

Combinatorial Bayesian Optimization and Its Applications

Jungtaek Kim

jungtaek.kim@pitt.edu

University of Pittsburgh

<https://jungtaek.github.io>

September 15, 2022

Table of Contents

Bayesian Optimization

Combinatorial Bayesian Optimization

Applications of Combinatorial Bayesian Optimization

Takeaway

References

Appendices

Bayesian Optimization

How Can We Make a Better Chocolate-Chip Cookie Using Mathematical Optimization or Machine Learning?



Taken from this link.

How Can We Make a Better Chocolate-Chip Cookie Using Mathematical Optimization or Machine Learning?

- ▶ Google Brain Team carried out this real-world optimization problem at their Pittsburgh and Mountain View offices [Kochanski et al., 2017].
- ▶ A goal is to find **an optimal recipe** where a list of ingredients to make cookies and **their search space** are given:
 - ▶ for example, in the third Pittsburgh study,
flour is a fixed quantity,
total sugar, chip quantity, and butter are optimized as continuous variables,
salt, vanilla extract, egg, orange extract, baking soda, and cayenne pepper are optimized as discrete variables,
and chip type, i.e., dark, milk, and white, is optimized as a categorical variable.

How Can We Make a Better Chocolate-Chip Cookie Using Mathematical Optimization or Machine Learning?

- ▶ Cookies are evaluated by [taster's surveys](#).
- ▶ Since baking cookies and evaluating them take much time, the authors employed [Bayesian optimization](#) in their problem.
- ▶ They conducted a pilot experiment for the first Pittsburgh study, 35 trials for the second Pittsburgh study, and 49 trials for the third Pittsburgh study, for 8 days.
- ▶ A recipe of the best-rated Pittsburgh trial is

167 grams of all-purpose flour, 196 grams of dark chocolate chips, 1/2 tsp. baking soda, 1/4 tsp. salt, 1/4 tsp. cayenne pepper, 108 grams of sugar, 30 grams of egg, 129 grams of butter, 3/8 tsp. orange extract, 1/2 tsp. vanilla extract.

Mathematical Optimization

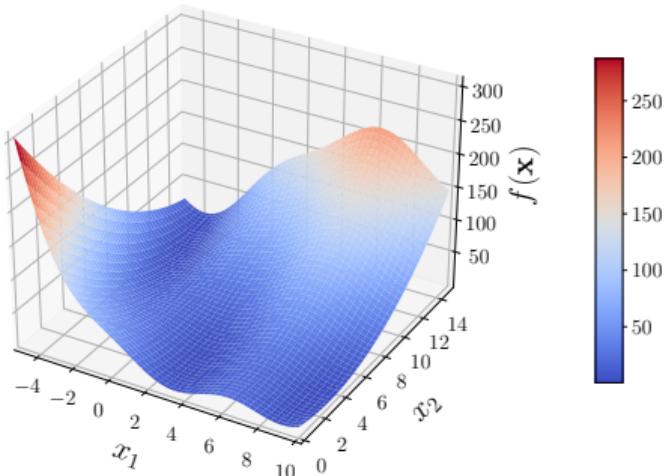


Figure 1: Branin function.

- ▶ Given an objective $f : \mathcal{A} \rightarrow \mathbb{R}$ where \mathcal{A} is some set, it seeks **minimum** or **maximum** of the target function:

$$\mathbf{x}^* = \arg \min f(\mathbf{x}), \quad (1)$$

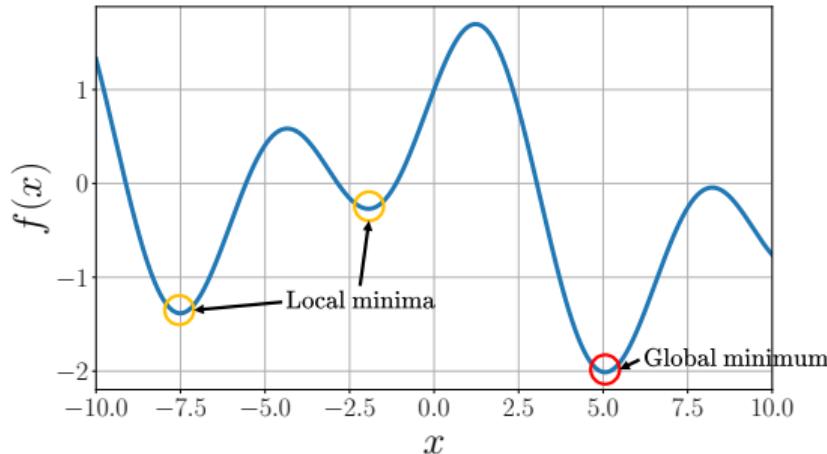
or

$$\mathbf{x}^* = \arg \max f(\mathbf{x}). \quad (2)$$

Mathematical Optimization

- ▶ To optimize an objective, we can select one of such strategies:
 - ▶ random searches;
 - ▶ gradient-based approaches;
 - ▶ convex programming;
 - ▶ evolutionary algorithms;
 - ▶ simulated annealing.
- ▶ Each strategy has the advantage in the corresponding conditions of optimization problem.
- ▶ However, under certain circumstances, **Bayesian optimization** is the most effective method to solve some class of mathematical optimization problems.

Global Optimization



- ▶ Global optimization solves a problem to find a **global minimizer** \mathbf{x}^* :

$$\mathbf{x}^* = \arg \min_{\mathbf{x} \in \mathcal{X}} f(\mathbf{x}), \quad (3)$$

where $\mathcal{X} \subset \mathbb{R}^d$ is a compact search space.

Black-Box Optimization

Definition 1 (Black-box function)

If an objective f , defined in (3), satisfies the following statements, we call it as a black-box function:

- (i) a function f is unknown, but evaluations of f are available;
- (ii) a gradient ∇f and Hessian matrix $\nabla^2 f$ are also unknown;
- (iii) the condition that f is Lipschitz continuous is known;
- (iv) moreover, differentiability and continuity of f are unknown,

on a compact search space \mathcal{X} .

Black-Box Optimization

- ▶ According to recent work [Hansen et al., 2010, Turner et al., 2020], we can apply some classes of possible candidates:
 - ▶ random search [Bergstra and Bengio, 2012];
 - ▶ evolutionary strategies [Hansen, 2006, 2016];
 - ▶ Lipschitzian optimization method without the Lipschitz constant [Jones et al., 1993, Jones and Martins, 2021];
 - ▶ Bayesian optimization [Kushner, 1964, Močkus, 1975];
 - ▶ sequential model-based optimization with tree-based surrogates [Hutter et al., 2011].
- ▶ Unfortunately, there is **no rule of thumb** for choosing the best approach to solving a certain objective without directly conducting the method on the optimization problem.

[Hansen et al., 2010] N. Hansen, A. Auger, R. Ros, S. Finck, and P. Pošík. Comparing results of 31 algorithms from the black-box optimization benchmarking BBOB-2009. In *GECCO*, 2010.

[Turner et al., 2020] R. Turner, D. Eriksson, M. McCourt, J. Kiili, E. Laaksonen, Z. Xu, and I. Guyon. Bayesian optimization is superior to random search for machine learning hyperparameter tuning: Analysis of the black-box optimization challenge 2020. In *NeurIPS Competition and Demonstration Track*, 2020. **11/40**

Bayesian Optimization

- ▶ Bayesian optimization is a promising method to find a global optimizer of black-box objective function.
- ▶ Evaluation of the objective is only available.
- ▶ Since we do not know a target function, it optimizes an acquisition function, instead of the target function.
- ▶ An acquisition function is defined with factors for exploiting available information up to current iteration and exploring an unexplored region.

Surrogate Models

- ▶ A surrogate model estimates a true objective function, where historical evaluations are given.
- ▶ To balance a trade-off between exploration and exploitation, it predicts a function estimate and its uncertainty estimate over any query $\mathbf{x} \in \mathcal{X}$.
- ▶ Gaussian process regression [Rasmussen and Williams, 2006] is widely used as a surrogate model.
- ▶ Also, Student- t process regression [Martinez-Cantin et al., 2018], random forest regression [Hutter et al., 2011], and Bayesian neural network [Springenberg et al., 2016] have been used.

[Rasmussen and Williams, 2006] C. E. Rasmussen and C. K. I. Williams. Gaussian Processes for Machine Learning. MIT Press, 2006.

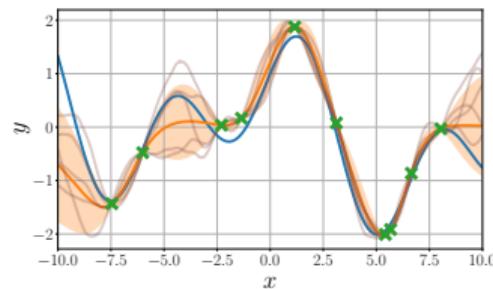
[Martinez-Cantin et al., 2018] R. Martinez-Cantin, K. Tee, and M. McCourt. Practical Bayesian optimization in the presence of outliers. In *AISTATS*, 2018.

[Hutter et al., 2011] F. Hutter, H. H. Hoos, and K. Leyton-Brown. Sequential model-based optimization for general algorithm configuration. In *FGON University of Pittsburgh NeurIPS*, 2011.

