Supplementary Material: Scalable Constrained Bayesian Optimization

David Eriksson[1]
Facebook
deriksson@fb.com

Matthias Poloczek[1]
Amazon
matpol@amazon.com

In Sect. 1 we describe how we leverage scalable GP regression to run SCBO and cEI with large sampling budgets. The value and scalability of our implementation is demonstrated by an experiment in Sect. 2. We summarize the hyperparameters of SCBO in Sect. 3 and give additional details on how we shrink and expand the trust region. We prove a consistency result for SCBO in Sect. 4. In Sect. 6 we show results for all baselines on the physics test problems where the objective and constraints have been transformed in the same fashion as in SCBO. Finally, Sect. 7 provides details on all benchmarks.

1 Gaussian process regression

As usual, the hyperparameters of the Gaussian process (GP) model are fitted by optimizing the log-marginal likelihood. The domain is rescaled to $[0, 1]^d$ and the function values are standardized to have mean zero and variance one before fitting the GP. We use a Matérn-5/2 kernel with ARD and a constant mean function that we optimize using L-BFGS-B. Following Snoek et al. [2012], a horseshoe prior is placed on the noise variance. We also learn a signal variance of the kernel.

Scaling BO to large number of evaluations is challenging due to the computational costs of inference. To compute the posterior distribution for n observations, we need to solve linear systems with an $n \times n$ kernel matrix. This is commonly done via a Cholesky decomposition which has a computational complexity of $\Theta(n^3)$ flops. When there are m constraints, the cost increases to $\Theta(mn^3)$ flops and thus may not scale to the large sampling budgets that we consider in this work. Thus, we leverage the parallelism of modern GPUs that allows to 'batch' several GP models which is provided in the GPyTorch package [Gardner et al., 2018]. Relying only on fast matrix vector multiplication (MVM), we can solve linear systems with the kernel matrix using the conjugate gradient (CG) method and approximate the log-determinant via the Lanczos process Dong et al. [2017], Ubaru et al. [2017]. GPyTorch extends this idea to a batch of GPs by computing fast MVMs with a 3D tensor representing a kernel matrix of size $(m + 1) \times n \times n$, where $m+1$ is the number of batched GPs. The MVMs are further sped up by a compiled CUDA kernel constructed via KeOPS [Charlier et al., 2018]. To the best of our knowledge, SCBO is the first Bayesian optimization algorithm to leverage batch GPs and KeOPS. Note that we also applied these ideas to scale cEI of Schonlau et al. [1998] to a large numbers of samples.

[1]This work was conducted while the authors were affiliated with Uber AI.

2 Achieving Efficiency via Batch GPs

Next we describe an experiment that demonstrates that the Cholesky decomposition, which is commonly used in GP regression, does not scale to the large sampling budgets that are required for the demanding benchmarks that we study in this work. We consider training GPs with a different number of training points. The true function is standardized and we assume observations are subject to normally distributed noise with mean zero and variance 0.01. We compare the computational cost of the Cholesky decomposition to the efficient batch GP implementation of GPyTorch, both in single precision. Table 1 provides run times for training, predictions, and sampling. We use 50 gradient steps for training. All computations were performed on an NVIDIA RTX 2080 TI. The two rightmost columns also show the error for the mean and variance predictions. We see that batch GPs achieve a large speed-up while preserving a high accuracy. For example, with the batch GP implementation fitting one GP with 1000 points takes 2.33 seconds, while fitting 100 GPs in a batch only takes 11.15 seconds. Moreover, the Cholesky approach becomes impractical in the large-data regime: training 100 GPs with 8000 points takes 37 minutes, compared to just about 2 minutes for the batch GP implementation!

