g(x)$  then takes the place of  $f(x)$  in the optimization problem. This approach corresponds to optimizing for a worst-case scenario, since the robust merit is defined as the worst (*i.e.*, maximum, in minimization tasks) value of  $f(x)$  in the neighborhood of  $x$ . Despite being computationally attractive, this approach is generally conservative and can result in robust solutions with poor average performance.<sup>36</sup>

A different way to approach the problem is to treat input parameters probabilistically as random variables. Probability distributions for input parameters can be defined assuming knowledge about the uncertainty or expected variability of the experimental conditions.<sup>36</sup> In this case, the objective function  $f(x)$  becomes a random quantity itself, with its own (unknown) probability density (Fig. 1a). The robust counterpart of  $f(x)$  can then be defined as its expectation value,

$$g(x) \equiv \mathbb{E}[f(\tilde{x})] = \int f(\tilde{x})p(\tilde{x})d\tilde{x}. \quad (3)$$

Here,  $\tilde{x} = x + \delta$ , where  $\delta$  is a random variable with probability density  $p(\delta)$ , which represents the uncertainty of the input conditions at  $x$  (see Section S.1† for a different, but equivalent formulation). This definition ensures that the solution of the robust optimization problem is average-case optimal. For example, assume  $f(x)$  is the yield of a reaction given the reaction

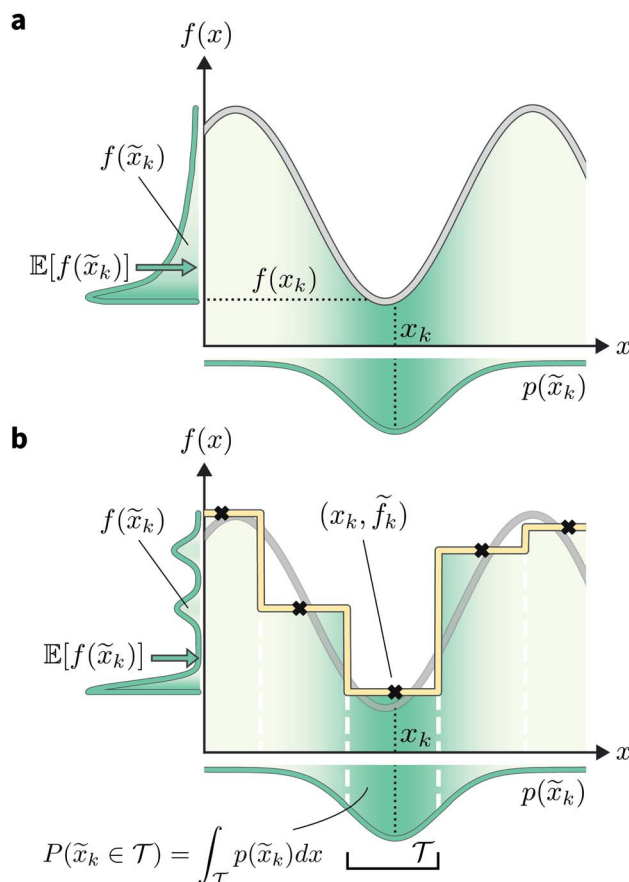


Fig. 1 Golem's approach to estimating robustness. (a) Effect of uncertain inputs on objective function evaluations. The true objective function is shown as a gray line. The probability distribution  $p(\tilde{x}_k)$  of possible input value realizations for the targeted location  $x_k$  is shown in green, below the  $x$ -axis. The distribution of output  $f(\tilde{x}_k)$  values caused by the input uncertainty are similarly shown next to the  $y$ -axis. The expectation of  $f(\tilde{x}_k)$  is indicated by a green arrow. (b) Schematic of Golem's core concept. The yellow line represents the surrogate function used to model the underlying objective function, shown in the background as a gray line. This surrogate is built with a regression tree, trained on five observations (black crosses). Note how the observations  $\tilde{f}_k$  are noisy, due to the uncertainty in the location of the input queries. In the noiseless query setting, and assuming no measurement error, the observations would lie exactly on the underlying objective function. Vertical white, dashed lines indicate how this model has partitioned the one-dimensional input space. Given a target location  $x_k$ , the probability that the realized input was obtained from partition  $\mathcal{T}$  can be computed by integrating the probability density  $p(\tilde{x}_k)$  over  $\mathcal{T}$ , which is available analytically.

conditions  $x$ . However, we know the optimized protocol will be used multiple times in the future without carefully monitoring the experimental conditions. By optimizing  $g(x)$  as defined above, instead of  $f(x)$ , and assuming that  $p(\tilde{x})$  captures the variability of future experimental conditions correctly, one can identify a set of experimental conditions that returns the best possible yield on average across multiple repeated experiments.

Despite its attractiveness, the probabilistic approach to robust optimization presents computational challenges. In fact,

the above expectation cannot be computed analytically for most combinations of  $f(x)$  and  $p(\tilde{x})$ . One solution is to approximate  $\mathbb{E}[f(\tilde{x})]$  by numerical integration, using quadrature or sampling approaches.<sup>48–50</sup> However, this strategy can become computationally expensive as the dimensionality of the problem increases and if  $g(x)$  is to be computed for many samples. As an alternative numerical approach, it has been proposed to use a small number of carefully chosen points in  $x$  to cheaply approximate the integral.<sup>51</sup> Selecting optimal points for arbitrary probability distributions is not straightforward, however.<sup>52</sup>

In Bayesian optimization, it is common to use Gaussian process (GP) regression to build a surrogate model of the objective function. A few approaches have been proposed in this context to handle input uncertainty.<sup>53,54</sup> Most recently, Fröhlich *et al.*<sup>55</sup> have introduced an acquisition function for GP-based Bayesian optimization for the identification of robust optima. This formulation is analytically intractable and the authors propose two numerical approximation schemes. A similar approach was previously proposed by Beland and Nair.<sup>56</sup> However, in its traditional formulation, GP regression scales cubically with the number of samples collected. In practice, this means that optimizing  $g(x)$  can become costly after collecting more than a few hundred samples. In addition, GPs do not inherently handle discrete or categorical variables<sup>57</sup> (e.g., type of catalyst), which are often encountered in practical chemical research. Finally, these approaches generally assume normally distributed input noise, as this tends to simplify the problem formulation. However, physical constraints on the experimental conditions may cause input uncertainty to deviate from this scenario, such that it would be preferable to be able to model any possible noise distribution.

In this work, we propose a simple, inexpensive, and flexible approach to probabilistic robust optimization. Golem enables the accurate modeling of experimental conditions and their variability for continuous, discrete, and categorical conditions, and for any (parametric) bounded or unbounded uncertainty distribution. By decoupling the estimation of the robust objective  $g(x)$  from the details of the optimization algorithm, Golem can be used with any experiment planning strategy, from design of experiment, to evolutionary and Bayesian optimization approaches.

### III. Formulating golem

Consider a robust optimization problem in which the goal is to find a set of input conditions  $x \in \mathcal{X}$  corresponding to the global minimum of the function  $g : \mathcal{X} \rightarrow \mathbb{R}$ ,

$$x^* = \arg \min_{x \in \mathcal{X}} g(x). \quad (4)$$

We refer to  $g(x)$ , as defined in eqn (3), as the robust objective function, while noting that other integrated measures of robustness may also be defined.

Assume a sequential optimization in which we query a set of conditions  $x_k$  at each iteration  $k$ . If the input conditions are noiseless, we can evaluate the objective function at  $x_k$  (denoted  $f_k$ ). After  $K$  iterations, we will have built a dataset

$\mathcal{D}_K = \{x_k, f_k\}_{k=1}^K$ . However, if the input conditions are noisy, the realized conditions will be  $\tilde{x}_k = x_k + \delta$ , where  $\delta$  is a random variable. As a consequence, we incur stochastic evaluations of the objective function, which we denote  $\tilde{f}_k$ . This is illustrated in Fig. 1a, where the Gaussian uncertainty in the inputs results in a broad distribution of possible output values. In this case, we will have built a dataset  $\tilde{\mathcal{D}} = \{x_k, \tilde{f}_k\}_{k=1}^K$ . Note that, while  $\tilde{x}_k$  generally refers to a random variable, when considered as part of a dataset  $\tilde{\mathcal{D}}$  it may be interpreted as a specific sample of such variable. Hence, for added clarity, in Fig. 1 we refer to the distributions on the y-axis as  $f(\tilde{x}_k)$ , while we refer to function evaluations on specific input values as  $\tilde{f}_k$ .

#### A. General formalism

The goal of Golem is to provide a simple and efficient means to estimate  $g(x)$  from the available data,  $\mathcal{D}_K$  or  $\tilde{\mathcal{D}}_K$ . This would allow us to create a dataset  $\mathcal{G}_K = \{x_k, g_k\}_{k=1}^K$  with robust merits, which can then be used to solve the robust optimization task in eqn (4). To do this, a surrogate model of the underlying objective function  $f(x)$  is needed. This model should be able to capture complex, non-linear relationships. In addition, it should be computationally cheap to train and evaluate, and be scalable to high-data regimes. At the same time, we would like to flexibly model  $p(\tilde{x})$ , such that it can satisfy physical constraints and closely approximate the true experimental uncertainty. At the core of Golem is the simple observation that when approximating  $f(x)$  with tree-based ML models, such as regression trees and random forest, estimates of  $g(x)$  can be computed analytically as a finite series for any parametric probability density  $p(\tilde{x})$ . A detailed derivation can be found in Section S.1.†

An intuitive depiction of Golem is shown in Fig. 1b. Tree-based models are piece-wise constant and rely on the rectangular partitioning of input space. Because of this discretization,  $\mathbb{E}[f(x)]$  can be obtained as a constant contribution from each partition  $\mathcal{T}$ , weighted by the probability of  $x$  being within each partition,  $P(x_k \in \mathcal{T})$ . Hence, an estimate of  $g(x)$  can be efficiently obtained as a sum over all partitions (eqn (20)†).