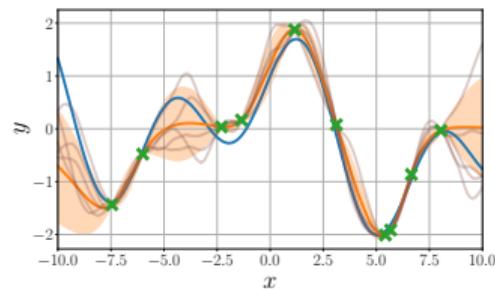
[Springenberg et al., 2016] J. T. Springenberg, A. Klein, S. Falkner, and F. Hutter. Bayesian optimization with robust Bayesian neural networks. In *NeurIPS*, 2016.



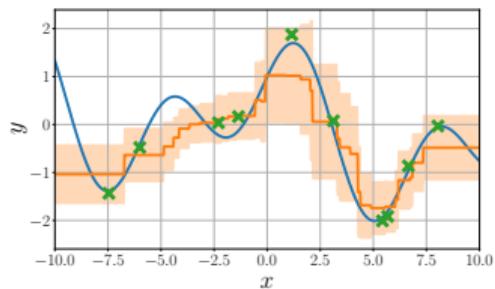
Surrogate Models



(a) Gaussian process



(b) Student- t process



(c) Random forest

Figure 2: Examples of surrogate models.

Acquisition Functions

- ▶ An acquisition function acquires **the next sample to evaluate** by a black-box function f .
 - ▶ It is designed to consider **both exploration and exploitation factors**.
 - ▶ As a popular choice of acquisition functions, the following acquisition functions:
 - ▶ probability of improvement (PI) [Kushner, 1964];
 - ▶ expected improvement (EI) [Močkus et al., 1978];
 - ▶ Gaussian process upper confidence bound (GP-UCB) [Srinivas et al., 2010],
- have been suggested.

[Kushner, 1964] H. J. Kushner. A new method of locating the maximum point of an arbitrary multipeak curve in the presence of noise. *Journal of Basic Engineering*, 86(1): 97–106, 1964.

[Močkus et al., 1978] J. Močkus, V. Tiesis, and A. Žilinskas. The application of Bayesian methods for seeking the extremum. *Towards Global Optimization*, 2:117–129, 1978.

[Srinivas et al., 2010] N. Srinivas, A. Krause, S. Kakade, and M. Seeger. Gaussian process optimization in the bandit setting: No regret and experimental design. In *ICML*, 2010.

Acquisition Functions

- ▶ Diverse acquisition functions have been also proposed:
 - ▶ knowledge gradient [Frazier et al., 2009];
 - ▶ entropy search [Hennig and Schuler, 2012];
 - ▶ predictive entropy search [Hernández-Lobato et al., 2014];
 - ▶ clustering-guided Gaussian process upper confidence bound [Kim and Choi, 2018];
 - ▶ portfolio allocation of various acquisition functions [Hoffman et al., 2011];
 - ▶ alternatives of expected improvement by tree-structured Parzen estimator [Bergstra et al., 2011] and class-probability estimation [Tiao et al., 2021].

[Frazier et al., 2009] P. I. Frazier, W. B. Powell, and S. Dayanik. The knowledge-gradient policy for correlated normal beliefs. *INFORMS Journal on Computing*, 21(4):599–613, 2009.

[Hennig and Schuler, 2012] P. Hennig and C. J. Schuler. Entropy search for information-efficient global optimization. *JMLR*, 13:1809–1837, 2012.

[Hernández-Lobato et al., 2014] J. M. Hernández-Lobato, M. W. Hoffman, and Z. Ghahramani. Predictive entropy search for efficient global optimization of black-box functions. In *NeurIPS*, 2014.

[Kim and Choi, 2018] J. Kim and S. Choi. Clustering-guided GP-UCB for Bayesian optimization. In *ICASSP*, 2018.

[Hoffman et al., 2011] M. Hoffman, E. Brochu, and N. de Freitas. Portfolio allocation for Bayesian optimization. In *UAI*, 2011.

[Bergstra et al., 2011] J. Bergstra, R. Bardenet, Y. Bengio, and B. Kégl. Algorithms for hyper-parameter optimization. In *NeurIPS*, 2011.

Acquisition Function Optimization

- ▶ We should find a **global optimizer** of acquisition function, to determine the next query point.
- ▶ But, in practice, either **local optimizer** or **multi-started local optimizer** can be a good option as a substitute of global optimizer.
- ▶ Analyses on these selections are provided in [Kim and Choi, 2020].
- ▶ The analyses allow us to choose local optimizer or multi-started local optimizer by showing **a bound of instantaneous regret difference** theoretically and empirically.

Overall Procedure of Bayesian Optimization

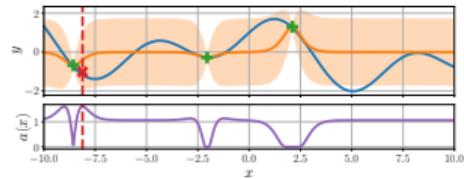
Algorithm 1 Overall Procedure of Bayesian Optimization

Input: A domain of interest $\mathcal{X} \subset \mathbb{R}^d$, an initial set of data \mathcal{D}_0 , an evaluation budget T , and a true unknown objective f .

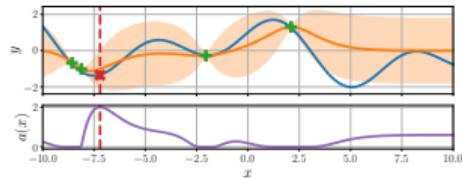
Output: The best optimizer found until T , \mathbf{x}_{best} .

- 1: **for** $t = 1, \dots, T$ **do**
 - 2: Construct a surrogate model $\hat{f}(\mathbf{x}; \mathcal{D}_{t-1})$.
 - 3: Choose the next point to evaluate by maximizing an acquisition function, defined with
 $\hat{f}: \mathbf{x}_t = \arg \max_{\mathbf{x} \in \mathcal{X}} a(\mathbf{x} | \mathcal{D}_{t-1})$.
 - 4: Evaluate \mathbf{x}_t by f : $y_t = f(\mathbf{x}_t) + \epsilon_t$, where ϵ_t is observation noise.
 - 5: Append (\mathbf{x}_t, y_t) to $\mathcal{D}_t = \mathcal{D}_{t-1} \cup \{(\mathbf{x}_t, y_t)\}$.
 - 6: **end for**
 - 7: Determine the best optimizer found until T : $\mathbf{x}_{\text{best}} = \arg \min_{(\mathbf{x}, y) \in \mathcal{D}_T} y$.
 - 8: **return** \mathbf{x}_{best}
-

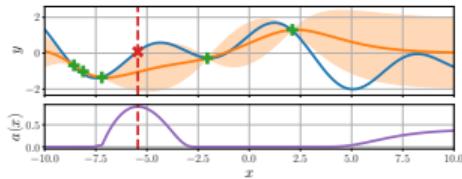
Bayesian Optimization Results with EI



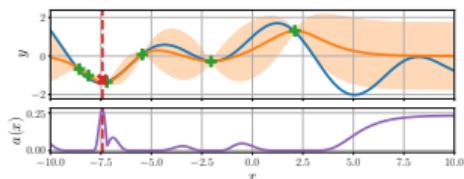
(a) Iteration 1



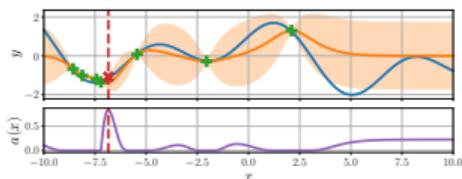
(b) Iteration 2



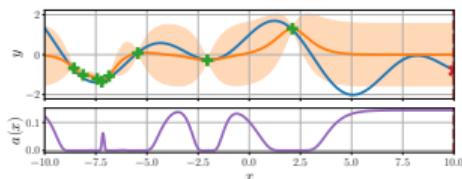
(c) Iteration 3



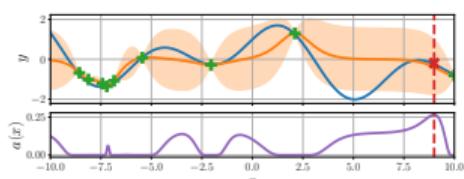
(d) Iteration 4



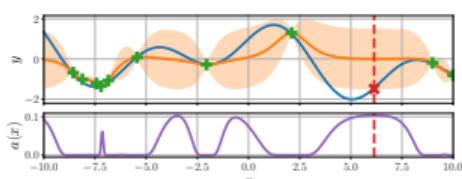
(e) Iteration 5



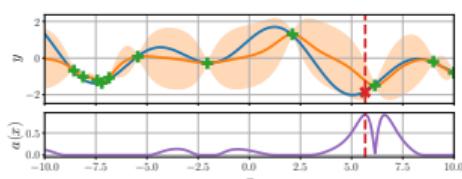
(f) Iteration 6



(g) Iteration 7



(h) Iteration 8



(i) Iteration 9

Figure 3: Bayesian optimization results with EI criterion.