Data size		Cholesky decomposition			Scalable batch GPs			Prediction error	
#GPs	Training points	Training	Prediction	Sampling	Training	Prediction	Sampling	Mean MAE	Variance MAE
1	1000	0.60 s	0.06 s	0.07 s	2.09 s	0.04 s	0.39 s	1.41e-03	4.44e-03
1	2000	1.03 s	0.08 s	0.09 s	2.17 s	0.05 s	0.39 s	2.12e-03	5.23e-04
1	4000	3.67 s	0.16 s	0.14 s	2.26 s	0.06 s	0.41 s	2.22e-04	4.19e-06
1	8000	22.87 s	0.49 s	0.32 s	4.73 s	0.13 s	0.45 s	3.94e-04	6.75e-05
10	1000	5.98 s	0.47 s	0.58 s	2.49 s	0.22 s	1.16 s	4.74e-02	1.00e-01
10	2000	10.13 s	0.76 s	0.77 s	3.61 s	0.31 s	1.30 s	5.00e-02	6.14e-02
10	4000	35.96 s	1.53 s	1.26 s	7.81 s	0.35 s	0.95 s	3.42e-04	9.89e-05
10	8000	227.30 s	4.72 s	2.88 s	17.73 s	0.70 s	1.08 s	5.24e-05	7.07e-06
50	1000	24.48 s	2.40 s	2.96 s	7.37 s	0.50 s	2.77 s	5.03e-03	3.38e-02
50	2000	49.02 s	3.72 s	3.88 s	9.65 s	0.90 s	3.12 s	4.06e-03	1.64e-03
50	4000	184.61 s	7.90 s	6.40 s	21.15 s	1.75 s	3.57 s	1.49e-04	2.10e-05
50	8000	1134.58 s	25.37 s	14.36 s	66.50 s	3.72 s	4.40 s	5.74e-04	1.15e-04
100	1000	55.94 s	5.09 s	6.11 s	10.41 s	1.03 s	6.85 s	1.85e-03	9.22e-04
100	2000	108.00 s	8.42 s	8.38 s	18.59 s	2.27 s	8.61 s	2.12e-02	2.40e-02
100	4000	365.88 s	16.64 s	12.54 s	44.08 s	3.91 s	8.42 s	4.56e-04	1.25e-04
100	8000	2303.86 s	89.19 s	28.48 s	144.12 s	13.14 s	10.50 s	1.24e-04	2.31e-05

Table 1: Computational cost for GP training, prediction, and sampling. The standard approach using the Cholesky decomposition in single precision is compared to a fast implementation using batch GPs. We take 50 gradient steps for training and predict/sample on 5000 test points. The mean and variance MAE between the two approaches is shown in the two rightmost columns.

3 Details on SCBO

In all experiments we use the following hyperparameters for SCBO that were adopted from TURBO [Eriksson et al., 2019]: $\tau_s = \max(3, \lceil d/10 \rceil)$, $\tau_f = \lceil d/q \rceil$, $L_{\min} = 2^{-7}$, $L_{\max} = 1.6$, $L_{\text{init}} = 0.8$, and perturbation probability $p_{\text{perturb}} = \min\{1, 20/d\}$, where d is the number of dimensions and q is the batch size. Note that $\tau_s = 3$ is used by [Eriksson et al., 2019], while we find that SCBO achieves better performance if we expand the trust region less frequently. Recall that we assume the domain has been scaled to the unit hypercube $[0, 1]^d$. A *success* occurs if at least one evaluation in the batch improves on the incumbent. In this case, we increment the success counter and reset the failure counter to zero. If no point in the batch improves the current best solution, we set the success counter to zero and increment the failure counter. We increase the side length to $\min(2L, L_{\max})$ and reset botch counters to zero if the success counter reaches τ_s. Similarly, if the failure counter reaches τ_f, we set the side length to $L/2$ and reset both counters. If $L < L_{\min}$, we terminate the trust region and spawn a new trust region that is initialized with a Sobol sequence. This trust region does not use any data collected by previous trust regions. The discretized candidate set of size r is also generated following the approach of Eriksson et al. [2019] and we use $r = \min(200d, 5000)$ for all experiments. In particular, we first create a scrambled Sobol sequence within the intersection of the trust region and the domain $[0, 1]^d$. We use the value in the Sobol sequence with probability p_{perturb} for a given candidate and dimension, and the value of the center otherwise.

4 Global Consistency of SCBO

We prove that SCBO converges to a global optimum as the number of samples tends to infinity.

Theorem 1. *Suppose that SCBO with default parameters is used in a multi-start framework under the following conditions:*

1. *The initial points $\{y_i\}$ for SCBO are chosen such that for any $\delta > 0$ and $x \in [0, 1]^d$ there exists $\nu(x, \delta) > 0$ such that the probability that at least one point in $\{y_i\}$ ends up in a ball centered at x with radius δ is at least $\nu(x, \delta)$.*
2. *The objective and constraints are bounded.*
3. *There is a unique global minimizer x^*.*
4. *SCBO considers any sampled point an improvement* only if *it improves the current best solution by at least some constant $\gamma > 0$.*

Then, SCBO with noise-free observations converges to the global minimizer x^*.