Tree-based models such as regression trees and random forests have a number of advantages that make them well-suited for this task. First, they are non-linear ML models that have proved to be powerful function approximators. Second, they are fast to train and evaluate, adding little overhead to the computational protocols used. In the case of sequential optimization, the dataset  $\mathcal{D}_K$  grows at each iteration  $k$ , such that the model needs to be continuously re-trained. Finally, they can naturally handle continuous, discrete, and categorical variables, so that uncertainty in all type of input conditions can be modeled. These reasons in addition to the fact that tree-based models allow for a closed-form solution to eqn (3) make Golem a simple yet effective approach for robust optimization. Note that while we decouple Golem's formulation from any specific optimization algorithm in this work, it is in principle possible to integrate this approach into tree-ensemble Bayesian optimization algorithms.<sup>58,59</sup> This can be achieved *via* an acquisition function that is based on Golem's estimate of the





robust objective, as well as its uncertainty, which can be estimated from the variance of  $g(x)$  across trees.

Fig. 2 shows a simple, one-dimensional example to provide intuition for Golem's behavior. In the top panel, the robust objective function is shown for different levels of normally-distributed input noise, parameterized by the standard deviation  $\sigma(x)$  reported. Note that, when there is no uncertainty and  $\sigma(x) = 0$  (gray line),  $p(\tilde{x})$  is a delta function and one recovers the original objective function. As the uncertainty increases, the global minimum of the robust objective shifts from being the one at  $x \approx 0.15$  to that at  $x \approx 0.7$ . In the two panels at the bottom, the same effect is shown under a realistic low-data scenario, in which only a few observations of the objective function are available (gray circles). Here, the dashed gray line represents the surrogate model used by Golem to estimate the robustness of each solution, given low (bottom left, green circles) and high (bottom right, blue circles) input noise. As in the top panel, which shows the continuous ground truth, here too the left-hand-side minimum is favored until the input noise is large enough such that the right-hand-side minimum provides better average-case performance.

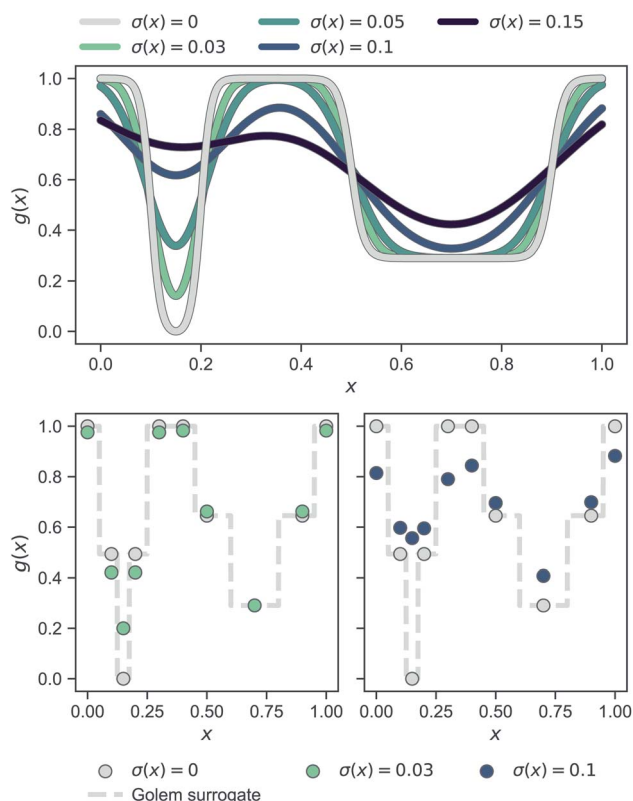


Fig. 2 One-dimensional example illustrating the probabilistic approach to robustness and Golem's behavior. The top panel shows how  $g(x)$ , which is defined as  $\mathbb{E}[f(x)] = \int f(x)p(\tilde{x})dx$ , changes as the standard deviation of normally-distributed input noise  $p(\tilde{x})$  is increased. Note that the curve for  $\sigma(x) = 0$  corresponds to the original objective function. The panels at the bottom show the robust merits of a finite set of samples as estimated by Golem from the objective function values.

## B. Multi-objective optimization

When experimental noise is present, optimizing for the robust objective might not be the only goal. Often, large variance in the outcomes of an experimental procedure is undesirable, such that one might want to minimize it. For instance, in a chemical manufacturing scenario, one would like to ensure maximum overall output across multiple plants and batches. However, it would also be important that the amount of product manufactured in each batch does not vary considerably. Thus, the optimal set of manufacturing conditions should not only provide high yields on average, but also consistent ones. The problem can thus be framed as a multi-objective optimization in which we would like to maximize  $\mathbb{E}[f(x)]$  while minimizing  $\sigma[f(x)] = \text{Var}[f(x)]^{1/2}$ . Golem can also estimate  $\sigma[f(x)]$  (Section S.1.D†), enabling such multi-objective optimizations. With  $\mathbb{E}[f(x)]$  and  $\sigma[f(x)]$  available, any scalarizing function may be used, including weighted sums and rank-based algorithms.<sup>23</sup>

## IV. Benchmark surfaces and basic usage

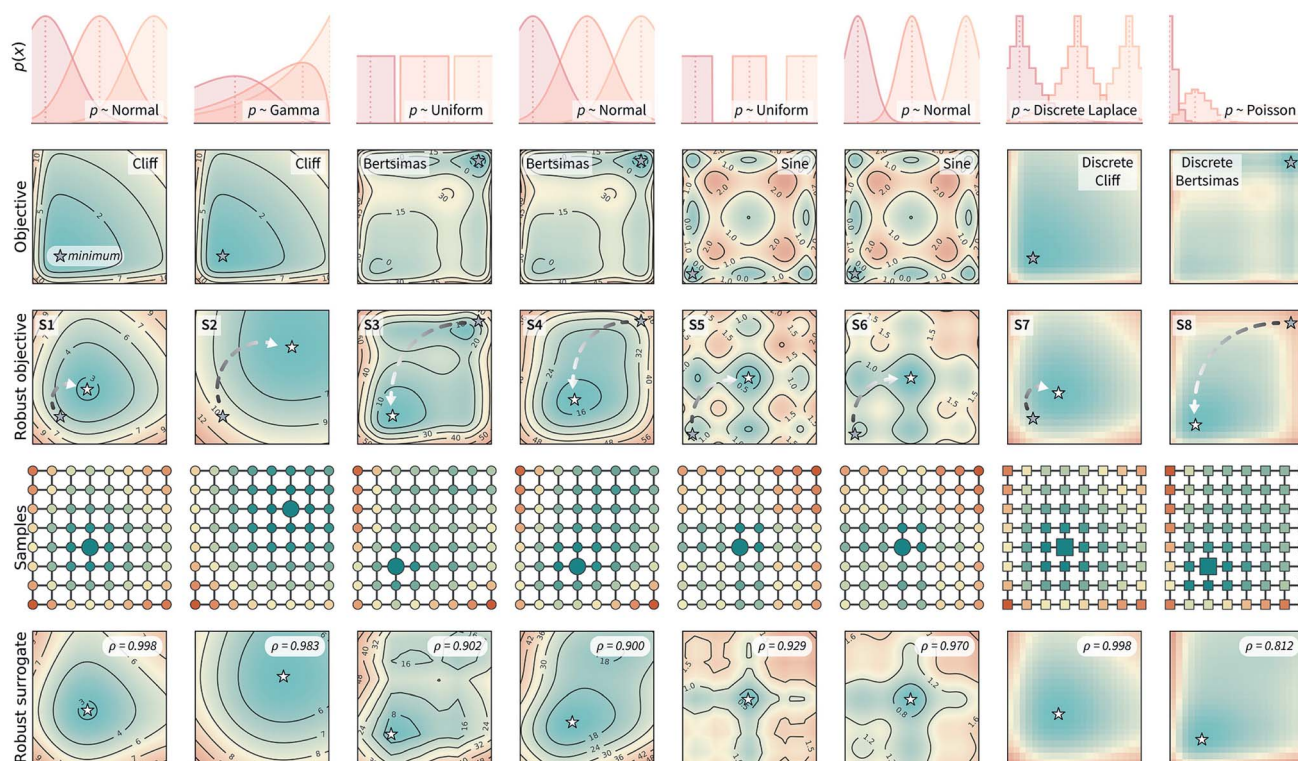
The performance of Golem, in conjunction with a number of popular optimization algorithms, was evaluated on a set of two-dimensional analytical benchmark functions. This allowed us to test the performance of the approach under different, hypothetical scenarios, test which optimization algorithms are most suited to be combined with Golem, and demonstrate the ways in which Golem may be deployed.

### A. Overview of the benchmark surfaces

Fig. 3 shows the benchmark functions that were used to evaluate Golem. These benchmarks were chosen to both challenge the algorithm and show its flexibility. We selected both continuous and discrete surfaces, and bounded and unbounded probability distributions to describe the input uncertainty. The objective functions considered are shown in the second row of Fig. 3. The *Bertsimas* function is taken from the work of Bertsimas *et al.*,<sup>46</sup> while *Cliff* and *Sine* are introduced in this work (Section S.2.A†). The first row of Fig. 3 shows the uncertainty applied to these objective functions in both input dimensions. These uncertainties induce the robust objective functions shown in the third row. The location of the global minimum is shown for each objective and robust objective, highlighting how the location of the global minimum is affected by the variability of the inputs. The eight robust objectives in the third row of Fig. 3 are labeled S1 to S8 and are the surfaces to be optimized. While we can only probe the objective functions in the second row, we use Golem to estimate their robust counterparts in the third row and locate their global minima.

These synthetic functions challenge Golem and the optimization algorithms in different ways. The rougher the surface and its robust counterpart, the more challenging it is expected to be to optimize. The smaller the difference in robust merit between the non-robust and robust minima (Section S.2.A, Table S1†), the harder it is for Golem to resolve the location of the true robust minimum, as more accurate estimates of  $g(x)$  are





**Fig. 3** Benchmark functions used to test Golem and its performance. The first two rows show the type of uncertainty (in both input dimensions) assumed and the objective functions used in the synthetic benchmarks. The location of the global minimum is marked by a gray star on the two-dimensional surface of each objective function. The third row shows how the input uncertainty transforms each objective function into its robust counterpart. These surfaces (referred to as S1 to S8) represent the robust objectives, which are not directly observable, but that we would like to optimize. The global minimum of these functions are marked by white stars, with an arrow indicating the shift in the location of the global minimum between non-robust and robust objectives. The fourth row shows a set of  $8 \times 8$  samples that have been collected from these surfaces. Each sample is colored by its robust merit as estimated by Golem using only these 64 samples. The larger marker (circle or square, for continuous and discrete surfaces, respectively) indicate the sample with best estimated robust merit. For all surfaces, Golem correctly estimates the most robust sample to be one in the vicinity of the true global minimum. The final row shows Golem's surrogate model of the robust objective, constructed from the grid of 64 samples shown in row four. This surrogate model is highly correlated with the true underlying robust objective, as indicated by Spearman's correlation coefficient ( $\rho$ ) reported at the top-right corner of each plot.

required. Finally, the steeper the objective function is outside the optimization domain, the less accurate Golem's estimate will be close to the optimization boundary, as samples are collected only within the optimization domain.