- ▶ Current version: 0.5.3
- ▶ Supported Python version: 3.6, 3.7, 3.8, 3.9, 3.10
- ▶ Web page: <https://bayeso.org>
- ▶ GitHub repository: <https://github.com/jungtaekkim/bayeso>
- ▶ Documentation: <https://bayeso.readthedocs.io>
- ▶ License: MIT license

Combinatorial Bayesian Optimization

Combinatorial Bayesian Optimization

- ▶ Unlike general Bayesian optimization, it attempts to find **an optimal combination** of an objective function defined on a combinatorial space.
- ▶ A **combinatorial space** \mathcal{C} is defined as a space of k categorical (or discrete) variables:

$$\mathcal{C} = \{c_1^{(1)}, \dots, c_1^{(N_1)}\} \times \cdots \times \{c_k^{(1)}, \dots, c_k^{(N_k)}\}, \quad (4)$$

where $c_i^{(j)}$ is category j of dimension i .

- ▶ The cardinality of the combinatorial space \mathcal{C} is

$$|\mathcal{C}| = N_1 \times \cdots \times N_k = N. \quad (5)$$

- ▶ We consider the problem of minimizing a black-box function $f : \mathcal{C} \rightarrow \mathbb{R}$:

$$\mathbf{c}^* = \arg \min_{\mathbf{c} \in \mathcal{C}} f(\mathbf{c}). \quad (6)$$

Combinatorial Bayesian Optimization

- ▶ We can follow a general procedure of Bayesian optimization, but **additional treatments** are required.
- ▶ Challenges to combinatorial Bayesian optimization are
 - ▶ a surrogate model defined on \mathcal{C} ;
 - ▶ acquisition function optimization on \mathcal{C} ;
 - ▶ if we do not solve a problem on \mathcal{C} directly, a mapping to \mathbb{R}^d and a recovery to \mathcal{C} ;
 - ▶ if we use a Boolean representation, thresholding to binary variables;
 - ▶ if we use a one-hot encoded representation, a sparsity-inducing model.

Bayesian Optimization of Combinatorial Structures

- ▶ It solves a combinatorial optimization problem on $\mathcal{B} = \{0, 1\}^d$; $\mathbf{c} \in \mathcal{C}$ can be mapped to $\mathbf{b} \in \mathcal{B}$ without loss of generality.
- ▶ It uses the following second-order model:

$$f_{\boldsymbol{\alpha}}(\mathbf{x}) = \alpha_0 + \sum_j \alpha_j x_j + \sum_{i,j>i} \alpha_{ij} x_i x_j, \quad (7)$$

as a surrogate model, where $\mathbf{x} \in \mathcal{B}$ and $\boldsymbol{\alpha} \in \mathbb{R}^p$. Note that $p = 1 + d + \binom{d}{2}$.

- ▶ A Bayesian treatment with a sparsity-inducing prior for $\boldsymbol{\alpha}$ is employed.
- ▶ An acquisition function, which is inspired by Thompson sampling [Thompson, 1933], is used.
- ▶ To induce sparsity more, l_1 norm of \mathbf{x} is added in the acquisition function.

Combinatorial Bayesian Optimization using the Graph Cartesian Product

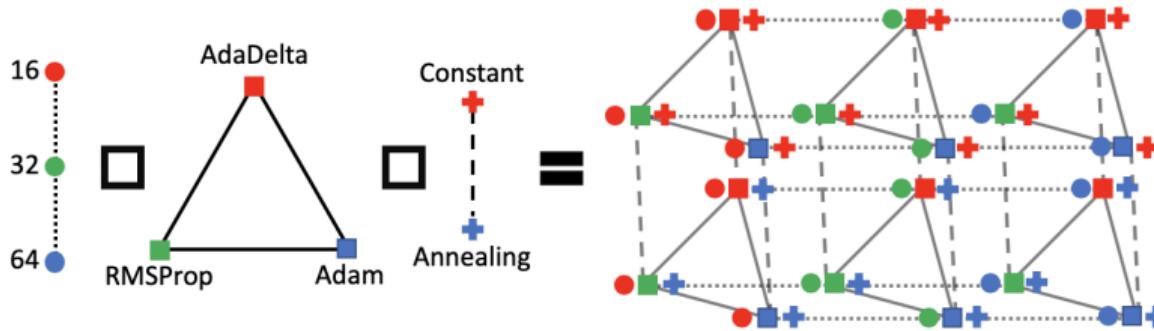


Figure 4: Combinatorial graph, i.e., graph Cartesian product of sub-graphs.

- ▶ This work defines a **combinatorial graph** that models \mathcal{C} directly.
- ▶ It uses Gaussian process regression with **diffusion kernels on graphs**.

Combinatorial Bayesian Optimization using the Graph Cartesian Product

- ▶ The eigendecomposition with graph Cartesian products is used to speed up computing the diffusion kernels.
- ▶ An automatic relevance determination technique is applied for weighting sub-graphs.
- ▶ Since a search space is a combinatorial graph, an acquisition function is optimized by using the breadth-first local search from a sufficient number of initial vertices.
- ▶ Twenty vertices with highest acquisition function values among 20,000 randomly sampled vertices are used as initial vertices.

Combining Latent Space and Structured Kernels for Bayesian Optimization over Combinatorial Spaces

- ▶ This work utilizes a deep generative model, which is composed of encoder and decoder.
- ▶ A combination $\mathbf{c} \in \mathcal{C}$ maps to $\mathbf{x} \in \mathbb{R}^d$ using a pretrained encoder of deep generative model.
- ▶ To incorporate information about the decoded structure, the authors define a structure-coupled kernel, which is used to construct a Gaussian process regression model.
- ▶ We can follow the conventional procedure of Bayesian optimization with the novel Gaussian process regression that is defined on \mathbb{R}^d and considers the decoded structure.

Combining Latent Space and Structured Kernels for Bayesian Optimization over Combinatorial Spaces

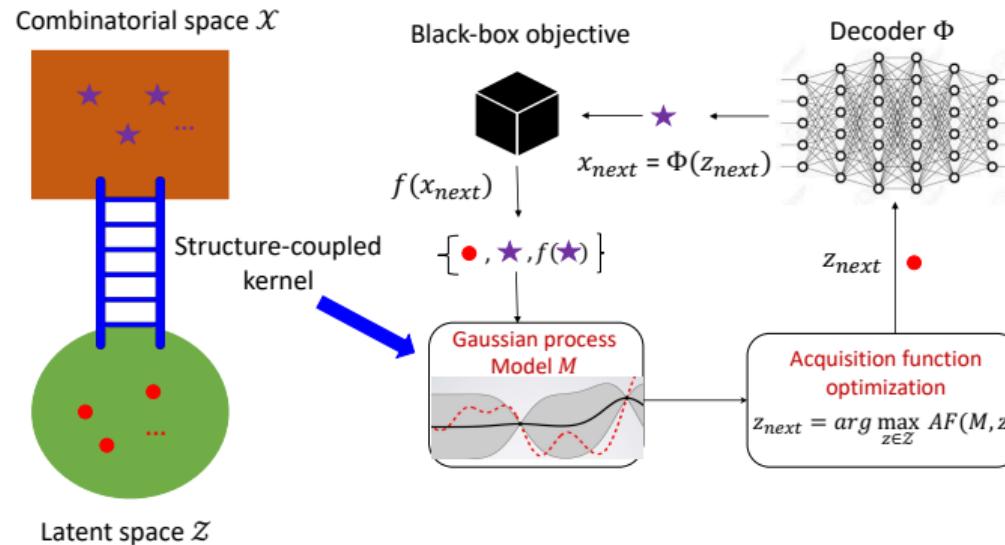


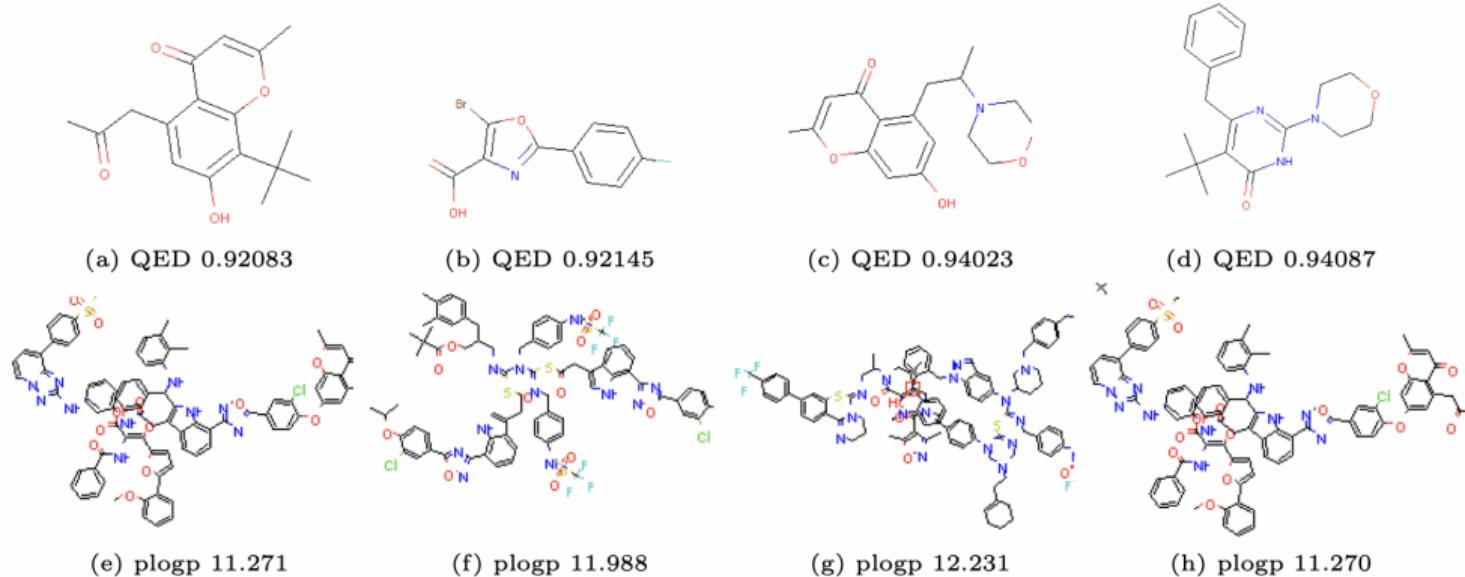
Figure 5: High-level conceptual illustration of the LADDER approach.