Note that condition (1) is met if the initial set is chosen uniformly at random. Conditions (2) and (3) hold almost surely under the prior for our domain and are common assumptions in global optimization [e.g., see Regis and Shoemaker, 2007, Spall, 2005]. Condition (4) is a straightforward design decision of the algorithm.

Proof. First observe that SCBO will take only a finite number of samples for any trust region due to conditions (2) and (4). Thus, SCBO will restart infinitely

often with a fresh trust region and hence there is an infinite subsequence $\{x_k(i)\}$ of initial points. This subsequence satisfies condition (1) by design. Thus, global convergence follows from the proof of global convergence for random search under condition (3) [e.g., see Spall, 2005]. □

Note that this result also implies consistency for the TURBO algorithm of Eriksson et al. [2019].

5 Additional ablation studies

In this section we provide two additional ablation studies that investigate the components of SCBO. In Fig. 1 we look at the effect of the copula and bilog transforms that are used by SCBO. We note that the bilog transform is more important on the 30D Keane function and that using both transforms works better than only using one of them. The Copula transform improves the performance on the Rosenbrock problem. Even though using the bilog transformation slows down the convergence, the final performance achieved by SCBO (that uses both transformations) is comparable to only using the Copula transformation.

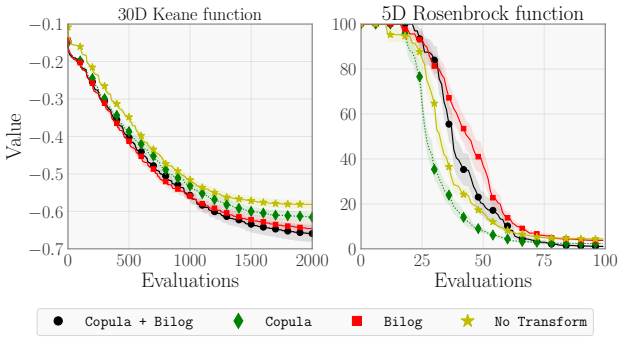

Figure 1: Individual contribution of the copula and bilog transforms in SCBO. **(Left)** The bilog transform is more important on the $30D$ Keane function. **(Right)** The copula transform improves the results on the $5D$ Rosenbrock function.

Another important component of SCBO is the perturbation probability from Sect. 3. Perturbing only a subset of the dimensions helps SCBO generate candidates that are closer to the current best point which improves the performance on the 30D Keane function, as we see in Fig. 2.

6 Results on the Transformed Physics Problems

Recall that SCBO applies the copula transformation to the objective function and the bilog transformation

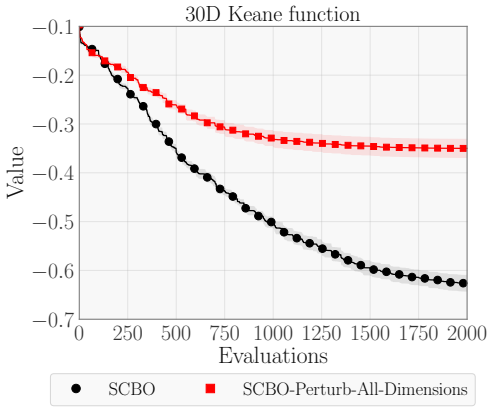

Figure 2: Results on the 30D Keane function that illustrates the importance of only perturbing a subset of dimensions when generating the candidate set.

to each constraint. In this section we study on the four physics benchmarks how the performance of all baselines is affected by these transformations. Fig. 3 summarizes the performances. We note that the transformations do not lead to noticeable changes in performance for the baselines, except for PESC that benefits on the 3D tension-compression string problem and on the 4D welded beam design. The performance was comparable with and without the transformations for the other test problems.

7 Details on the Benchmarks

In this section we provide additional information for the test problems. We refer the reader to the original papers for more details.

7.1 3D Tension-Compression String

The tension-compression string problem was described by Hedar and Fukushima [2006]. The goal is to minimize the weight subject to constraints on minimum deflection, shear stress, surge frequency, limits on outside diameter, and on design variables [Coello and Montes, 2002]. The first constraint is very sensitive to changes in the input parameters and cannot be modeled accurately by a global GP model.