S1–S6 evaluate performance on continuous spaces, while S7 and S8 on discrete ones. The function denoted *Cliff* has a single minimum, which is shifted in the robust objectives S1 and S2. The *Bertsimas* function has a global minimum indicated at the top-right corner of the surface, and a broader minimum at the bottom-left corner. The latter is the global minimum of the robust objective functions S3 and S4. The *Sine* function is the most rugged and challenging, with nine minima (eight local and one global). S2 and S8 describe input uncertainty *via* distributions that do not allow values outside some of the bounds of the optimization domain. This is used to demonstrate Golem's flexibility and ability to satisfy physical constraints. For instance, if the uncertain input variable is dispensed volume, one should be able to assign zero probability to negative volumes.

## B. Reweighting previous results

One possible use of Golem is to reweight the merits of previously tested experimental conditions. Imagine, for instance, that we have accurately and precisely evaluated how temperature and catalyst concentration affect the yield of a reaction in the laboratory. To achieve this, we have performed 64 experiments using a uniformly spaced  $8 \times 8$  grid. Based on this data, we know which of the measured conditions provide the best yield. However, the same reaction will be used in other laboratories, or in larger-scale manufacturing, where these two variables will not be precisely controlled because, *e.g.*, precise control is expensive or requires a complex experimental setup. Therefore, we would like to reweight the merit of each of the 64 conditions previously tested, and identify which conditions are robust against variations in temperature and pressure. Golem allows one to easily compute these robust merits given the uncertainty in the input conditions. We tested Golem under this scenario and the results are shown in Fig. 3. In particular, the fourth row shows the grid of 64 samples taken from the objective function and reweighted with Golem. The color of each



sample indicates their robust merit as estimated by Golem, with blue being more robust and red less robust. The largest marker indicates the sample estimated to have the best robust merit, which is in close proximity to the location of the true robust minimum for all surfaces considered.

Based on these 64 samples, Golem can also build a surrogate model of the robust objective. This model is shown in the last row of Fig. 3. These estimates closely resemble the true robust surfaces in the third row. In fact, the Spearman's rank correlations ( $\rho$ ) between Golem's surrogates and the true robust objectives were  $\geq 0.9$  for seven out of eight surfaces tested. For S8 only, while the estimated location of the global robust minimum was still correct,  $\rho \approx 0.8$  due to boundary effects. In fact, while the robust objective depends also on the behavior of the objective function outside of the defined optimization domain, we sample the objective only within this domain. This lack of information causes the robustness estimates of points close to the boundaries to be less accurate than for those farther from them (Fig. S4†). Another consequence of this fact is that the robust surrogate does not exactly match the true robust objective also in the limit of infinite sampling within the optimization domain (Section S2.B†).

To further clarify the above statement, by “defined optimization domain” we refer to a subset of the physically-meaningful domain that the researcher has decided to consider. Imagine, for instance, that we have a liquid dispenser which we will use to dispense a certain solvent volume. The smallest volume we can dispense is zero, while the largest might be the volume in the reservoir used (e.g., 1 L). These limits are physical bounds we cannot exceed. However, for practical purposes, we will likely consider a maximum volume much smaller than the physical limit (e.g., 5 mL). In this example, 0–5 mL would constitute the defined optimization domain, while 0–1 L are physical bounds on the domain. In the context of uncertain experimental conditions, it can thus be the case that a noisy dispenser might provide 5.1 mL of liquid despite this exceeding the desired optimization boundary. The same cannot, however, be the case for the lower bound in this example, since a negative volume is physically impossible. As a consequence, while we allow an optimization algorithm to query the objective function only within the user-defined optimization domain, a noisy experimental protocol might result in the evaluation of the objective function outside of this domain.

Golem allows to take physical bounds into account by modeling input uncertainty with bounded probability distributions. Yet, it cannot prevent boundary effects that are the consequence of the unknown behaviour of the objective function outside of the defined optimization domain. This issue, unfortunately, cannot be resolved in a general fashion, as it would require a data-driven model able to extrapolate arbitrarily far from the data used for training. A practical solution may be to consider a “data collection domain” as a superset of the optimization domain, which is used for collecting data at the boundaries but which the optimization solution is not selected from. In the examples in Fig. 3 (row 4), this would mean using the datapoints on the perimeter of the two-dimensional grid only for estimating the robustness of the internal points more

accurately. We conclude by reiterating how, notwithstanding this inescapable boundary effect, as shown in Fig. 3 there is a high correlation between Golem's estimates and the true robustness values.

## V. Optimization benchmarks

With increasing levels of automation and interest in self-driving laboratories, sequential approaches that make use of all data collected to select the next, most informative experiment are becoming the methods of choice for early prototypes of autonomous science. In this case, rather than re-evaluating previously performed experiments, one would like to steer the optimization towards robust solutions during the experimental campaign. Golem allows for this in combination with popular optimization approaches, by mapping objective function evaluations onto an estimate of their robust merits at each iteration of the optimization procedure. We evaluated the ability of six different optimization approaches to identify robust solutions when used with Golem and without. The algorithms tested include three Bayesian optimization approaches (*Gryffin*,<sup>22,24</sup> *GPpyOpt*,<sup>60</sup> *Hyperopt*<sup>61</sup>), a genetic algorithm (*Genetic*),<sup>62</sup> a random sampler (*Random*), and a systematic search (*Grid*). *Gryffin*, *GPpyOpt*, and *Hyperopt* use all previously collected data to decide which set of parameters to query next, *Genetic* uses part of the collected data, while *Random* and *Grid* are totally agnostic to previous measurements.

In these benchmarks, we allowed the algorithms to collect 196 samples for continuous surfaces and 64 for the discrete ones. We repeated each optimization 50 times to collect statistics. For *Grid*, we created a set of  $14 \times 14$  uniformly-spaced samples ( $8 \times 8$  for the discrete surfaces) and then selected them at random at each iteration. For all algorithms tested, we performed the optimization with and without Golem. Algorithm performance in the absence of Golem constitutes a naïve baseline. Optimization performance is quantified using normalized cumulative robust regret, defined in S2.C.† This regret is a relative measure of how fast each algorithm identifies increasingly robust solutions, allowing the comparison of algorithm performance with respect to a specific benchmark function.

### A. Noiseless queries with uncertainty in future experiments

Here, we tested Golem under a scenario where queries during the optimization are deterministic, i.e., noiseless. It is assumed that uncertainty in the inputs will arise only in future experiments. This scenario generally applies to the development of experimental protocols that are expected to be repeated under loose control of experimental conditions.

The results of the optimization benchmarks under this scenario are summarized in Fig. 4, which shows the distributions of cumulative regrets for all algorithms considered, with and without Golem, across the eight benchmark surfaces. For each algorithm, Fig. 4 also quantifies the probability that the use of Golem resulted in better performance in the identification robust solutions. Overall, these results showed that Golem





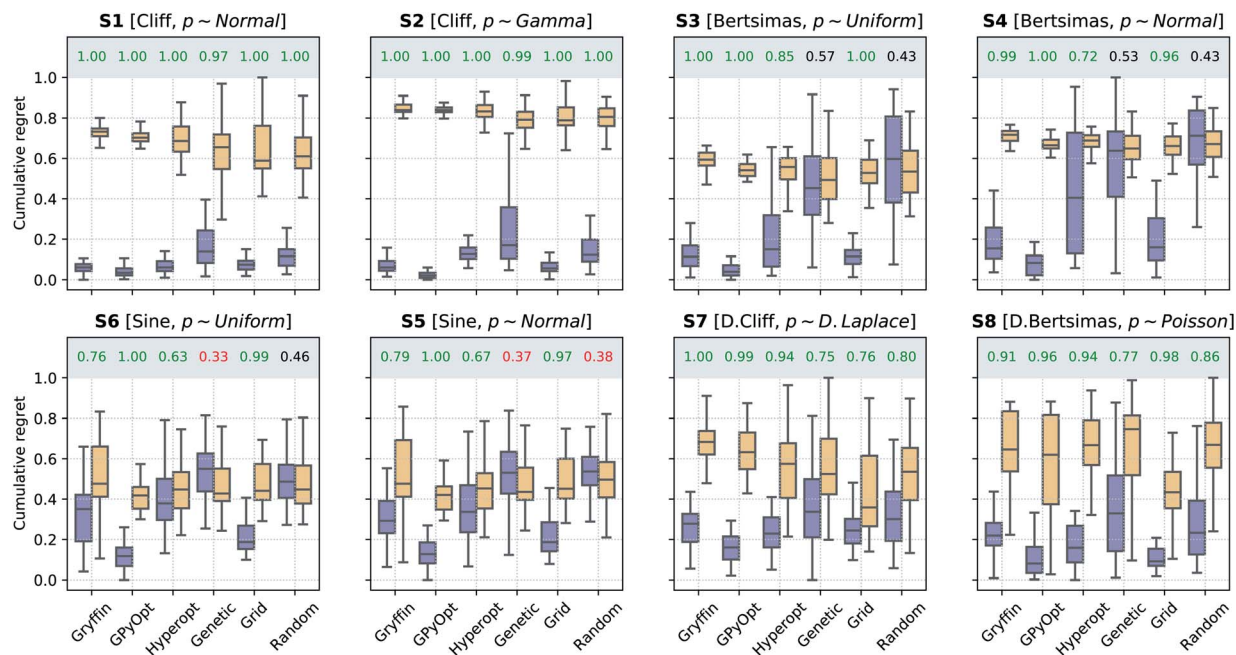


Fig. 4 Robust optimization performance of multiple algorithms, with and without Golem, in benchmarks where queries were noiseless. Box plots show the distributions of cumulative regrets obtained across 50 optimization repeats with and without Golem, in purple and yellow, respectively. The boxes show the first, second, and third quartiles of the data, with whiskers extending up to 1.5 times the interquartile range. At the top of each plot, we report the probability that the use of Golem improved upon the performance of each algorithm. Probabilities are in green if the performance with Golem was significantly better (considering a 0.05 significance level) than without, and in red if it was significantly worse, as computed by bootstrap.

allowed the optimization algorithms to identify solutions that were more robust than those identified without Golem.