Combinatorial Bayesian Optimization with Random Mapping Functions to Convex Polytopes

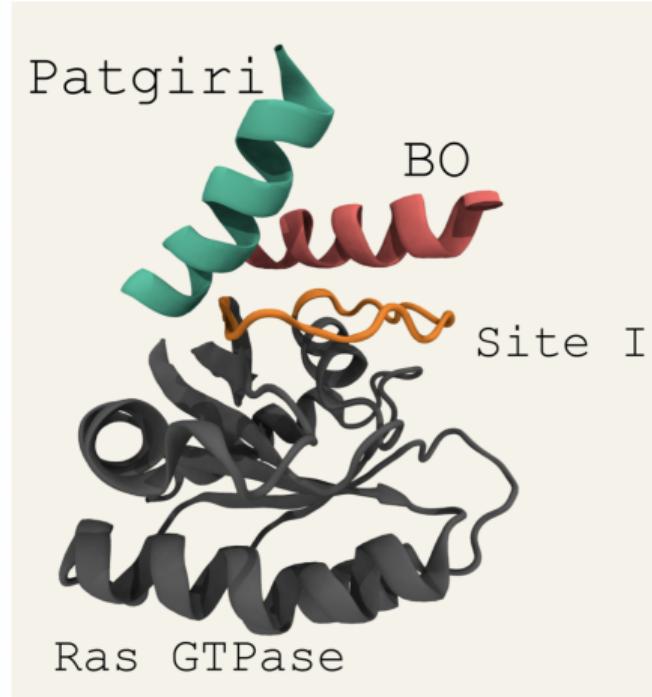
- ▶ To handle a combinatorial variable $\mathbf{c} \in \mathcal{C}$, it first maps \mathbf{c} to $\mathbf{x} \in \mathcal{X} \subset \mathbb{R}^d$.
- ▶ To map \mathbf{c} to \mathbf{x} , we use **a random matrix**, which is capable of preserving a similarity on the original space by the modified Johnson-Lindenstrauss lemma.
- ▶ By applying the random mapping function, we are able to define a surrogate model on \mathcal{X} and optimize an acquisition function through the conventional Bayesian optimization.
- ▶ To recover \mathbf{x}^\dagger to \mathbf{c}^\dagger , we use **a lookup table**.
- ▶ The lookup table is preemptively constructed by the random matrix and all $\mathbf{c} \in \mathcal{C}$.

Applications of Combinatorial Bayesian Optimization

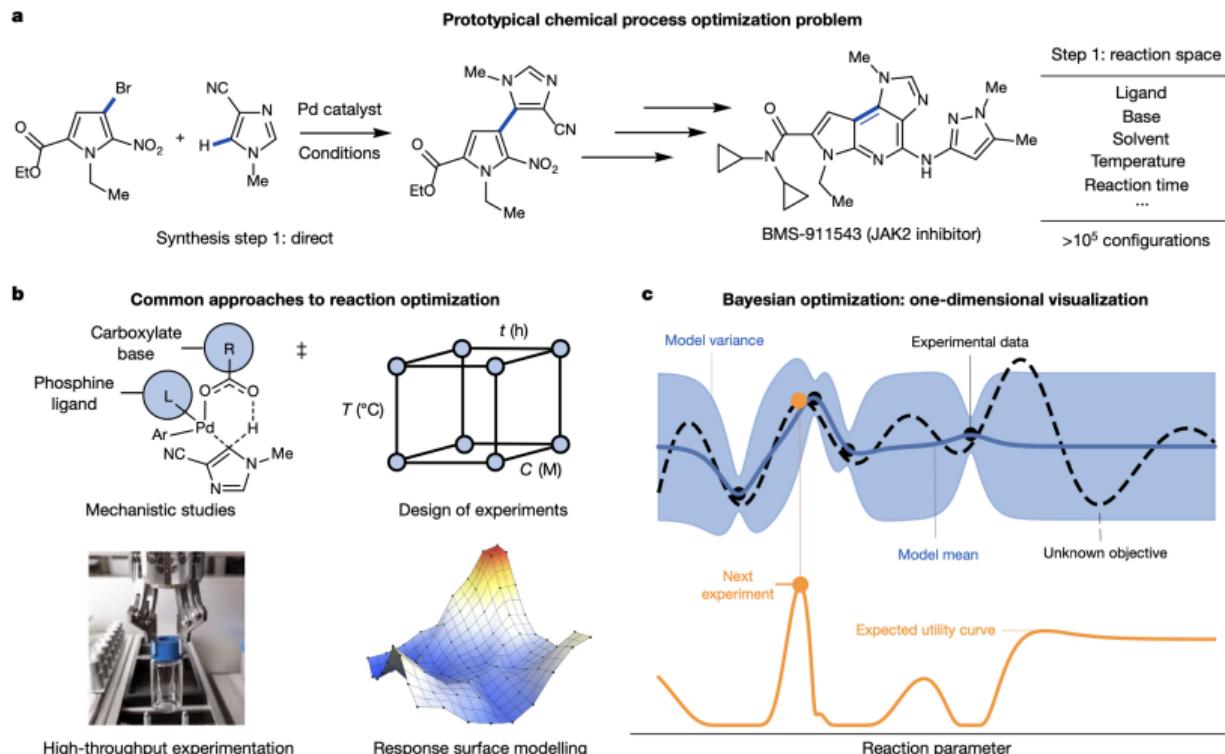
Molecule Design



Protein Structure Design

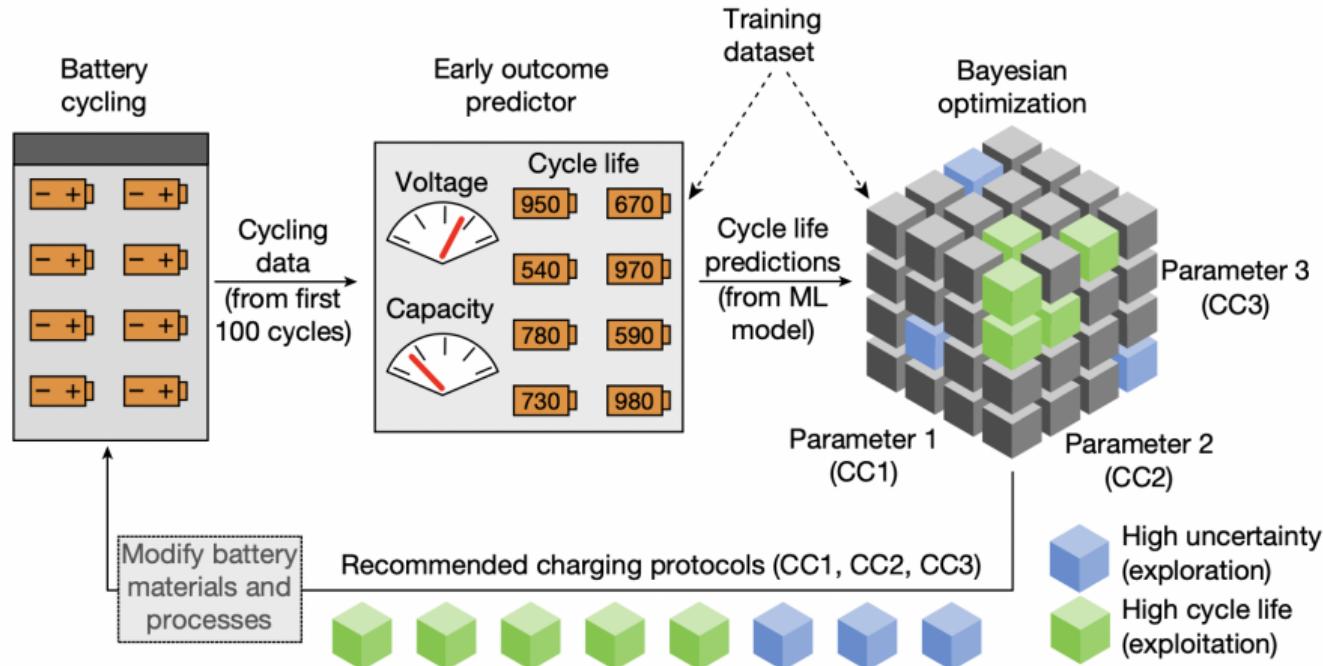


Chemical Reaction Optimization



[Shields et al., 2021] B. J. Shields, J. Stevens, J. Li, M. Parasram, F. Damani, J. I. M. Alvarado, J. M. Janey, R. P. Adams, and A. G. Doyle. Bayesian reaction optimization as a tool for chemical synthesis. *Nature*, 590:89–96, 2021.

