OPT2025: 17th Annual Workshop on Optimization for Machine Learning

Lipschitz Optimization via Weighted Sampling Based on Expected Potential Maximizers Reduction

Hideyuki Masui MASUI.HIDEYUKI@BC.MITSUBISHIELECTRIC.CO.JP
Koki Nakane NAKANE.KOKI@CW.MITSUBISHIELECTRIC.CO.JP
Renshi Nagasawa NAGASAWA.RENSHI@DP.MITSUBISHIELECTRIC.CO.JP
Mitsubishi Electric Corporation, Japan

Abstract

We address the problem of black-box optimization under the assumption that the objective function is Lipschitz-continuous. In traditional Lipschitz optimization algorithms, potential maximizers are defined based on lower and upper bounds estimated from observed data, and new query points are selected uniformly at random from this set. In contrast, we propose a weighted sampling strategy guided by a probabilistic model, where each candidate point is assigned a weight corresponding to its expected reduction of the potential maximizers. This enables more informative and efficient exploration. To retain theoretical guarantees, we incorporate a small probability of uniform sampling from the potential maximizers, ensuring convergence to the global optimum. We demonstrate the effectiveness of the proposed method through numerical experiments on standard benchmark functions and hyperparameter optimization tasks using UCI datasets.

1. Introduction

Lipschitz optimization (LO) [14, 16] leverages the Lipschitz continuity of the objective function to discard suboptimal regions. Recent method, notably AdaLIPO [11], guarantee convergence under minimal assumptions by alternating between global exploration and local exploitation. However, its reliance on uniform random sampling—particularly during exploitation—can cause inefficiencies, especially in high-dimensional or structured domains. Within LO, HALO [5] improves scalability by estimating local Lipschitz constants. On the other hand, in black-box optimization, Bayesian optimization (BO) [7] has gained wide attention. BO employs probabilistic surrogates, typically Gaussian processes (GP) [15], to balance exploration and exploitation. Lipschitz-BO [1] further integrates Lipschitz constraints directly into acquisition functions.

Our approach takes the opposite perspective: rather than enriching BO with LO, we integrate BO-inspired acquisition strategies into the LO framework. We extend AdaLIPO, which alternates between exploration and exploitation but samples uniformly in the latter. Instead, we introduce a weighted scheme inspired by MILE [19], sampling candidates by their expected reduction of potential maximizers. This improves the efficiency of LO while preserving convergence guarantees.

Contributions. Our main contributions are as follows:

- We introduce the first analytical formulation of the expected potential maximizers reduction (EPMR), enabling tractable computation without Monte Carlo integration.
- We integrate EPMR into Lipschitz optimization (AdaLIPO) via a weighted sampling strategy that preserves its global convergence guarantee.

- We empirically demonstrate, on benchmark functions and hyperparameter optimization tasks, that our proposed algorithm accelerates the contraction of potential maximizers and achieves lower regret than existing Lipschitz optimization baselines.

2. Background and Related Work

2.1. Problem Setup and Lipschitz Optimization

We consider the global optimization of a black-box function $f : \mathcal{X} \to \mathbb{R}$, where $\mathcal{X} \subset \mathbb{R}^d$ is a compact search space. The aim of the optimization is to identify the global maximizer:

$$\boldsymbol{x}^* = \arg\max_{\boldsymbol{x} \in \mathcal{X}} f(\boldsymbol{x}), \tag{1}$$

under a limited budget of evaluations. Lipschitz optimization assumes that the objective function f is Lipschitz-continuous with some (unknown or known) Lipschitz constant $L > 0$, i.e.,

$$|f(\boldsymbol{x}) - f(\boldsymbol{x}')| \leq L\|\boldsymbol{x} - \boldsymbol{x}'\|, \quad \forall \boldsymbol{x}, \boldsymbol{x}' \in \mathcal{X}.$$