A few additional trends can be extracted from Fig. 4. The Bayesian optimization algorithms (*Gryffin*, *GPyOpt*, *Hyperopt*) and systematic searches (*Grid*) seemed to benefit more from the use of Golem than genetic algorithms (*Genetic*) and random searches (*Random*). In fact, the former approaches benefited from Golem across all benchmark functions, while the latter did so only for half the benchmarks. The better performance of *Grid* as compared to *Random*, in particular, may appear surprising. We found that the main determinant of this difference is the fact that *Grid* samples the boundaries of the optimization domain, while *Random* is unlikely to do so. By forcing random to sample the optimization boundaries, we recovered performances comparable to *Grid* (Section S2.D†). We also hypothesized that uniformity of sampling might be beneficial to Golem, given that the accuracy of the robustness estimate depends on how well the objective function is modeled in the vicinity of the input location considered. We indeed found that low-discrepancy sequences provided, in some cases, slightly better performance than random sampling. However, this effect was minor compared to that of forcing the sampling of the optimization domain boundaries (Section S2.D†).

*Genetic* likely suffered from the same pathology, given it is initialized with random samples. Thus, in this context, initialization with a grid may be more appropriate. Genetic algorithms are also likely to suffer from a second effect. Given that we can only estimate the robust objective, Golem induces a history-

dependent objective function. Contrary to Bayesian optimization approaches, genetic algorithms consider only a subset of the data collected during optimization, as they discard solutions with bad fitness. Given that the robustness estimates change during the course of the optimization, these algorithms may drop promising solutions early in the search, which are then not recovered in the latter stages when Golem would have more accurately estimated their robustness. The use of more complex genetic algorithm formulations, exploring a more diverse set of possible solutions,<sup>63</sup> could improve this scenario and is a possibility left for future work.

## B. Noisy queries with uncertainty in current experiments

In a second scenario, queries during the optimization are stochastic, *i.e.*, noisy, due the presence of substantial uncertainty in the current experimental conditions. This case applies to any optimization campaign in which it is not possible to precisely control the experimental conditions. However, we assume one can model the uncertainty  $p(\vec{x})$ , at least approximately. For instance, this uncertainty might be caused by some apparatus (*e.g.*, a solid dispenser) that is imprecise, but can be calibrated and the resulting uncertainty quantified. The optimization performances of the algorithms considered, with and without Golem, are shown in Fig. 5. Note that, to model the robust objective exactly,  $p(\vec{x})$  should also be known exactly. While this is not a necessary assumption of the approach, the accuracy of Golem's estimates is proportional to the accuracy of the  $p(\vec{x})$  estimates. As the  $p(\vec{x})$  estimate provided to Golem



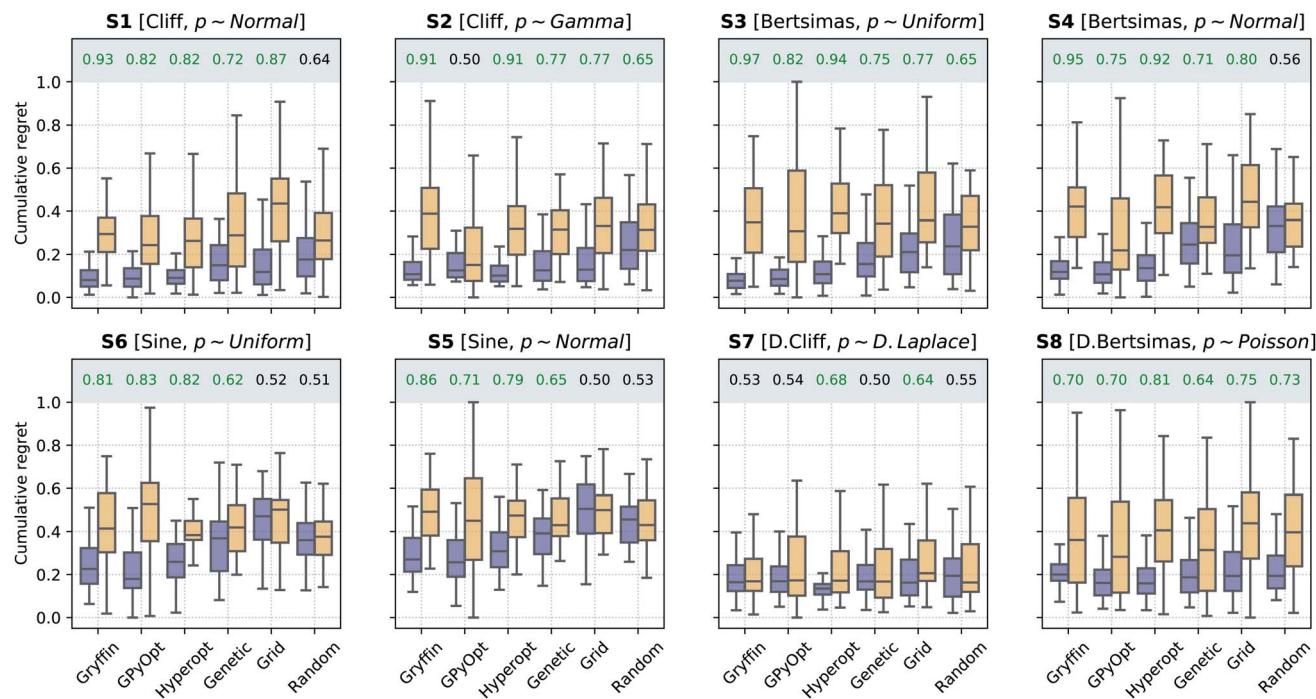


Fig. 5 Robust optimization performance of multiple algorithms, with and without Golem, in benchmarks where queries were noisy. Box plots show the distributions of cumulative regrets obtained across 50 optimization repeats with and without Golem, in purple and yellow, respectively. The boxes show the first, second, and third quartiles of the data, with whiskers extending up to 1.5 times the interquartile range. At the top of each plot, we report the probability that the use of Golem improved the performance of each algorithm. Probabilities are in green if the performance with Golem was significantly better (considering a 0.05 significance level) than without, and in red if it was significantly worse, as computed by bootstrap.

deviates from its true values, Golem under- or over-estimate the robustness of the optimal solution, depending on whether the input uncertainty is under- or over-estimated. We will illustrate this point in more detail in Section VI.A.

Generally speaking, this is a more challenging scenario than when queries are noiseless. As a consequence of the noisy experimental conditions, the dataset collected does not correctly match the realized control factors  $x$  with their associated merit  $f(x)$ . Hence, the surrogate model is likely to be a worse approximation of the underlying objective function than when queries are noiseless. While the development of ML models capable of recovering the objective function  $f(x)$  based on noisy queries  $\tilde{x}$  is outside the scope of this work, such models may enable even more accurate estimates of robustness with Golem. We are not aware of approaches capable of performing such an operation, but it is a promising direction for future research. In fact, being able to recover the (noiseless) objective function from a small number of noisy samples  $\tilde{f}$  would be beneficial not only for robustness estimation, but for the interpretation of experimental data more broadly.

Because of the above-mentioned challenge in the construction of an accurate surrogate model, in some cases, the advantage of using Golem might not seem as stark as in the noiseless setting. This effect may be seen in surfaces S1 and S2, where the separation of the cumulative regret distributions is larger in Fig. 4 than it is in Fig. 5. Nonetheless, across all benchmark functions and algorithms considered, the use of

Golem was beneficial in the identification of robust solutions in the majority of cases, and never detrimental, as shown by Fig. 5. In fact, Golem appears to be able to recover significant correlations with the true robust objectives  $g(x)$  even when correlation with the objective functions  $f(x)$  is lost due to noise the queried locations (Fig. S6†).

Optimization with noisy conditions is significantly more challenging than traditional optimization tasks with no input uncertainty. However, the synthetic benchmarks carried out suggest that Golem is able to efficiently guide optimization campaigns towards robust solutions. For example, Fig. 6 shows the location of the best input conditions as identified by GPpyOpt with and without Golem. Given the significant noise present, without Golem, the optima identified by different repeated experiments are scattered far away from the robust minimum. When Golem is used, the optima identified are considerably more clustered around the robust minimum.

### C. Effect of forest size and higher input dimensions

All results shown thus far were obtained using a single regression tree as Golem's surrogate model. However, Golem can also use tree-ensemble approaches, such as random forest<sup>64</sup> and extremely randomized trees.<sup>65</sup> We thus repeated the synthetic benchmarks discussed above using these two ML models, with forest sizes of 10, 20, and 50 (Section S2.F†). Overall, for these two-dimensional benchmarks we did not observe significant improvements when using larger forest sizes. For the



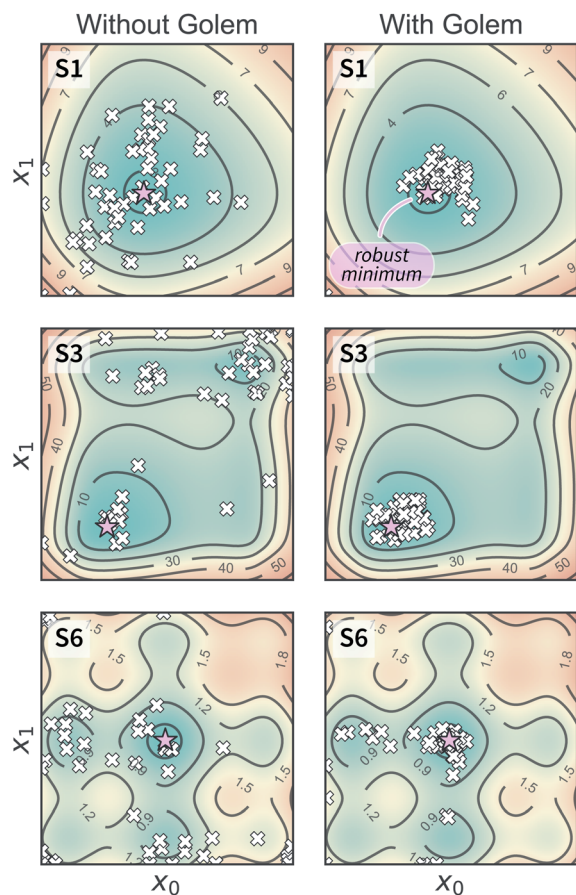


Fig. 6 Location of the optimal input parameters identified with and without Golem. The results shown were obtained with *GPyOpt* as the optimization algorithm. A pink star indicates the location of the true robust minimum. White crosses (one per optimization repeat, for a total of 50) indicate the locations of the optimal conditions identified by the algorithm without (on the left) and with (on the right) Golem.

benchmarks in the noiseless setting, regression trees appeared to provide slightly better performance against the *Bertsimas* functions (Fig. S7†). The lack of regularization may have provided a small advantage in this case, where Golem is trying to resolve subtle differences between competing minima. Yet, a single regression tree performed as well as ensembles. For the benchmarks in the noisy setting, random forest and extremely randomized trees performed slightly better overall (Fig. S8†). However, larger forests did not appear to provide considerable advantage over smaller ones, suggesting that for these low-dimensional problems, small forests or even single trees can generally be sufficient.