Battery Lifetime Optimization



Sequential Assembly

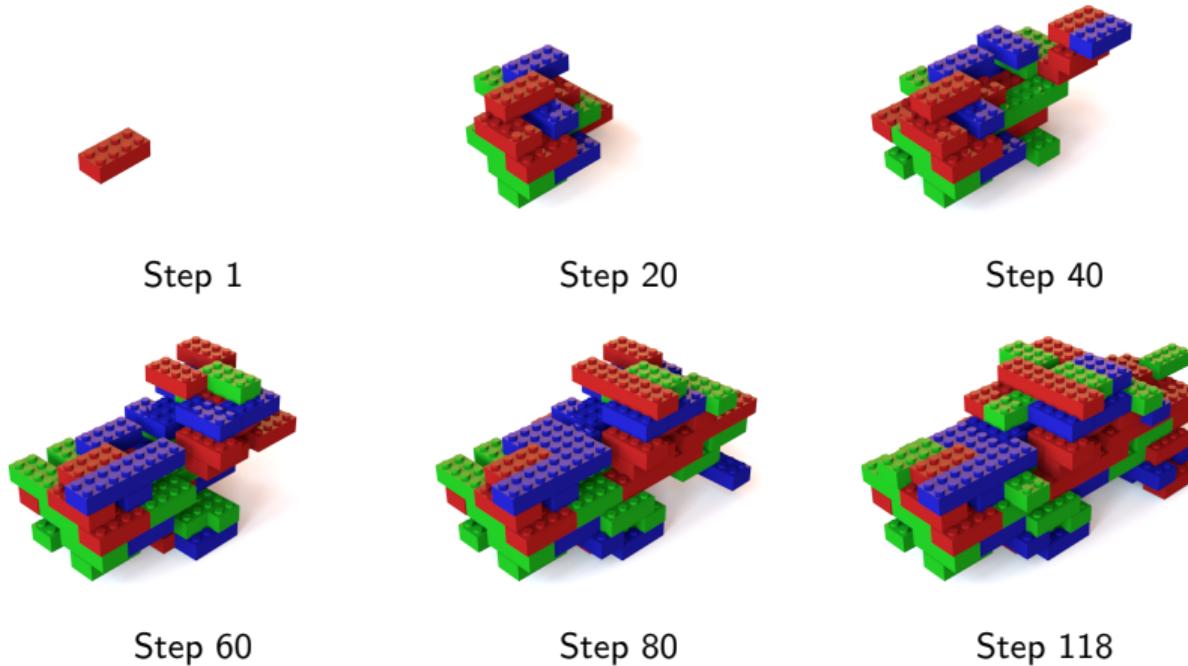


Figure 6: Generated assembling sequence that creates a *car* shape with 118 unit primitives.

Takeaway

- ▶ Bayesian optimization is a powerful method to optimize a black-box function.
- ▶ Instead of methods based on heuristic or prior knowledge, it provides a structured approach to finding an optimal solution.
- ▶ Combinatorial Bayesian optimization is more challenging than the general Bayesian optimization, but there are much more interesting applications.
- ▶ The potential of Bayesian optimization has not been fully exploited yet :)

Thank you!

References I

- P. M. Attia, A. Grover, N. Jin, K. A. Severson, T. M. Markov, Y.-H. Liao, M. H. Chen, B. Cheong, N. Perkins, Z. Yang, P. K. Herring, M. Aykol, S. J. Harris, R. D. Braatz, S. Ermon, and W. C. Chueh. Closed-loop optimization of fast-charging protocols for batteries with machine learning. *Nature*, 578(7795):397–402, 2020.
- R. Baptista and M. Poloczek. Bayesian optimization of combinatorial structures. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 462–471, Stockholm, Sweden, 2018.
- J. Bergstra and Y. Bengio. Random search for hyper-parameter optimization. *Journal of Machine Learning Research*, 13:281–305, 2012.
- J. Bergstra, R. Bardenet, Y. Bengio, and B. Kégl. Algorithms for hyper-parameter optimization. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 24, pages 2546–2554, Granada, Spain, 2011.
- A. Deshwal and J. Doppa. Combining latent space and structured kernels for Bayesian optimization over combinatorial spaces. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 34, pages 8185–8200, Virtual, 2021.
- P. I. Frazier, W. B. Powell, and S. Dayanik. The knowledge-gradient policy for correlated normal beliefs. *INFORMS Journal on Computing*, 21(4):599–613, 2009.
- N. Hansen. The CMA evolution strategy: a comparing review. *Towards a new evolutionary computation*, pages 75–102, 2006.
- N. Hansen. The CMA evolution strategy: A tutorial. *arXiv preprint arXiv:1604.00772*, 2016.
- N. Hansen, A. Auger, R. Ros, S. Finck, and P. Pošík. Comparing results of 31 algorithms from the black-box optimization benchmarking BBOB-2009. In *Proceedings of the Annual Conference on Genetic and Evolutionary Computation (GECCO)*, pages 1689–1696, Portland, Oregon, USA, 2010.
- P. Hennig and C. J. Schuler. Entropy search for information-efficient global optimization. *Journal of Machine Learning Research*, 13:1809–1837, 2012.
- J. M. Hernández-Lobato, M. W. Hoffman, and Z. Ghahramani. Predictive entropy search for efficient global optimization of black-box functions. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 27, pages 918–926, Montreal, Quebec, Canada, 2014.
- M. Hoffman, E. Brochu, and N. de Freitas. Portfolio allocation for Bayesian optimization. In *Proceedings of the Annual Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 327–336, Barcelona, Spain, 2011.
- F. Hutter, H. H. Hoos, and K. Leyton-Brown. Sequential model-based optimization for general algorithm configuration. In *Proceedings of the International Conference on Learning and Intelligent Optimization (LION)*, pages 507–523, Rome, Italy, 2011.
- D. R. Jones and J. R. R. A. Martins. The DIRECT algorithm: 25 years later. *Journal of Global Optimization*, 79(3):521–566, 2021.

References II

- D. R. Jones, C. D. Perttunen, and B. E. Stuckman. Lipschitzian optimization without the Lipschitz constant. *Journal of Optimization Theory and Applications*, 79(1):157–181, 1993.
- J. Kim and S. Choi. BayesO: A Bayesian optimization framework in Python. <https://bayeso.org>, 2017.
- J. Kim and S. Choi. Clustering-guided GP-UCB for Bayesian optimization. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages 2461–2465, Calgary, Alberta, Canada, 2018.
- J. Kim and S. Choi. On local optimizers of acquisition functions in Bayesian optimization. In *Proceedings of the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML-PKDD)*, pages 675–690, Virtual, 2020.
- J. Kim, H. Chung, J. Lee, M. Cho, and J. Park. Combinatorial 3D shape generation via sequential assembly. In *Neural Information Processing Systems Workshop on Machine Learning for Engineering Modeling, Simulation, and Design (ML4Eng)*, Virtual, 2020.
- J. Kim, S. Choi, and M. Cho. Combinatorial Bayesian optimization with random mapping functions to convex polytopes. In *Proceedings of the Annual Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 1001–1011, Eindhoven, the Netherlands, 2022.
- G. Kochanski, D. Golovin, J. Karro, B. Solnik, S. Moitra, and D. Sculley. Bayesian optimization for a better dessert. In *Neural Information Processing Systems Workshop on Bayesian Optimization (BayesOpt)*, Long Beach, California, USA, 2017.
- K. Korovina, S. Xu, K. Kandasamy, W. Neiswanger, B. Póczos, J. Schneider, and E. P. Xing. ChemBO: Bayesian optimization of small organic molecules with synthesizable recommendations. In *Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 3393–3403, Virtual, 2020.
- H. J. Kushner. A new method of locating the maximum point of an arbitrary multipeak curve in the presence of noise. *Journal of Basic Engineering*, 86(1):97–106, 1964.
- R. Martinez-Cantin, K. Tee, and M. McCourt. Practical Bayesian optimization in the presence of outliers. In *Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 1722–1731, Lanzarote, Canary Islands, Spain, 2018.
- J. Močkus. On Bayesian methods for seeking the extremum. In *Optimization Techniques IFIP Technical Conference*, pages 400–404, Novosibirsk, Russia, 1975.
- J. Močkus, V. Tesis, and A. Žilinskas. The application of Bayesian methods for seeking the extremum. *Towards Global Optimization*, 2:117–129, 1978.
- C. Oh, J. Tomczak, E. Gavves, and M. Welling. Combinatorial Bayesian optimization using the graph cartesian product. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 32, pages 2914–2924, Vancouver, British Columbia, Canada, 2019.