7.2 4D Pressure Vessel Design

This problem was studied by Coello and Montes [2002] and has four constraints. The original problem does not specify bound constraints, so we use $0 \leq x_1, x_2 \leq 10$, $10 \leq x_3 \leq 50$, and $150 \leq x_4 \leq 200$. This domain contains the best solution found by Coello and Montes [2002]. The goal is to minimize the total cost of designing the vessel, which includes the cost of the material,

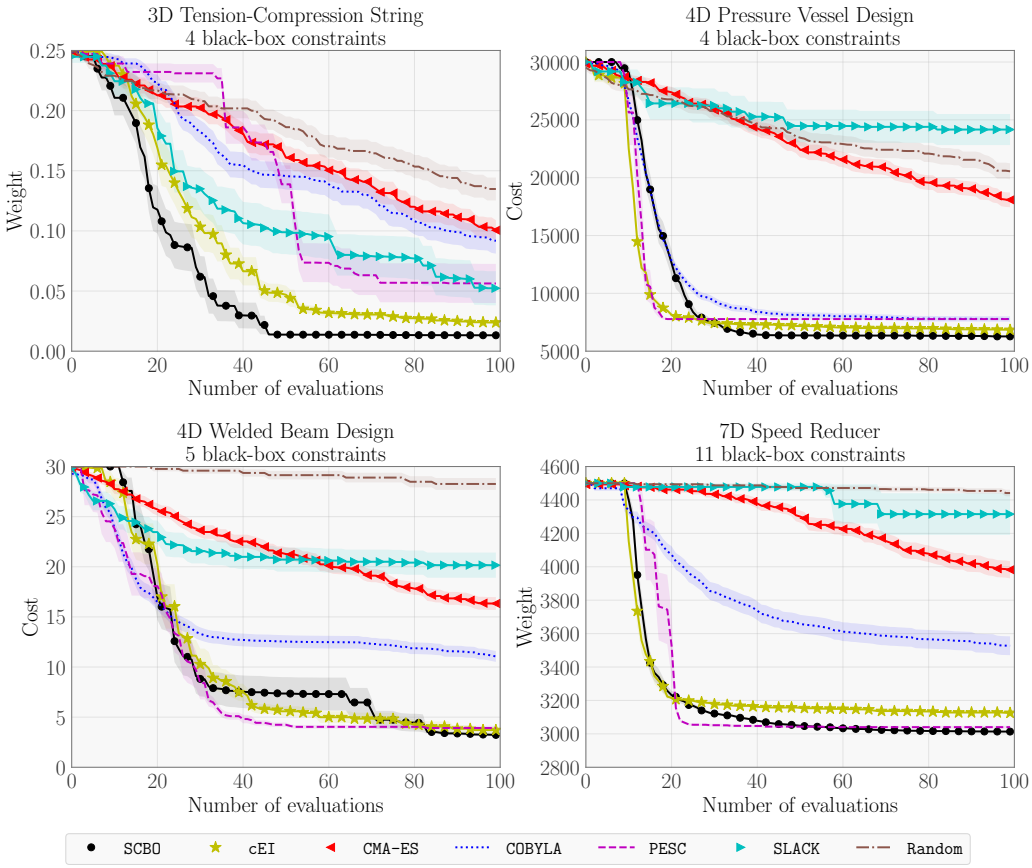

Figure 3: Results on the transformed physics problems, where the transformations of SCBO have been applied directly to the objective functions and constraints. **(Upper left)** SCBO and cEI outperform the other methods on the Tension-compression string problem. **(Upper right)** SCBO finds the best solutions on the pressure vehicle design problem, followed by cEI, PESC, and COBYLA. **(Lower left)** PESC, and cEI are eventually outperformed by SCBO on the welded beam design problem. **(Lower right)** SCBO and PESC perform the best on the speed reducer problem.

forming, and welding. The variables describe the thickness of the shell, thickness of the head, inner radius, and length of the cylindrical section of the vessel. The thickness of the shell and thickness of the head have to be multiples of 0.0625 and are rounded to the closest such value before evaluating the objective and constraints.

7.3 4D Welded Beam Design

This problem was considered by Hedar and Fukushima [2006] and has 5 constraints. The objective is to minimize the cost subject to constraints on shear stress, bending stress in the beam, buckling load on the bar, end deflection of the beam, and three additional side constraints.

7.4 7D Speed Reducer

The 7D speed reducer model has 11 black-box constraints and was described by Lemonge et al. [2010]. The objective is to minimize the weight of a speed reducer. The design variables are the face width, the module of teeth, the number of teeth on pinion, the length of shaft one between the bearings, the length of shaft two between the bearings, the diameter of shaft one, and the diameter of shaft two.