To study the performance of different tree-ensemble approaches also on higher-dimensional search spaces, we conducted experiments, similar to the ones described above, on three-, four-, five-, and six-dimensional versions of benchmark surface S1. In these tests, we consider two dimensions to be uncertain, while the additional dimensions are noiseless. Here, too, we studied the effect of forest type and size on the results, but we focused on the Bayesian optimization algorithms. In this case, we observed better performance of Golem when using

random forest or extremely randomized trees as the surrogate model. In the noiseless setting, extremely randomized trees returned slightly better performance than random forest, in particular for *GPyOpt* and *Hyperopt* (Fig. S9†). The correlation of optimization performance with forest size was weaker. Yet, for each combination of optimization algorithms and benchmark surface, the best overall performance was typically achieved with larger forest sizes of 20 or 50 trees. While less marked, similar trends were observed for the same tests in the noisy setting (Fig. S10†). In this scenario, random forest returned slightly better performance than extremely randomized trees for *Hyperopt*. Overall, surrogate models based on random forest or extremely randomized trees appear to provide better performance across different scenarios.

We then investigated Golem's performance across varying search space dimensionality and number of uncertain conditions. To do this, we conducted experiments on three-, four-, five-, and six-dimensional versions of benchmark surface S1, with one to six uncertain inputs. These tests showed that Golem was still able to guide the optimizations towards better robust solutions. In the noiseless setting, the performance of *GPyOpt* and *Hyperopt* was significantly better with Golem for all dimensions and number of uncertain variables tested (Fig. S11†). The performance of *Gryffin* was significantly improved by Golem in roughly half of the cases. Overall, given a certain search space dimensionality, the positive effect of Golem became more marked with a higher number of uncertain inputs. This observation does not imply that the optimization task is easier with more uncertain inputs (it is in fact more challenging), but that the use of Golem provides a more significant advantage in such scenarios. On the contrary, given a specific number of uncertain inputs, the effect of Golem was less evident with increasing number of input dimensions. Indeed, additional input dimensions make it more challenging for Golem to resolve whether the observed variability in the objective function evaluations is due to the uncertain variables or the expected behavior of the objective function along the additional dimensions. Similar overall results were observed in the noisy input setting (Fig. S12†). However, statistically significant improvements were found in a smaller fraction of cases. Here, we did not observe a significant benefit in using Golem when having a small (1–2) number of uncertain inputs, but this became more evident with a larger (3–6) number of uncertain inputs. In fact, the same trends with respect to the dimensionality of the search space and the number of uncertain inputs were observed also in the noisy query setting. One important observation is that Golem was almost never (one out of 108 tests) found to be detrimental to optimization performance, suggesting that there is very little risk in using the approach when input uncertainty is present, as in the worst-case scenario Golem would simply leave the performance of the optimization algorithm used unaltered.

Overall, these results suggest that Golem is also effective on higher-dimensional surfaces. In addition, it was found that the use of surrogate models based on forests can, in some cases, provide a better optimization performance. Given the limited computational cost of Golem, we thus generally recommend the



use of an ensemble tree method as the surrogate model. Forest sizes of 20 to 50 trees were found to be effective. Yet, given that larger ensembles will not negatively affect the estimator performance, and that the runtime scales linearly with the number of trees, larger forests may be used as well.

## VI. Chemistry applications

In this section, we provide an example application of Golem in chemistry. Specifically, we consider the calibration of an HPLC protocol, in which six controllable parameters (Fig. 7a, Section S.3†) can be varied to maximize the peak area, *i.e.*, the amount of drawn sample reaching the detector.<sup>26,66</sup> Imagine we ran 1386 experiments in which we tested combinations of these six parameters at random. The experiment with the largest peak area provides the best set of parameters found. The parameter values corresponding to this optimum are highlighted in Fig. 7b by a gray triangle pointing towards the abscissa. With the collected data, we can build a surrogate model of the response surface. The one shown as a gray line in Fig. 7b was built with 200 extremely randomized trees.<sup>65</sup> Fig. 7b shows the predicted peak area when varying each of the six controllable parameters independently around the optimum identified.

### A. Analysis of prior experimental results

Golem allows us to speculate how the expected performance of this HPLC protocol would be affected by varying levels of noise in the controllable parameters. We modeled input noise *via* truncated normal distributions that do not support values below zero. This choice satisfies the physical constraints of the experiment, given that negative volumes, flows, and times are not possible. We considered relative uncertainties corresponding to a standard deviation of 10%, 20%, and 30% of the allowed range for each input parameter. The protocol performance is most affected by uncertainty in the tubing volume (variable P3, Fig. 7b). A relative noise of 10% would result in an average peak area of around 1500 a.u., a significant drop from the maximum observed at over 2000. It follows that to achieve consistent high performance with this protocol, efforts should be spent in improving the precision of this variable.

While the protocol performance (*i.e.*, expected peak area) is least robust against uncertainty in P3, the location of the optimum setting for P3 is not particularly affected. Presence of noise in the sample loop (variable P1) has a larger effect on the location of its optimal settings. In fact, noise in P1 requires larger volumes to be drawn into the sample loop to be able to achieve average optimal responses. The optimal parameter settings for the push speed (P5) and wait time (P6) are also affected by the presence of noise. However, the protocol performance is fairly insensitive to changes in these variables, with expected peak areas of around 2000 a.u. for any of their values within the range studied.

Fig. 7 also illustrates the effect of under- or over-estimating experimental condition uncertainty on Golem's robustness estimates. Imagine that the true uncertainty in variable P3 is 20%. This may be the true uncertainty encountered in the future

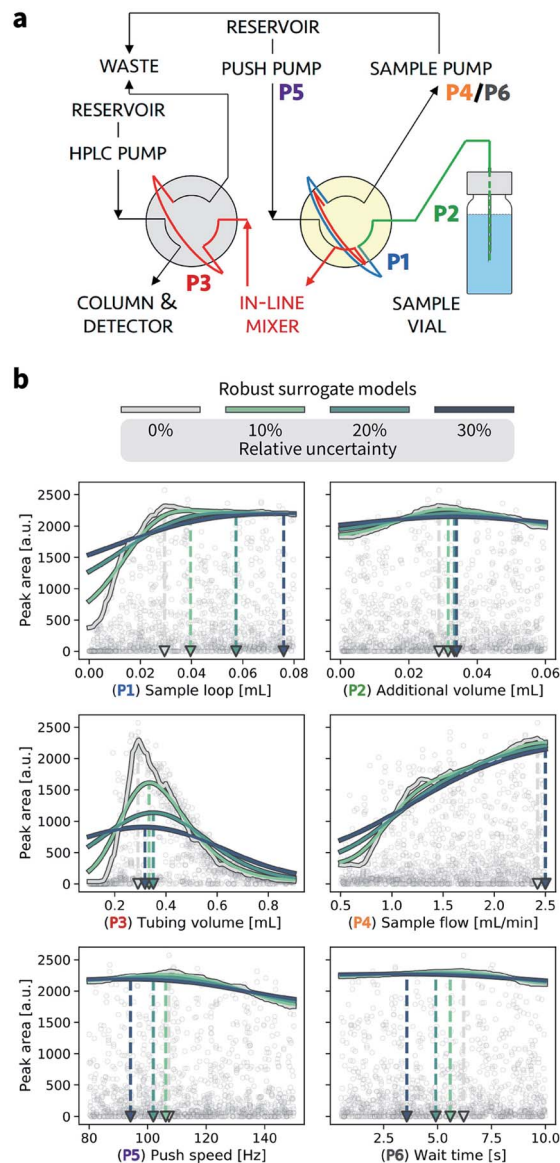


Fig. 7 Analysis of the robustness of an HPLC calibration protocol. (a) Flow path for the HPLC sampling sequence performed by a robotic platform. The six parameters (P1–P6) are color coded. The yellow shade highlights the arm valve, and the gray shade the HPLC valve. (b) Golem analysis of the effect of input noise on expected protocol performance. A surrogate model of the response surface is shown in gray. Uncertainties were modeled with truncated normal distributions with standard deviations of 10%, 20%, 30% of each parameter's range. The corresponding robust surrogate models are shown in light green, dark green, and blue. Triangular markers and dashed lines indicate the location of the optima for each parameter under different levels of noise.

deployment of the protocol, or it may be the uncertainty encountered while trying to optimize it. If we assume, incorrectly, the uncertainty to be 10%, Golem will predict the protocol to return, on average, an area of ~1500 a.u., while we will find that the true average performance of the protocol provides an area slightly above 1000 a.u. That is, Golem will overestimate the robustness of the protocol. On the other hand,



if we assumed the uncertainty to be 30%, we would underestimate the robustness of the protocol, as we would expect an average area below 1000 a.u. In the case of variable P3, however, the location of the optimum is only slightly affected by uncertainty, such that despite the incorrect prediction, Golem would still accurately identify the location of the global optimum. That is, a tubing volume of  $\sim 0.3$  mL provides the best average outcome whether the true uncertainty is 10%, 20%, or 30%. In fact, while ignoring uncertainty altogether (*i.e.* assuming 0% uncertainty) would result in the largest overestimate of robustness, it would still have minimal impact in practice given that the prediction of the optimum location would still be accurate. This is not the case if we considered P1. If we again assume that the true uncertainty in this variable is 20%, providing Golem with an uncertainty model with 10% standard deviation would result in a protocol using a sample loop volume of  $\sim 0.04$  mL, while the optimal one should be  $\sim 0.06$  mL. Providing Golem with a 30% uncertainty instead would result in an underestimate of the protocol robustness and an unnecessarily conservative choice of  $\sim 0.08$  mL as the sample loop volume.