References III

- C. E. Rasmussen and C. K. I. Williams. *Gaussian Processes for Machine Learning*. MIT Press, 2006.
- B. J. Shields, J. Stevens, J. Li, M. Parasram, F. Damani, J. I. M. Alvarado, J. M. Janey, R. P. Adams, and A. G. Doyle. Bayesian reaction optimization as a tool for chemical synthesis. *Nature*, 590:89–96, 2021.
- J. T. Springenberg, A. Klein, S. Falkner, and F. Hutter. Bayesian optimization with robust Bayesian neural networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 29, pages 4134–4142, Barcelona, Spain, 2016.
- N. Srinivas, A. Krause, S. Kakade, and M. Seeger. Gaussian process optimization in the bandit setting: No regret and experimental design. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 1015–1022, Haifa, Israel, 2010.
- W. R. Thompson. On the likelihood that one unknown probability exceeds another in view of the evidence of two samples. *Biometrika*, 25(3/4):285–294, 1933.
- L. C. Tiao, A. Klein, M. Seeger, E. V. Bonilla, C. Archambeau, and F. Ramos. BORE: Bayesian optimization by density-ratio estimation. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 10289–10300, Virtual, 2021.
- R. Turner, D. Eriksson, M. McCourt, J. Kiili, E. Laaksonen, Z. Xu, and I. Guyon. Bayesian optimization is superior to random search for machine learning hyperparameter tuning: Analysis of the black-box optimization challenge 2020. In *Proceedings of the NeurIPS 2020 Competition and Demonstration Track*, pages 3–26, Virtual, 2020.
- Z. Yang, K. A. Milas, and A. D. White. Now what sequence? pre-trained ensembles for Bayesian optimization of protein sequences. *bioRxiv*, 2022.

Gaussian Process

- ▶ A collection of random variables, any finite number of which have a joint Gaussian distribution [Rasmussen and Williams, 2006].
- ▶ Generally, a Gaussian process is defined as

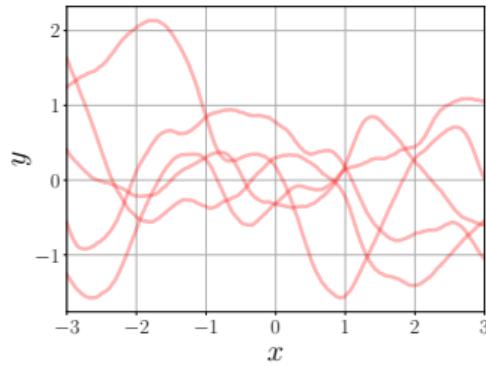
$$f \sim \mathcal{GP}(m(\mathbf{x}), k(\mathbf{x}, \mathbf{x}')), \quad (8)$$

where

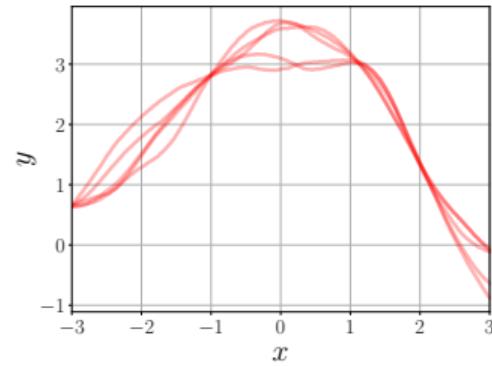
$$m(\mathbf{x}) = \mathbb{E}[f(\mathbf{x})], \quad (9)$$

$$k(\mathbf{x}, \mathbf{x}') = \mathbb{E}[(f(\mathbf{x}) - m(\mathbf{x}))(f(\mathbf{x}') - m(\mathbf{x}'))]. \quad (10)$$

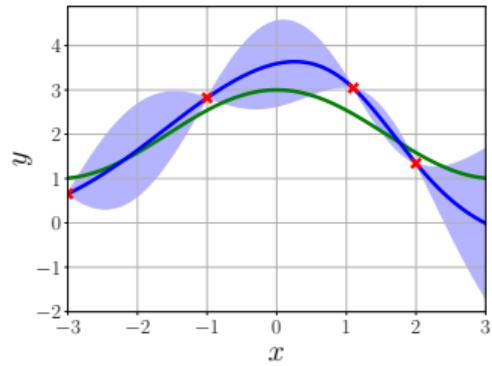
Gaussian Process Regression



(a) From prior function dist.



(b) From posterior function dist.



(c) Predictive dist.

Figure 7: Gaussian process regression for a function $\cos(x) + 2$ with an observation noise.

Gaussian Process Regression

- ▶ One of popular covariance functions, the exponentiated quadratic covariance function in one dimension is defined as

$$k(x, x') = s^2 \exp\left(-\frac{1}{2l^2}(x - x')^2\right) + \sigma_n^2 \delta_{xx'}, \quad (11)$$

where s is a signal scale, l is a length scale and σ_n^2 is a noise variance [Rasmussen and Williams, 2006].

- ▶ Posterior mean function $\mu(\mathbf{x}^*; \mathbf{X}, \mathbf{y})$ and variance function $\sigma^2(\mathbf{x}^*; \mathbf{X}, \mathbf{y})$:

$$\mu(\mathbf{x}^*; \mathbf{X}, \mathbf{y}) = \mathbf{k}(\mathbf{x}^*, \mathbf{X})(\mathbf{K}(\mathbf{X}, \mathbf{X}) + \sigma_n^2 \mathbf{I})^{-1} \mathbf{y}, \quad (12)$$

$$\sigma^2(\mathbf{x}^*; \mathbf{X}, \mathbf{y}) = k(\mathbf{x}^*, \mathbf{x}^*) - \mathbf{k}(\mathbf{x}^*, \mathbf{X})(\mathbf{K}(\mathbf{X}, \mathbf{X}) + \sigma_n^2 \mathbf{I})^{-1} \mathbf{k}(\mathbf{X}, \mathbf{x}^*), \quad (13)$$

where $\mathbf{X} \in \mathbb{R}^{n \times d}$ and $\mathbf{y} \in \mathbb{R}^n$.

Gaussian Process Regression

- ▶ If non-zero mean prior is given, posterior mean and variance functions:

$$\mu(\mathbf{x}^*; \mathbf{X}, \mathbf{y}) = \mathbf{k}(\mathbf{x}^*, \mathbf{X})(\mathbf{K}(\mathbf{X}, \mathbf{X}) + \sigma_n^2 \mathbf{I})^{-1}(\mathbf{y} - \boldsymbol{\mu}_p(\mathbf{X})) + \mu_p(\mathbf{x}^*), \quad (14)$$

$$\sigma^2(\mathbf{x}^*; \mathbf{X}, \mathbf{y}) = k(\mathbf{x}^*, \mathbf{x}^*) - \mathbf{k}(\mathbf{x}^*, \mathbf{X})(\mathbf{K}(\mathbf{X}, \mathbf{X}) + \sigma_n^2 \mathbf{I})^{-1}\mathbf{k}(\mathbf{X}, \mathbf{x}^*), \quad (15)$$

where μ_p is a prior mean function, and $\boldsymbol{\mu}_p(\mathbf{X}) = [\mu_p(\mathbf{x}_1), \dots, \mu_p(\mathbf{x}_n)]$.

Student-*t* Process Regression

- ▶ If non-zero mean prior is given, posterior mean and variance functions:

$$\mu(\mathbf{x}^*; \mathbf{X}, \mathbf{y}) = \mathbf{k}(\mathbf{x}^*, \mathbf{X})\tilde{\mathbf{K}}^{-1}\tilde{\mathbf{y}} + \mu_p(\mathbf{x}^*), \quad (16)$$

$$\sigma^2(\mathbf{x}^*; \mathbf{X}, \mathbf{y}) = \frac{\nu + \tilde{\mathbf{y}}^\top \tilde{\mathbf{K}}^{-1} \tilde{\mathbf{y}} - 2}{\nu + n - 2} \left(k(\mathbf{x}^*, \mathbf{x}^*) - \mathbf{k}(\mathbf{x}^*, \mathbf{X})\tilde{\mathbf{K}}^{-1}\mathbf{k}(\mathbf{X}, \mathbf{x}^*) \right), \quad (17)$$

where μ_p is a prior mean function, $\boldsymbol{\mu}_p(\mathbf{X}) = [\mu_p(\mathbf{x}_1), \dots, \mu_p(\mathbf{x}_n)]$, $\tilde{\mathbf{y}} = \mathbf{y} - \boldsymbol{\mu}_p(\mathbf{X})$, and $\tilde{\mathbf{K}} = \mathbf{K}(\mathbf{X}, \mathbf{X}) + \sigma_n^2 \mathbf{I}$.

- ▶ The parameter ν for the posterior distribution is set to $\nu + n$.