This assumption bounds $f(\boldsymbol{x})$ using the observed data $\mathcal{D}_t = \{(\boldsymbol{x}_i, f(\boldsymbol{x}_i))\}_{i=1}^t$, as follows:

$$f^l_{L,t}(\boldsymbol{x}) = \max_{i \in [1,\dots,t]} \left[f(\boldsymbol{x}_i) - L\|\boldsymbol{x} - \boldsymbol{x}_i\| \right], \quad f^u_{L,t}(\boldsymbol{x}) = \min_{i \in [1,\dots,t]} \left[f(\boldsymbol{x}_i) + L\|\boldsymbol{x} - \boldsymbol{x}_i\| \right],$$

where $f^l_{L,t}$ and $f^u_{L,t}$ are the lower and upper bounds. Based on these bounds, the set of potential maximizers [11] is then defined as the points not yet ruled out from being global maximizer.

Definition 1 (Potential Maximizers) *Given a Lipschitz constant $L > 0$ and observed data $\mathcal{D}_t$, the set of potential maximizers is defined as*

$$\mathcal{X}_{L,t} \equiv \left\{ \boldsymbol{x} \in \mathcal{X} \;\middle|\; f^u_{L,t}(\boldsymbol{x}) \geq \max_{i \in [1,\dots,t]} f(\boldsymbol{x}_i) \right\}. \tag{2}$$

This is the subset of the search space where the global maximizer (1) may still exist under Lipschitz continuity. AdaLIPO estimates the Lipschitz constant from the observed data $\mathcal{D}_t$ as follows:

$$\hat{L}_t = \max_{i \neq j,\ i,j \in [1,\dots,t]} \frac{|f(\boldsymbol{x}_i) - f(\boldsymbol{x}_j)|}{\|\boldsymbol{x}_i - \boldsymbol{x}_j\|}. \tag{3}$$

In AdaLIPO, exploration or exploitation is decided by $b_t \sim \mathcal{B}(p)$, a Bernoulli variable with success probability p for exploration. If $b_t = 1$ (exploration), the next query is sampled uniformly from $\mathcal{X}$. Otherwise if $b_t = 0$ (exploitation), it is sampled uniformly from the potential maximizers $\mathcal{X}_{L,t}$. The pseudocode is given in Algorithm 1 in Appendix A.

2.2. Bayesian Optimization and Level-Set Estimation

Bayesian optimization is a framework for optimizing expensive black-box functions by using a probabilistic surrogate model. Gaussian process regression is commonly used to model the objective function. The posterior distribution based on the observed data $\mathcal{D}_t$ is modeled as normal distribution:

$$f(\boldsymbol{x}) \mid \mathcal{D}_t \sim \mathcal{N}(\mu_t(\boldsymbol{x}), \sigma_t^2(\boldsymbol{x})), \tag{4}$$

Then, BO maximizes an acquisition function to determine the next query point:

$$\boldsymbol{x}_{t+1} = \arg\max_{\boldsymbol{x}\in\mathcal{X}} \alpha(\boldsymbol{x}; \mu_t(\boldsymbol{x}), \sigma_t(\boldsymbol{x}))^{1}. \tag{5}$$

While EI [12] and UCB [17] select points using the current posterior mean and variance, other methods assess a candidate's informativeness by its anticipated effect on future predictions. For example, MILE [19], developed for level-set estimation, selects the point expected to most reduce the volume of $\mathcal{I}_t = \{\boldsymbol{x} \in \mathcal{X} \mid P(f(\boldsymbol{x}) > \tau) > 1 - \delta\}$, where τ is the target threshold and δ is the allowed misclassification probability. The acquisition function is defied as bellow:

$$\alpha_{\text{MILE}}(\boldsymbol{x}) = \mathbb{E}_y \left[|\mathcal{I}_{t|\boldsymbol{x}}| \right] - |\mathcal{I}_t|, \tag{6}$$

where $\mathcal{I}_{t|\boldsymbol{x}}$ denotes the updated estimate after observing $(\boldsymbol{x}, y = f(\boldsymbol{x}))$. Although originally designed for level-set estimation, the idea of quantifying the global impact of a query point motivates our algorithm. Details of GP (4), acquisition function (5) and MILE (6) are provided in Appendix B.