In summary, as anticipated in Section V.B, while an approximate estimate of  $p(\tilde{x})$  does not prevent the use of Golem, it can affect the quality of its predictions. When uncertainty is underestimated, the optimization solutions identified by Golem will tend to be less robust than expected. On the contrary, when uncertainty is overestimated, Golem's solutions will tend to be overly conservative (*i.e.*, Golem will favor plateaus in the objective function despite more peaked optima would provide better average performance). The errors in Golem's estimates will be proportional to the error in the estimates of the input uncertainty provided to it, but the magnitude of these errors is difficult to predict as it depends on the objective function, which is unknown and application-specific. Note that, ignoring input uncertainty corresponds to assuming  $p(\tilde{x})$  is a delta function in Golem. This choice, whether implicitly or explicitly made, results in the largest possible overestimate of robustness when uncertainty is in fact present. The associated error in the expected robustness is likely to be small when the true uncertainty is small, but may be large otherwise.

It is important to note that, above, we analyzed only one-dimensional slices of the six-dimensional parameter space. Given interactions between these parameters, noise in one parameter can affect the optimal setting of a different one (Section S.3.B†). Golem can identify these effects by studying its multi-dimensional robust surrogate model. Furthermore, for simplicity, here we considered noise in each of the six controllable parameters one at a time. It is nevertheless possible to consider concurrent noise in as many parameters as desired.

This example shows how Golem may be used to analyze prior experimental results and study the effect of input noise on protocol performance and the optimal setting of its controllable parameters.

## B. Optimization of a noisy HPLC protocol

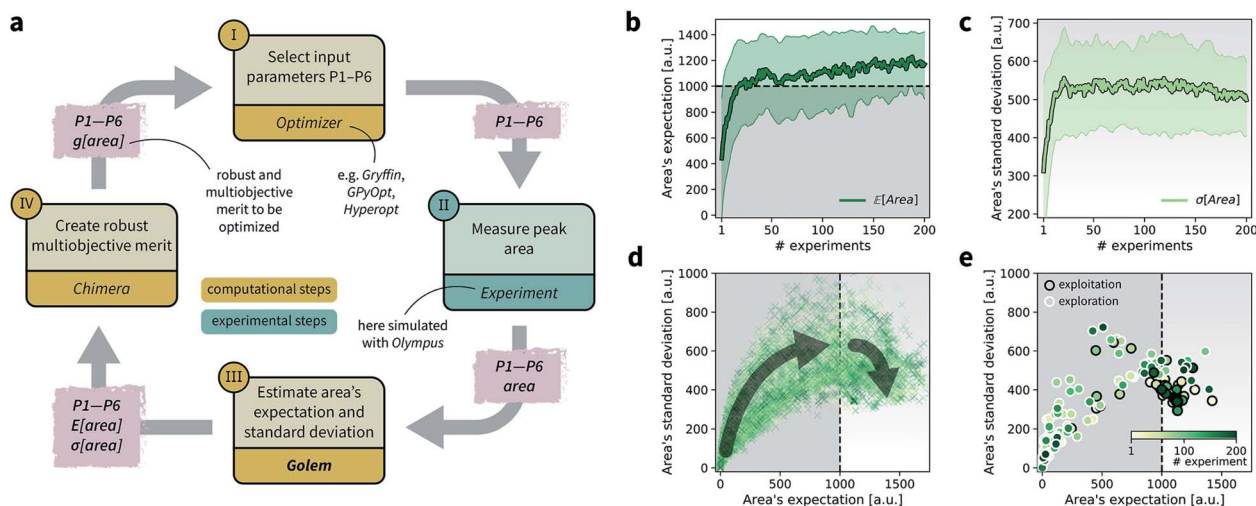
As a realistic and challenging example, we consider the optimization of the aforementioned HPLC sampling protocol under the presence of significant noise in P1 and P3 (noisy query

setting). In this first instance, we assume that the other conditions contain little noise and can thus be approximated as noiseless. As before, we consider normally distributed noise, truncated at zero. We assume a standard deviation of 0.008 mL for P1, and 0.08 mL for P3. In this example, we assume we are aware of the presence of input noise in these parameters, and are interested in achieving a protocol that returns an expected peak area,  $\mathbb{E}[\text{area}]$ , of at least 1000 a.u. As a secondary objective, we would like to minimize the output variability,  $\sigma[\text{area}]$ , as much as possible while maintaining  $\mathbb{E}[\text{area}] > 1000$  a.u.

To achieve the optimization goals, we use Golem to estimate both  $\mathbb{E}[\text{area}]$  and  $\sigma[\text{area}]$  as the optimization proceeds (Fig. 8a). We then use *Chimera*<sup>23</sup> to scalarize these two objectives into a single robust and multi-objective function,  $g[\text{area}]$ , to be optimized. *Chimera* is a scalarizing function that enables multi-objective optimization *via* the definition of a hierarchy of objectives and associated target values. As opposed to the post-hoc analysis discussed in the previous section, in this example we start with no prior experiment being available and let the optimization algorithm request new experiments in order to identify a suitable protocol. Here we perform virtual HPLC runs using *Olympus*,<sup>26</sup> which allows to simulate experiments *via* Bayesian Neural Network models. These probabilistic models capture the stochastic nature of experiments, such that they return slightly different outcomes every time an experiment is simulated. In other words, they simulate the heteroskedastic noise present in the experimental measurements. While measurement noise is not the focus of this work, it is another source of uncertainty routinely encountered in an experimental setting. As such, it is included in this example application. Bayesian optimization algorithms are generally robust to some level of measurement noise, as this source of uncertainty is inferred by the surrogate model. However, the combination of output and input noise in the same experiment is particularly challenging, as both sources of noise manifest themselves as noisy measurements despite the different origin. In fact, in addition to measurement noise, here we inject input noise into the controllable parameters P1 and P3. Hence, while the optimization algorithm may request a specific value for P1 and P3, the actual, realized ones will differ. This setup therefore contains noise in both input experimental conditions and measurements.

While large input noise would be catastrophic in most standard optimization campaigns (as shown in Section 5.2, Fig. 6), Golem allows the optimization to proceed successfully. With the procedure depicted in Fig. 8a, on average, *Gryffin* was able to identify parameter settings that achieve  $\mathbb{E}[\text{area}] > 1000$  a.u. after less than 50 experiments (Fig. 8b). Equivalent results were obtained with *GPyOpt* and *Hyperopt* (Fig. S15†). The improvements in this objective are, however, accompanied by a degradation in the second objective, output variability, as measured by  $\sigma[\text{area}]$  (Fig. 8c). This effect is due to the inevitable trade-off between the two competing objectives being optimized. After having reached its primary objective, the optimization algorithm mostly focused on improving the second objective, while satisfying the constraint defined for the first one. This behavior is visible in Fig. 8d and e. Early in the





**Fig. 8** Setup and results for the optimization of an HPLC protocol under noisy experimental conditions. (a) Procedure and algorithms used for the robust optimization of the HPLC protocol. First, the optimization algorithm selects the conditions of the next experiment to be performed. Second, the HPLC experiment is carried out and the associated peak's area recorded. Note that, in this example, P1 and P3 are noisy such that their values realized in the experiment do not correspond to those requested by the optimizer. Third, Golem is used to estimate the expected peak's area,  $\mathbb{E}[\text{area}]$ , as well as its variability  $\sigma[\text{area}]$ , based on a model of input noise for P1 and P3. Finally, the *Chimera* scalarizing function is used to combine these two objectives into a single figure of merit to be optimized. (b–e) Results of 50 optimization repeats performed with *Gryffin*. Equivalent results obtained with *GPyOpt* and *Hyperopt* are shown in Fig. S15†. (b) Optimization trace for the primary objective, i.e. the maximization of  $\mathbb{E}[\text{area}]$  above 1000 a.u. The average and standard deviation across 50 optimization repeats are shown. (c) Optimization trace for the secondary objective, i.e. the minimization of  $\sigma[\text{area}]$ . The average and standard deviation across 50 optimization repeats are shown. (d) Objective function values sampled during all optimization runs. The arrows indicate the typical trajectory of the optimizations, which first try to achieve values of  $\mathbb{E}[\text{area}]$  above 1000 a.u. and then try to minimize  $\sigma[\text{area}]$ . A Pareto front that describes the trade-off between the two objectives becomes visible, as larger area's expectation values are accompanied by larger variability. (e) Objective function values sampled during a sample optimization run. Each experiment is color-coded (yellow to dark green) to indicate at which stage of the optimization it was performed. Exploration (white rim) and exploitation (black rim) points are indicated, as *Gryffin* explicitly alternates between these two strategies. Later exploitation points (dark green, black rim) tend to focus on the minimization of  $\sigma[\text{area}]$ , having already achieved  $\mathbb{E}[\text{area}] > 1000$  a.u.

optimization, *Gryffin* is more likely to query parameter settings with low  $\mathbb{E}[\text{area}]$  and  $\sigma[\text{area}]$  values. At a later stage, with more information about the response surface, the algorithm focused on lowering  $\sigma[\text{area}]$  while keeping  $\mathbb{E}[\text{area}]$  above 1000 a.u. Due to input uncertainty, the Pareto front highlights an irreducible amount of output variance for any non-zero values of expected area (Fig. 8d). An analysis of the true robust objectives shows that, given the  $\mathbb{E}[\text{area}] > 1000$  a.u. constraint, the best achievable  $\sigma[\text{area}]$  values are  $\sim 300$  a.u. (Fig. S14†).