Random Forest Regression

- Posterior mean and variance functions:

$$\begin{aligned}\mu(\mathbf{x}^*; \{\mathcal{T}_b\}_{b=1}^B, \mathbf{X}, \mathbf{y}) &= \frac{1}{B} \sum_{b=1}^B \mu_b(\mathbf{x}^*) \\ &= \frac{1}{B} \sum_{b=1}^B \sum_{\tau \in \boldsymbol{\tau}_{b,l}} \mu_\tau 1_{\mathbf{x}^* \in \tau},\end{aligned}\tag{18}$$

$$\begin{aligned}\sigma^2(\mathbf{x}^*; \{\mathcal{T}_b\}_{b=1}^B, \mathbf{X}, \mathbf{y}) &= \frac{1}{B} \sum_{b=1}^B (\sigma_b^2(\mathbf{x}^*) + \mu_b^2(\mathbf{x}^*)) - \mu(\mathbf{x}^*; \{\mathcal{T}_b\}_{b=1}^B, \mathbf{X}, \mathbf{y})^2 \\ &= \frac{1}{B} \sum_{b=1}^B \left(\left(\sum_{\tau \in \boldsymbol{\tau}_{b,l}} \sigma_\tau 1_{\mathbf{x}^* \in \tau} \right)^2 + \left(\sum_{\tau \in \boldsymbol{\tau}_{b,l}} \mu_\tau 1_{\mathbf{x}^* \in \tau} \right)^2 \right) \\ &\quad - \left(\frac{1}{B} \sum_{b=1}^B \mu_b(\mathbf{x}^*) \right)^2.\end{aligned}\tag{19}$$

Popular Acquisition Functions (Minimization Case)

- ▶ Suppose that

$$(\mathbf{x}^\dagger, y^\dagger) = \arg \min_{(\mathbf{x}, y) \in \mathcal{D}_{t-1}} y, \quad (20)$$

$$\mu(\mathbf{x}; \mathbf{X}, \mathbf{y}) = \mu(\mathbf{x}; \mathcal{D}_{t-1}), \quad (21)$$

$$\sigma(\mathbf{x}; \mathbf{X}, \mathbf{y}) = \sigma(\mathbf{x}; \mathcal{D}_{t-1}). \quad (22)$$

- ▶ PI criterion [Kushner, 1964] is defined as

$$a_{\text{PI}}(\mathbf{x} \mid \mathcal{D}_{t-1}) = \begin{cases} \Phi\left(\frac{y^\dagger - \mu(\mathbf{x}; \mathcal{D}_{t-1})}{\sigma(\mathbf{x}; \mathcal{D}_{t-1})}\right) & \text{if } \sigma^2(\mathbf{x}; \mathcal{D}_{t-1}) > 0, \\ 0 & \text{otherwise,} \end{cases} \quad (23)$$

where Φ is a cumulative distribution function of the standard normal distribution.

Popular Acquisition Functions (Minimization Case)

- ▶ EI criterion [Močkus et al., 1978] is defined as

$$a_{\text{EI}}(\mathbf{x} \mid \mathcal{D}_{t-1}) = \begin{cases} \sigma(\mathbf{x}; \mathcal{D}_{t-1}) (z(\mathbf{x})\Phi(z(\mathbf{x})) + \phi(z(\mathbf{x}))) & \text{if } \sigma^2(\mathbf{x}; \mathcal{D}_{t-1}) > 0, \\ 0 & \text{otherwise,} \end{cases} \quad (24)$$

where $z(\mathbf{x}) = \frac{y^\dagger - \mu(\mathbf{x}; \mathcal{D}_{t-1})}{\sigma(\mathbf{x}; \mathcal{D}_{t-1})}$, Φ is a cumulative distribution function of the standard normal distribution, and ϕ is a probability density function of the standard normal distribution.

- ▶ GP-UCB criterion [Srinivas et al., 2010] is defined as

$$a_{\text{UCB}}(\mathbf{x} \mid \mathcal{D}_{t-1}) = -\mu(\mathbf{x}; \mathcal{D}_{t-1}) + \beta_t \sigma(\mathbf{x}; \mathcal{D}_{t-1}), \quad (25)$$

where β_t is a trade-off hyperparameter at iteration t .

On Local Optimizers of Acquisition Functions in Bayesian Optimization

Theorem 2 (Instantaneous regret difference between global and local optimizers)

Given $\delta_l \in [0, 1]$ and $\epsilon_l, \epsilon_1, \epsilon_2 > 0$, the regret difference for a local optimizer $\mathbf{x}_{t,l}$ at iteration t , $|r_{t,g} - r_{t,l}|$ is less than ϵ_l with a probability at least $1 - \delta_l$:

$$\mathbb{P}(|r_{t,g} - r_{t,l}| < \epsilon_l) \geq 1 - \delta_l, \quad (26)$$

where $\delta_l = \frac{\gamma}{\epsilon_1}(1 - \beta_g) + \frac{M}{\epsilon_2}$, $\epsilon_l = \epsilon_1 \epsilon_2$, $\gamma = \max_{\mathbf{x}_i, \mathbf{x}_j \in \mathcal{X}} \|\mathbf{x}_i - \mathbf{x}_j\|_2$ is the size of \mathcal{X} , β_g is the probability that a local optimizer of the acquisition function collapses with its global optimizer, and M is the Lipschitz constant.

On Local Optimizers of Acquisition Functions in Bayesian Optimization

Theorem 3 (Instantaneous regret difference between global and multi-started local optimizers)

Given $\delta_m \in [0, 1)$ and $\epsilon_m, \epsilon_2, \epsilon_3 > 0$, a regret difference for a multi-started local optimizer $\mathbf{x}_{t,m}$, determined by starting from N initial points at iteration t , is less than ϵ_m with a probability at least $1 - \delta_m$:

$$\mathbb{P}(|r_{t,g} - r_{t,m}| < \epsilon_m) \geq 1 - \delta_m, \quad (27)$$

where $\delta_m = \frac{\gamma}{\epsilon_3} (1 - \beta_g)^N + \frac{M}{\epsilon_2}$, $\epsilon_m = \epsilon_2 \epsilon_3$, $\gamma = \max_{\mathbf{x}_i, \mathbf{x}_j \in \mathcal{X}} \|\mathbf{x}_i - \mathbf{x}_j\|_2$ is the size of \mathcal{X} , β_g is the probability that a local optimizer of the acquisition function collapses with its global optimizer, and M is the Lipschitz constant.

- ▶ By following our intuition, this bound is tighter than the bound provided in Theorem 2.

On Local Optimizers of Acquisition Functions in Bayesian Optimization

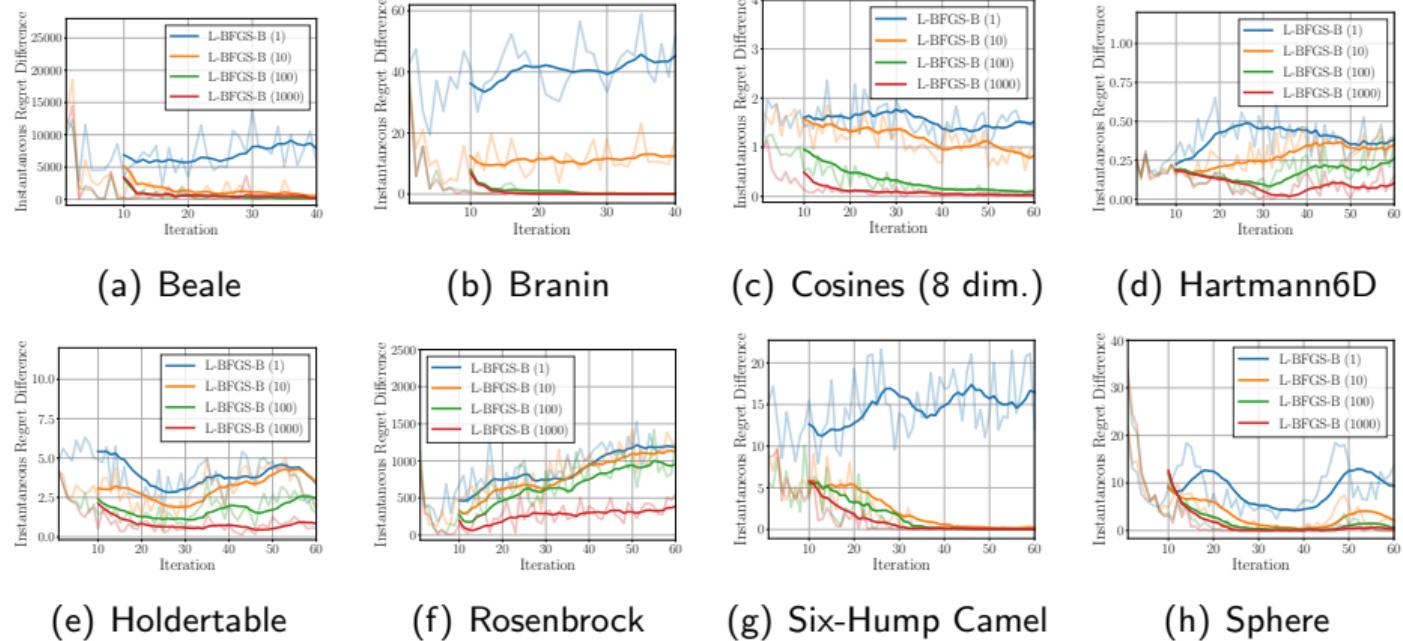


Figure 8: Empirical results on Theorems 2 and 3.