3. Proposed Algorithm

3.1. Expected Potential Maximizers Reduction

In AdaLIPO, the exploitation phase samples uniformly from the potential maximizers $\mathcal{X}_{L,t}$ without regard to their informativeness. While this guarantees eventual convergence, it can slow down the contraction of $\mathcal{X}_{L,t}$, particularly in high-dimensional or structured domains.

Inspired by the lookahead principle in MILE (6), we propose a utility function that quantifies the informativeness of a candidate $\boldsymbol{x} \in \mathcal{X}_{L,t}$ by measuring the expected reduction in the number of potential maximizers after observing $(\boldsymbol{x}, y)$. Since computing the exact volume of $\mathcal{X}_{L,t}$ is intractable, we approximate it using a finite subset $\mathcal{S}_{L,t} \subset \mathcal{X}_{L,t}$ sampled uniformly at each iteration. Then, we define the expected potential maximizer reduction (EPMR) as bellow:

$$\alpha_{\text{EPMR}}(\boldsymbol{x}) = |\mathcal{S}_{L,t}| - \mathbb{E}_y \left[\left| \mathcal{S}_{L,t|\boldsymbol{x}} \right| \right], \tag{7}$$

where $\mathcal{S}_{L,t|\boldsymbol{x}}$ denotes the subset of potential maximizers after observing $(\boldsymbol{x}, y)$. Assuming a Gaussian process surrogate model with predictive mean $\mu_t(\boldsymbol{x})$ and variance $\sigma_t^2(\boldsymbol{x})$, the EPMR can be computed analytically as follows:

Theorem 2 *Let $y_t^* = \max_{i=1,\dots,t} y_i$. Then,*

$$\begin{aligned}\alpha_{EPMR}(\boldsymbol{x}) = \sum_{\boldsymbol{x}'\in\mathcal{S}_{L,t}} \Bigg[&\max\left\{ \Phi\left(\frac{y_t^* - L\|\boldsymbol{x}' - \boldsymbol{x}\| - \mu_t(\boldsymbol{x})}{\sigma_t(\boldsymbol{x})} \right) - \Phi\left(\frac{f_{L,t}^l(\boldsymbol{x}) - \mu_t(\boldsymbol{x})}{\sigma_t(\boldsymbol{x})} \right), 0 \right\} \\ &+ \max\left\{ \Phi\left(\frac{f_{L,t}^u(\boldsymbol{x}) - \mu_t(\boldsymbol{x})}{\sigma_t(\boldsymbol{x})} \right) - \Phi\left(\frac{f_{L,t}^u(\boldsymbol{x}') - \mu_t(\boldsymbol{x})}{\sigma_t(\boldsymbol{x})} \right), 0 \right\} \Bigg],\end{aligned} \tag{8}$$

where $\Phi(\cdot)$ is the standard normal cumulative distribution function.

Eq. 8 sums over two mutually exclusive exclusion events for each $\boldsymbol{x}' \in S_{L,t}$: (i) its updated upper bound becomes smaller than both its previous bound and the new best value, or (ii) the new best value exceeds its upper bound. The detailed proof is given in Appendix C.