The traces showing the optimization progress (Fig. 8b–c) display considerable spread around the average performance. This is expected and due to the fact that both  $\mathbb{E}[\text{area}]$  and  $\sigma[\text{area}]$  are estimates based on scarce data, as they cannot be directly observed. As a consequence, these estimates fluctuate as more data is collected. In addition, it may be the case that while Golem estimates  $\mathbb{E}[\text{area}]$  to be over 1000 a.u., its true value for a certain set of input conditions may actually be below 1000, and *vice versa*. In fact, at the end of the 50 repeated optimization runs, 10 (i.e., 20%) of the identified optimal solutions had true  $\mathbb{E}[\text{area}]$  below 1000 a.u. (this was the case for 24% of the optimizations with *GPyOpt*, and 34% for those with *Hyperopt*). However, when using ensemble trees as the surrogate model, it is possible to obtain an estimate of uncertainty for Golem's expectation estimates. With this uncertainty estimate, one can control the probability that Golem's estimates satisfy the

objective's constraint that was set. For instance, to have a high probability of the estimate of  $\mathbb{E}[\text{area}]$  being above 1000 a.u., we can setup the optimization objective in *Chimera* with the constraint that  $\mathbb{E}[\text{area}] - 1.96 \times \sigma(\mathbb{E}[\text{area}]) > 1000$  a.u., which corresponds to optimizing against the lower bound of the 95% confidence interval of Golem's estimate. Optimizations set up in this way correctly identified optimal solutions with  $\mathbb{E}[\text{area}] > 1000$  a.u. in all 50 repeated optimization runs (Fig. S16†).

As a final test, we simulate the example above, in which we targeted the optimization of the lower-bound estimate of  $\mathbb{E}[\text{area}]$ , with all experimental conditions containing a considerable amount of noise. For all input variables we consider normally distributed noise truncated at zero, with a standard deviation of 0.008 mL for P1, 0.06 mL for P2, 0.08 mL for P3, 0.2 mL min<sup>-1</sup> for P4, 8 Hz for P5, and 1 s for P6. This is an even more challenging optimization scenario, with input noise compounding from all variables. In this case, *Hyperopt* achieved  $\mathbb{E}[\text{area}] > 1000$  a.u. after about 100 experiments on average, *Gryffin* achieved  $\mathbb{E}[\text{area}]$  values around the targeted value of 1000 a.u. after 120–130 experiments, and *GPyOpt* only when close to 200 experiments (Fig. S17†). As expected, the noisier the experimental conditions (larger noise and/or more noisy variables) the less efficient the optimization. However, Golem still enabled the algorithms tested to achieve the desired objective of

$\mathbb{E}[\text{area}] > 1000$  within the pre-defined experimental budget. After 200 experiments, *Hyperopt* correctly identified solutions with  $\mathbb{E}[\text{area}] > 1000$  a.u. in 78% of the optimization runs, *GPyOpt* in 70%, and *Gryffin* in 42%. We stress that Golem is not a substitute to developing precise experimental protocols. A noise-free (or reduced-noise) experimental protocol will always allow for faster optimization and better average performance. While Golem can mitigate the detrimental effects of input noise on optimization, it is still highly desirable to minimize noise in as many input conditions as possible.

This example application shows how Golem can easily be integrated into a Bayesian optimization loop for the optimization of experimental protocols with noisy experimental conditions.

## VII. Conclusion

In summary, Golem provides a simple, inexpensive, yet flexible approach for the optimization of experimental protocols under noisy experimental conditions. It can be applied retrospectively, for the analysis of previous results, as well as on-the-fly in conjunction with most experiment planning strategies to drive optimizations toward robust solutions. The approach was found to perform particularly well when used with systematic searches and Bayesian optimization algorithms. Optimization under noisy conditions is considerably more challenging than typical optimization tasks. When such noise is known but cannot be removed or corrected for, Golem enables optimizations that would otherwise be infeasible.

## Data availability

An open-source implementation of the Golem algorithm is available on GitHub, at <https://github.com/aspuru-guzik-group/golem>, under an MIT license.

## Author contributions

Matteo Aldeghi: conceptualization, methodology, software, investigation, formal analysis, writing, visualization. Florian Häse: conceptualization, methodology, software. Riley J. Hickman: conceptualization, methodology. Isaac Tamblin: supervision, project administration, funding acquisition. Alán Aspuru-Guzik: conceptualization, resources, supervision, project administration, funding acquisition.

## Conflicts of interest

A.A.-G. is the Chief Visionary Officer and a founding member of Kebotix, Inc.

## Acknowledgements

The authors thank Melodie Christensen and Lars Yunker for valuable and insightful discussions. The authors acknowledge the generous support of the National Research Council (NRC) of the Government of Canada. M.A. is supported by a Postdoctoral

Fellowship of the Vector Institute. F.H. acknowledges financial support from the Herchel Smith Graduate Fellowship and the Jacques-Emile Dubois Student Dissertation Fellowship. R.J.H. gratefully acknowledges the Natural Sciences and Engineering Research Council of Canada (NSERC) for provision of the Postgraduate Scholarships-Doctoral Program (PGSD3-534584-2019). A.A.-G. acknowledges support from the Canada 150 Research Chairs program and CIFAR. A.A.-G. acknowledges the generous support from Anders G. Frøseth. All computations reported in this paper were completed on the computing clusters of the Vector Institute. Resources used in preparing this research were provided, in part, by the Province of Ontario, the Government of Canada through CIFAR, and companies sponsoring the Vector Institute. Finally, we thank the anonymous reviewers for their critical reading and comments that resulted in a much-improved manuscript.

## References

- 1 M. Christensen, L. Yunker, F. Adediji, F. Häse, L. Roch, T. Gensch, G. dos Passos Gomes, T. Zepel, M. Sigman, A. Aspuru-Guzik and J. Hein, Data-science driven autonomous process optimization, *Commun. Chem.*, 2021, **4**(112), 11.
- 2 B. J. Shields, J. Stevens, J. Li, M. Parasram, F. Damani, I. Jesus, A. Martinez, J. M. Janey, R. P. Adams and A. G. Doyle, Bayesian reaction optimization as a tool for chemical synthesis, *Nature*, 2021, **590**(7844), 89–96.
- 3 C. A. Nicolaou and N. Brown, Multi-objective optimization methods in drug design, *Drug Discovery Today: Technol.*, 2013, **10**(3), e427–e435.
- 4 R. Gómez-Bombarelli, J. N. Wei, D. Duvenaud, J. M. Hernández-Lobato, B. Sánchez-Lengeling, D. Sheberla, J. Aguilera-Iparraguirre, T. D. Hirzel, R. P. Adams and A. Aspuru-Guzik, Automatic chemical design using a data-driven continuous representation of molecules, *ACS Cent. Sci.*, 2018, **4**(2), 268–276.
- 5 N. T. Shijing Sun, P. Hartono, Z. D. Ren, F. Oviedo, A. M. Buscemi, M. Layurova, D. X. Chen, T. Ogunfunmi, J. Thapa, S. Ramasamy, C. Settens, B. L. DeCost, A. G. Kusne, Z. Liu, I. Siyu and P. Tian, Ian Marius Peters, Juan-Pablo Correa-Baena, and Tonio Buonassisi. Accelerated development of perovskite-inspired materials via high-throughput synthesis and machine-learning diagnosis, *Joule*, 2019, **3**(6), 1437–1451.
- 6 R. Winter, F. Montanari, A. Steffen, H. Briem, N. Frank and D.-A. Clevert, Efficient multi-objective molecular optimization in a continuous latent space, *Chem. Sci.*, 2019, **10**, 8016–8024.
- 7 Z. Yao, B. Sánchez-Lengeling, N. Scott Bobbitt, B. J. Bucior, S. G. Hari Kumar, S. P. Collins, T. Burns, T. K. Woo, O. K. Farha, R. Q. Snurr and A. Aspuru-Guzik, Inverse design of nanoporous crystalline reticular materials with deep generative models, *Nature Machine Intelligence*, 2021, **3**(1), 76–86.