7.5 10D Ackley

In this problem we consider the popular 10-dimensional Ackley function

$$f(x) = -20\exp\left(-0.2\sqrt{\frac{1}{d}\sum_{i=1}^{d}x_i^2}\right) - \exp\left(\frac{1}{d}\sum_{i=1}^{d}\cos(2\pi x_i)\right) + 20 + \exp(1)$$

in the domain $[-5, 10]^{10}$ subject to the constraints $c_1(x) = \sum_{i=1}^{10} x_i \leq 0$ and $c_2(x) = \|x\|_2 - 5 \leq 0$. This

function is multi-modal and hard to optimize. Additionally, the size of the feasible region is small making this problem even more challenging. The optimal value is zero and is attained at the origin.

7.6 30D Keane Bump

For the Keane bump benchmark [Keane, 1994], the goal is to minimize

$$f(x) = -\left| \frac{\sum_{i=1}^{d} \cos^4(x_i) - 2\prod_{i=1}^{d} \cos^2(x_i)}{\sqrt{\sum_{i=1}^{d} i x_i^2}} \right|$$

subject to $c_1(x) = 0.75 - \prod_{i=1}^{d} x_i \leq 0$ and $c_2(x) = \sum_{i=1}^{d} x_i - 7.5d \leq 0$ over the domain $[0, 10]^d$. We consider the case $d = 30$ in our experiment. The Keane benchmark is notoriously famous for being challenging for Bayesian optimization as it is hard to model with a global GP model.

7.7 12D Robust Multi-point Optimization

The goal is to learn a robust controller for the lunar lander in the OpenAI gym[1]. The state space for the lunar lander is the position, angle, time derivatives, and whether or not either leg is in contact with the ground. For each frame there are four possible actions: firing a booster engine left, right, up, or doing nothing. The original objective was to maximize the average final reward over m randomly generated terrains, initial positions, and velocities. We extend this formulation to be more robust by adding m constraints that no individual reward is below 200, which asserts that the lunar lander successfully lands in every scenario. Moreover, we fix the m terrains throughout the optimization process therefore making the function evaluations deterministic (that is, without noise).

7.8 60D Rover Trajectory Planning

This problem was considered by Wang et al. [2018]. The goal is to optimize the trajectory of a rover, where the trajectory is determined by fitting a B-spline to 30 design points in a 2D plane. The reward function is $f(\vec{x}) = c(\vec{x}) + 5$, where $c(\vec{x})$ penalizes any collision with an object along the trajectory by -20. We add the constraint that the trajectory has to start and end at the pre-specified start and end locations. Additionally, we add 15 additional obstacles that are impassable and introduce constraints $c_i(x)$ for each i-th obstacle o_i based on the final trajectory $\gamma(x)$ as follows:

$$c_i(x) = \begin{cases} -d(o_i, \gamma(x)) & \text{if } \gamma(x) \cap o_i = \emptyset, \\ \max\limits_{\alpha \in \gamma(x) \cap o_i} \min\limits_{\beta \in \partial o_i} d(\alpha, \beta) & \text{otherwise,} \end{cases}$$

where ∂o_i is the boundary of o_i. That is, trajectories that do not collide with the object will be feasible under this constraint with a constraint value equal to the minimum distance between the trajectory and the object. Trajectories that collide with the object will be given a larger constraint value if they intersect close to the center of the object.

7.9 124D Vehicle Design with 68 Constraints (MOPTA08)

MOPTA08 is a large-scale multi-disciplinary optimization problem from the vehicle industry [Anjos, 2008]. There are 124 variables that describe gages, materials, and shapes as well as 68 performance constraints. Note that a problem with this many input dimensions and black-box constraints is out of reach for existing methods in Bayesian optimization.

7.10 2D Toy problem

This problem was proposed by Hernández-Lobato et al. [2016]. The goal is to minimize the function $f(x) = x_1 + x_2$ subject to $c_1(x) = 1.5 - x_1 - 2x_2 - 0.5\sin(2\pi(x_1^2 - 2x_2)) \leq 0$ and $c_2(x) = x_1^2 + x_2^2 - 1.5 \leq 0$. The objective and constraints are all smooth low-dimensional functions. The domain is the unit square $[0, 1]^2$.