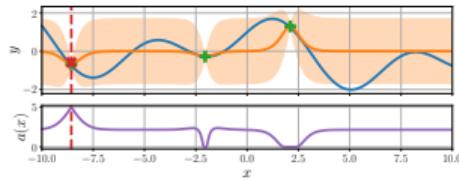
On Local Optimizers of Acquisition Functions in Bayesian Optimization

Table 1: Time (sec.) consumed in optimizing acquisition functions.

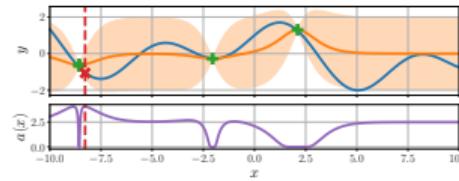
	Beale	Branin	Cosines (8 dim.)	Hart- mann6D	Holder- table	Rosen- brock	Six-Hump Camel	Sphere
DIRECT	3.434	2.987	2.508	0.728	2.935	13.928	4.639	10.707
L-BFGS-B (1)	0.010	0.004	0.023	0.026	0.017	0.005	0.010	0.030
L-BFGS-B (10)	0.096	0.036	0.224	0.253	0.177	0.050	0.100	0.311
L-BFGS-B (100)	0.977	0.363	2.224	2.533	1.760	0.504	0.969	3.048
L-BFGS-B (1000)	9.720	3.633	22.306	25.305	17.629	5.049	9.682	30.764

- ▶ Multi-started local optimizer provides a more efficient approach than global optimizer, in terms of computational complexities.

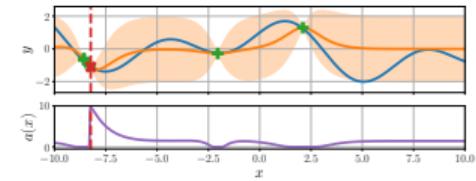
Bayesian Optimization Results with PI



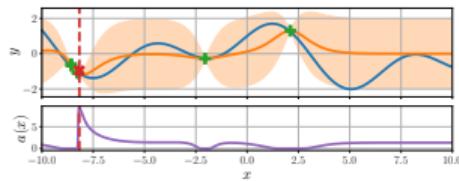
(a) Iteration 1



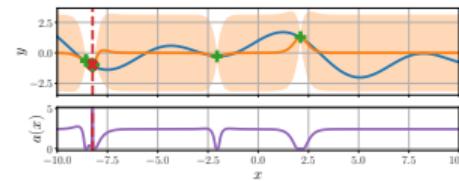
(b) Iteration 2



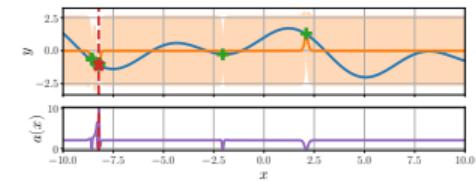
(c) Iteration 3



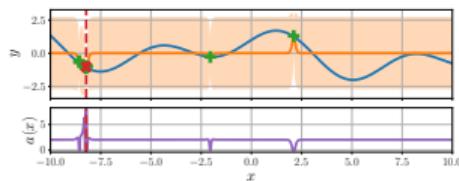
(d) Iteration 4



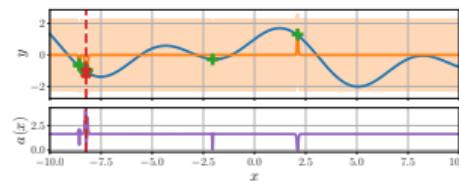
(e) Iteration 5



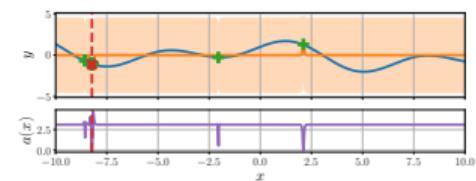
(f) Iteration 6



(g) Iteration 7



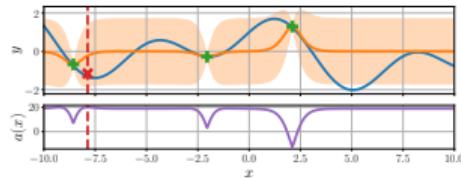
(h) Iteration 8



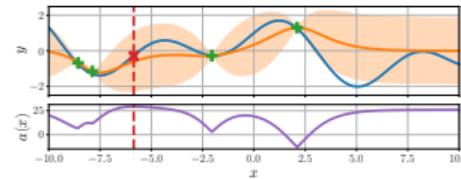
(i) Iteration 9

Figure 9: Bayesian optimization results with PI criterion.

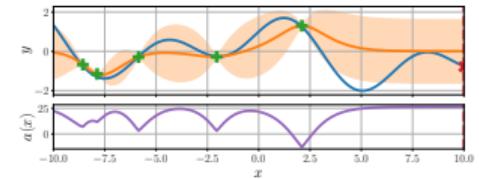
Bayesian Optimization Results with GP-UCB



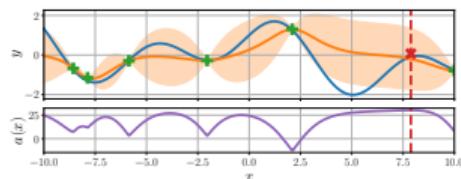
(a) Iteration 1



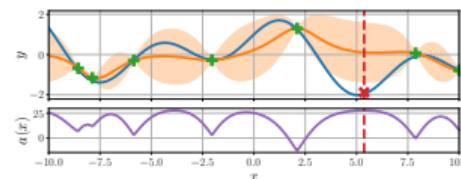
(b) Iteration 2



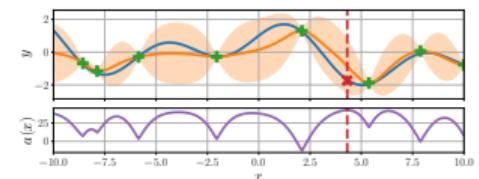
(c) Iteration 3



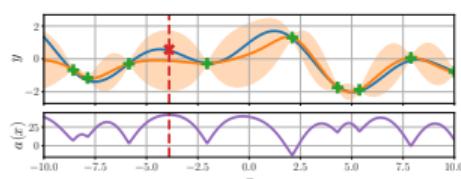
(d) Iteration 4



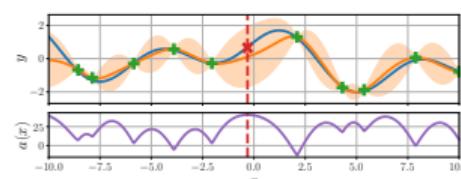
(e) Iteration 5



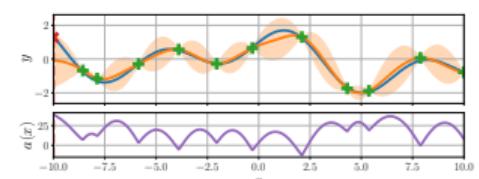
(f) Iteration 6



(g) Iteration 7



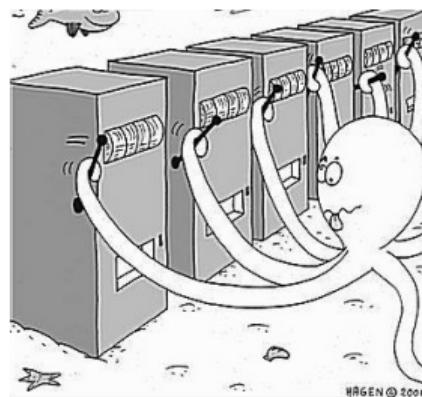
(h) Iteration 8



(i) Iteration 9

Figure 10: Bayesian optimization results with GP-UCB criterion.

Relationship to Multi-Armed Bandit Problem



- ▶ Each machine returns a reward $\hat{r}_a \sim p_{\theta_a}(r_a)$ where $a \in \{1, \dots, K\}$.
- ▶ It minimizes a cumulative regret $T\mu^* - \sum_{t=1}^T \hat{r}_{a_t}$ where $\mu^* = \max_{a \in \{1, \dots, K\}} \mu_a$.
- ▶ Bayesian optimization can be considered as **infinite bandits with dependent arms**.

Relationship to Thompson Sampling

- ▶ Thompson sampling is usually applied in multi-armed bandit problems.
- ▶ For the case of a beta-Bernoulli bandit, Thompson sampling is defined as follows.

Algorithm 2 Thompson Sampling for a Beta-Bernoulli Bandit

```
1: for  $t = 1, 2, \dots, T$  do
2:   for  $k = 1, \dots, K$  do
3:     Sample  $\hat{\theta}_k \sim \text{beta}(\alpha_k, \beta_k)$ .
4:   end for
5:    $x_t \leftarrow \arg \max_k \hat{\theta}_k$ .
6:   Apply  $x_t$  and observe  $r_t$ .
7:    $(\alpha_{x_t}, \beta_{x_t}) \leftarrow (\alpha_{x_t} + r_t, \beta_{x_t} + 1 - r_t)$ .
8: end for
```

- ▶ After sampling the possibilities, it chooses a maximizer of those sampled values.