- 8 F. Häse, L. M. Roch and A. Aspuru-Guzik, Next-Generation Experimentation with Self-Driving Laboratories, *Trends Chem.*, 2019, **1**(3), 282–291.
- 9 P. S. Gromski, A. B. Henson, J. M. Granda and C. Leroy, How to explore chemical space using algorithms and automation, *Nat. Rev. Chem.*, 2019, **3**(2), 119–128.
- 10 H. S. Stein and J. M. Gregoire, Progress and prospects for accelerating materials science with automated and autonomous workflows, *Chem. Sci.*, 2019, **10**, 9640–9649.
- 11 T. Dimitrov, C. Kreisbeck, J. S. Becker, A. Aspuru-Guzik and K. S. Semion, Autonomous molecular design: Then and now, *ACS Appl. Mater. Interfaces*, 2019, **11**(28), 24825–24836.
- 12 C. W. Coley, N. S. Eyke and K. F. Jensen, Autonomous discovery in the chemical sciences part i: Progress, *Angew. Chem., Int. Ed.*, 2020, **59**(51), 22858–22893.
- 13 C. W. Coley, N. S. Eyke and K. F. Jensen, Autonomous discovery in the chemical sciences part ii: Outlook, *Angew. Chem., Int. Ed.*, 2020, **59**(52), 23414–23436.
- 14 M. M. Flores-Leonar, L. M. Mejía-Mendoza, A. Aguilar-Granda, B. Sanchez-Lengeling, T. Hermann, C. Amador-Bedolla and A. Aspuru-Guzik, Materials acceleration platforms: On the way to autonomous experimentation, *Green Sustain. Chem.*, 2020, **25**, 100370.
- 15 P. Nikolaev, D. Hooper, F. Webber, R. Rao, K. Decker, M. Krein, J. Poleski, R. Barto and B. Maruyama, Autonomy in materials research: a case study in carbon nanotube growth, *npj Comput. Mater.*, 2016, **2**(1), 1–6.
- 16 B. Maruyama, K. Decker, M. Krein, J. Poleski, R. Barto, F. Webber and P. Nikolaev, Autonomous experimentation applied to carbon nanotube synthesis, *Microsc. Microanal.*, 2017, **23**(S1), 182.
- 17 J. M. Granda, L. Donina, V. Dragone, D.-L. Long and L. Cronin, Controlling an organic synthesis robot with machine learning to search for new reactivity, *Nature*, 2018, **559**(7714), 377–381.
- 18 B. P. MacLeod, F. G. L. Parlane, T. D. Morrissey, F. Häse, L. M. Roch, K. E. Dettelbach, R. Moreira, L. P. E. Yunker, M. B. Rooney, J. R. Deeth, V. Lai, G. J. Ng, H. Situ, R. H. Zhang, M. S. Elliott, T. H. Haley, D. J. Dvorak, A. Aspuru-Guzik, J. E. Hein and C. P. Berlinguette, Self-driving laboratory for accelerated discovery of thin-film materials, *Sci. Adv.*, 2020, **6**(20), eaaz8867.
- 19 S. Langner, F. Häse, J. Darío Perea, S. Tobias, J. Hauch, M. R. Loïc, T. Heumueller, A. Aspuru-Guzik and J. B. Christoph, Beyond Ternary OPV: High-Throughput Experimentation and Self-Driving Laboratories Optimize Multicomponent Systems, *Adv. Mater.*, 2020, **32**(14), 1907801.
- 20 J. Grizou, L. J. Points, A. Sharma and L. Cronin, A curious formulation robot enables the discovery of a novel protocell behavior, *Sci. Adv.*, 2020, **6**(5), eaay4237.
- 21 H. Tao, T. Wu, S. Kheiri, M. Aldeghi, A. Aspuru-Guzik and E. Kumacheva, Self-driving platform for metal nanoparticle synthesis: Combining microfluidics and machine learning, *Adv. Funct. Mater.*, 2021, 2106725.
- 22 F. Häse, M. R. Loïc, C. Kreisbeck and A. Aspuru-Guzik, Phoenix: A bayesian optimizer for chemistry, *ACS Cent. Sci.*, 2018, **4**(9), 1134–1145.
- 23 F. Häse, L. M. Roch and A. Aspuru-Guzik, Chimera: enabling hierarchy based multi-objective optimization for self-driving laboratories, *Chem. Sci.*, 2018, **9**(39), 7642–7655.
- 24 F. Häse, M. Aldeghi, R. J. Hickman, L. M. Roch and A. Aspuru-Guzik, Gryffin: An algorithm for bayesian optimization of categorical variables informed by expert knowledge, *Appl. Phys. Rev.*, 2021, **8**, 031406.
- 25 I. M. Pendleton, C. Gary, Z. Li, M. Ani Najeeb, S. A. Friedler, A. J. Norquist, E. M. Chan and J. Schrier, Experiment specification, capture and laboratory automation technology (escalate): a software pipeline for automated chemical experimentation and data management, *MRS Commun.*, 2019, **9**(3), 846–859.
- 26 F. Häse, M. Aldeghi, R. J. Hickman, L. M. Roch, M. Christensen, E. Liles, J. E. Hein, and A. Aspuru-Guzik., *Olympus: a benchmarking framework for noisy optimization and experiment planning*, 2020.
- 27 K. Felton, R. Jan, and A. Lapkin. *Summit: Benchmarking Machine Learning Methods for Reaction Optimisation*, 2020, vol. 12.
- 28 B. Rohr, H. S. Stein, D. Guevarra, Y. Wang, J. A. Haber, M. Aykol, S. K. Suram and J. M. Gregoire, Benchmarking the acceleration of materials discovery by sequential learning, *Chem. Sci.*, 2020, **11**, 2696–2706.
- 29 P. I. Frazier, *A tutorial on bayesian optimization*, 2018, arXiv, 1807.02811.
- 30 B. Shahriari, S. Kevin, Z. Wang, R. P. Adams and N. De Freitas, Taking the Human Out of the Loop: A Review of Bayesian Optimization Bobak, *Proc. IEEE*, 2015, **104**(1), 148–175.
- 31 M. Gen and L. Lin, *Genetic Algorithms*, American Cancer Society, 2008, pp. 1–15.
- 32 J. McCall, Genetic algorithms for modelling and optimisation, *J. Comput. Appl. Math.*, 2005, **184**(1), 205–222, Special Issue on Mathematics Applied to Immunology.
- 33 M. Srinivas and L. M. Patnaik, Genetic algorithms: a survey, *Computer*, 1994, **27**(6), 17–26.
- 34 B. Burger, P. M. Maffettone, V. V. Gusev, C. M. Aitchison, B. Yang, X. Wang, X. Li, B. M. Alston, B. Li, R. Clowes, N. Rankin, B. Harris, R. Sebastian Sprick and A. I. Cooper, A mobile robotic chemist, *Nature*, 2020, **583**(7815), 237–241.
- 35 A. E. Gongora, B. Xu, W. Perry, C. Okoye, P. Riley, K. G. Reyes, E. F. Morgan and K. A. Brown, A bayesian experimental autonomous researcher for mechanical design, *Sci. Adv.*, 2020, **6**(15), eaaz1708.
- 36 H. G. Beyer and B. Sendhoff, Robust optimization - A comprehensive survey, *Comput. Methods Appl. Mech. Eng.*, 2007, **196**(33–34), 3190–3218.
- 37 S. P. Jones, George Box and robust design, *Appl. Stoch Model Bus. Ind.*, 2014, **30**(1), 46–52.
- 38 D. Bertsimas, D. B. Brown and C. Caramanis, Theory and applications of robust optimization, *SIAM Rev.*, 2011, **53**(3), 464–501.



- 39 B. George, Dantzig. Linear programming under uncertainty, *Manage. Sci.*, 1955, **1**(3–4), 197–206.
- 40 W. B. Powell, *A unified framework for stochastic optimization*, 2019.
- 41 K. Zhou, J. C. Doyle, and K. Glover, *Robust and Optimal Control*, Prentice-Hall, Inc., USA, 1996.
- 42 Q. Chen, Comparing Probabilistic and Fuzzy Set Approaches for Designing in the Presence of Uncertainty, PhD thesis, Virginia Polytechnic Institute and State University, 2000.
- 43 M. Inuiguchi, Fuzzy programming approaches to robust optimization, in *Modeling Decisions for Artificial Intelligence*, ed. Vicenç Torra, Yasuo Narukawa, Beatriz López, and Mateu Villaret, Springer Berlin Heidelberg, Berlin, Heidelberg, 2012, pp. 11–12.
- 44 L. Jeff Hong and G. Liu, Simulating sensitivities of conditional value at risk, *Manag. Sci.*, 2009, **55**(2), 281–293.
- 45 S. Cakmak, R. Astudillo Marban, P. Frazier, and E. Zhou, Bayesian optimization of risk measures, in *Advances in Neural Information Processing Systems*, ed. H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, Curran Associates, Inc., 2020, vol. 33, pp. 20130–20141.
- 46 D. Bertsimas, O. Nohadani and K. Meng Teo, Robust Optimization for Unconstrained Simulation-Based Problems, *Oper. Res.*, 2009, **58**(1), 161–178.
- 47 I. Bogunovic, S. Jegelka, J. Scarlett and V. Cevher, Adversarially robust optimization with Gaussian processes, *Adv. Neural Inf. Process. Syst.*, 2018, 5760–5770.
- 48 P. Dellaportas and A. David, Stephens. Bayesian analysis of errors-in-variables regression models, *Biometrics*, 1995, **51**(3), 1085–1095.
- 49 M. Pearce and J. Branke, Bayesian simulation optimization with input uncertainty, in *2017 Winter Simulation Conference*, WSC, 2017, pp. 2268–2278.
- 50 S. Toscano-Palmerin and P. I. Frazier, *Bayesian optimization with expensive integrands*, 2018.
- 51 J. Nogueira, R. Martinez-Cantin, A. Bernardino, and L. Jamone, Unscented bayesian optimization for safe robot grasping, in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2016, pp. 1967–1972.
- 52 S. J. Julier, The scaled unscented transformation, in *Proceedings of the 2002 American Control Conference*, IEEE Cat. No.CH37301, 2002, vol. 6, pp. 4555–4559.
- 53 A. Girard and R. Murray-Smith, *Learning a Gaussian Process Model with Uncertain Inputs*, 2003, pp. 1–10.
- 54 A. C. Damianou, M. K. Titsias, and N. D. Lawrence, *Variational Inference for Uncertainty on the Inputs of Gaussian Process Models*, 2014, pp. 1–51.
- 55 L. P. Fröhlich, E. D. Klenske, V. Julia, C. Daniel, and M. N. Zeilinger. *Noisy-input entropy search for efficient robust bayesian optimization*, 2020.
- 56 J. J. Beland and P. B. Nair, Bayesian Optimization Under Uncertainty, *31st Conference on Neural Information Processing Systems (NIPS 2017) Workshop on Bayesian optimization (BayesOpt 2017)*, (1), 2017, pp. 1–5.
- 57 E. C. Garrido-Merchán and D. Hernández-Lobato, Dealing with categorical and integer-valued variables in Bayesian Optimization with Gaussian processes, *Neurocomputing*, 2020, **380**, 20–35.
- 58 M. Lindauer, K. Eggensperger, M. Feurer, S. Falkner, A. Biedenkapp, and F. Hutter, *Smac v3: Algorithm configuration in python*, 2017, <https://github.com/automl/SMAC3>.
- 59 M. Mistry, R. M. Lee, N. Sudermann-Merx, A. Thebelt, J. Kronqvist and R. Misener, *ENTMOOT: A Framework for Optimization over Ensemble Tree Models*, 2020, arXiv, 2003.04774.
- 60 The GPyOpt authors, *Gpyopt: A bayesian optimization framework in python*, 2016, <http://github.com/SheffieldML/GPyOpt>.
- 61 J. S. Bergstra, D. Yamins, and D. D. Cox, *Making a science of model search: Hyperparameter optimization in hundreds of dimensions for vision architectures*, 2013.
- 62 F.-A. Fortin, F.-M. De Rainville, M.-A. Gardner, M. Parizeau and C. Gagné, DEAP: Evolutionary algorithms made easy, *Journal of Machine Learning Research*, 2012, **13**, 2171–2175.
- 63 A. K. Nigam, P. Friederich, M. Krenn, and A. Aspuru-Guzik. *Augmenting genetic algorithms with deep neural networks for exploring the chemical space*, 2020.
- 64 L. Breiman, Random Forests, *Mach. Learn.*, 2001, **45**(1), 5–32.
- 65 P. Geurts, D. Ernst and L. Wehenkel, Extremely randomized trees, *Mach. Learn.*, 2006, **63**(1), 3–42.
- 66 M. R. Loïc, F. Häse, C. Kreisbeck, T. Tamayo-Mendoza, L. P. E. Yunker, J. E. Hein and A. Aspuru-Guzik, Chemos: An orchestration software to democratize autonomous discovery, *PLoS One*, 2020, **15**(4), 1–18.