7.11 5D Rosenbrock function

The goal is to minimize the Rosenbrock function $f(x) = \sum_{i=1}^{4} \left[100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2\right]$ subject to two constraints involving the Dixon-Price[2] (DP) function $c_1(x) = f_{\text{DP}}(x) - 10 \leq 0$ and the Levy[3] function $c_2(x) = f_{\text{Levy}}(x) - 10 \leq 0$. We created this problem to illustrate a setting where the objective and constraints are poorly scaled. `SCBO` excels on this problem as the use of the trust region and robust transformations makes it possible to quickly make progress. The domain is $[-3, 5]^5$.

7.12 Restart Frequency of `SCBO`

The `SCBO` algorithm *restarts* with a new trust region if it fails to sample a better point for a certain number of consecutive steps; see Sect. 3 and Sect. 3 of the main document for more details. The average number of iterations between restarts with a new trust region is 42 for the 2D Toy problem, 88 for the 5D Rosenbrock function, 2095 for the 30D Keane function, and 2710 for the 60D Rover problem.

[1] `https://gym.openai.com/envs/LunarLander-v2`

[2] `https://www.sfu.ca/~ssurjano/dixonpr.html`

[3] `https://www.sfu.ca/~ssurjano/levy.html`

References

M. Anjos. The MOPTA 2008 benchmark, 2008. URL `http://www.miguelanjos.com/jones-benchmark`.

B. Charlier, J. Feydy, and J. A. Glaunés. `KeOps`, 2018. URL `https://github.com/getkeops/keops`.

C. A. C. Coello and E. M. Montes. Constraint-handling in genetic algorithms through the use of dominance-based tournament selection. *Advanced Engineering Informatics*, 16(3):193–203, 2002.

K. Dong, D. Eriksson, H. Nickisch, D. Bindel, and A. G. Wilson. Scalable log determinants for Gaussian process kernel learning. In *Advances in Neural Information Processing Systems*, pages 6327–6337, 2017.

D. Eriksson, M. Pearce, J. Gardner, R. D. Turner, and M. Poloczek. Scalable global optimization via local Bayesian optimization. In *Advances in Neural Information Processing Systems 32*, pages 5497–5508. 2019.

J. Gardner, G. Pleiss, K. Q. Weinberger, D. Bindel, and A. G. Wilson. GPyTorch: Blackbox matrix-matrix Gaussian process inference with GPU acceleration. In *Advances in Neural Information Processing Systems*, pages 7576–7586, 2018.

A.-R. Hedar and M. Fukushima. Derivative-free filter simulated annealing method for constrained continuous global optimization. *Journal of Global optimization*, 35(4):521–549, 2006.

J. M. Hernández-Lobato, M. A. Gelbart, R. P. Adams, M. W. Hoffman, and Z. Ghahramani. A general framework for constrained Bayesian optimization using information-based search. *The Journal of Machine Learning Research*, 17(1):5549–5601, 2016. Code available at: `https://github.com/HIPS/Spearmint/tree/PESC`. Last accessed on 02/03/2020.

A. Keane. Experiences with optimizers in structural design. In *Proceedings of the conference on adaptive computing in engineering design and control*, volume 94, pages 14–27, 1994.

A. Lemonge, H. Barbosa, C. Borges, and F. Silva. Constrained optimization problems in mechanical engineering design using a real-coded steady-state genetic algorithm. *Mecánica Computacional*, 29:9287–9303, 2010.

R. G. Regis and C. A. Shoemaker. A stochastic radial basis function method for the global optimization of expensive functions. *INFORMS Journal on Computing*, 19(4):497–509, 2007.

M. Schonlau, W. J. Welch, and D. R. Jones. Global versus local search in constrained optimization of computer models. *Lecture Notes-Monograph Series*, pages 11–25, 1998.

J. Snoek, H. Larochelle, and R. P. Adams. Practical Bayesian optimization of machine learning algorithms. In *Advances in Neural Information Processing Systems*, pages 2951–2959, 2012.

J. C. Spall. *Introduction to stochastic search and optimization: estimation, simulation, and control*, volume 65. John Wiley & Sons, 2005.

S. Ubaru, J. Chen, and Y. Saad. Fast estimation of $tr(f(A))$ via stochastic Lanczos quadrature. *SIAM Journal on Matrix Analysis and Applications*, 38(4): 1075–1099, 2017.

Z. Wang, C. Gehring, P. Kohli, and S. Jegelka. Batched large-scale Bayesian optimization in high-dimensional spaces. In *International Conference on Artificial Intelligence and Statistics*, pages 745–754, 2018.