1. For simplicity, we omit the dependence of α on μ_t and σ_t in the following.

3.2. Weighted Sampling and Mixing Exploitation Strategy

We leverage the informativeness measure defined in Section 3.1 by introducing a weighted sampling strategy for selecting the next query point. Specifically, we assign sampling weights to a finite set of potential maximizers $\mathcal{S}_{L,t} \subset \mathcal{X}_{L,t}$ according to their expected potential maximizers reduction (8), and sample the next query point from the following mixture distribution:

$$Q_t(\boldsymbol{x}) = \gamma \cdot \frac{1}{|\mathcal{S}_{L,t}|} + (1-\gamma) \cdot \frac{\alpha_{\mathrm{EPMR}}(\boldsymbol{x})}{\sum_{\boldsymbol{x}' \in \mathcal{S}_{L,t}} \alpha_{\mathrm{EPMR}}(\boldsymbol{x}')} \quad \text{for } \boldsymbol{x} \in \mathcal{S}_{L,t}, \tag{9}$$

where $\gamma \in [0,1]$ controls the trade-off between uniform exploitation (as performed in AdaLIPO) and informativeness-driven exploitation based on EPMR. This approach allows the algorithm to prioritize points that are expected to significantly reduce the volume of the potential maximizers, while retaining sufficient exploration to encourage global optimization. The overall algorithm follows the same high-level structure as AdaLIPO, as summarized in Algorithm D.

4. Experiments

4.1. Experiment Settings

We evaluate the proposed method on synthetic benchmarks [18] and UCI regression tasks [10] to assess regret reduction efficiency against: (i) model-free optimizers (random search [3], CMA-ES [8]), (ii) model-based Bayesian optimization (TPE [4], GP-EI [12]), and (iii) Lipschitz optimization baseline (DIRECT [9], AdaLIPO [11]).[2] All methods are initialized with $N_{\mathrm{init}} = 10$ random points and run for a total budget of $N_{\mathrm{init}} + N_{\mathrm{iter}} = 100$ queries unless stated otherwise. The Lipschitz constant L is estimated via Eq. 3. For our algorithm, EPMR is approximated using $N_{\mathrm{sample}} = 1000$ candidates for $\mathcal{S}_{L,t}$ drawn from $\mathcal{X}_{L,t}$ at each iteration. we set the exploration rate $q = 0.1$ for AdaLIPO and set the mixture ratio of AdaLIPO $\gamma = 0.05$ for our algorithm. Each experiment is repeated with 50 random seeds, and we report the mean of simple regret:

$$r_t = f(\boldsymbol{x}^*) - \max_{i=1,\ldots,t} f(\boldsymbol{x}_i). \tag{10}$$

For statistical comparison, we applied Wilcoxon signed-rank tests [6] with a significance level of $p < 0.05$ across benchmark functions and query budgets. In the tables, the best-performing method is shown in bold. When our proposed method (AdaLIPO /w EPMR) significantly outperforms other Lipschitz-based baselines (DIRECT and AdaLIPO), its regret values are underlined.

4.2. Benchmark Functions

We use functions from [18] and report two representative cases in Table 1: Ackley ($d = 5$) and Hartmann ($d = 6$). These differ in conditioning and landscape complexity, illustrating distinct optimization challenges. Results for other functions appear in Table 3 in Appendix E.1. Our algorithm achieves more efficient exploration than AdaLIPO and DIRECT. The results highlight the effectiveness of the proposed weighted sampling strategy.

2. CMA-ES, TPE, and GP-EI were implemented using the Optuna framework [2]

Table 1: Simple Regret for Benchmark Function

Function	Ackley ($d = 5$)				Hartmann ($d = 6$)			
#Iteration	25	50	75	100	25	50	75	100
Random Search [3]	18.63	17.98	17.51	17.14	0.604	0.409	0.284	0.223
CMA-ES [8]	**13.75**	**11.87**	**11.33**	11.00	0.748	0.512	0.390	0.332
TPE [4]	16.56	14.00	11.96	10.27	0.374	0.142	0.079	0.047
GP-EI [12]	18.38	13.48	13.23	12.88	0.749	**0.001**	**0.000**	**0.000**
DIRECT [9]	18.55	17.20	15.90	15.22	0.796	0.7000	0.695	0.695
AdaLIPO [11]	18.46	16.94	16.29	15.18	0.365	0.128	0.079	0.057
AdaLIPO w/ EPMR	18.66	17.47	13.15	10.44	**0.252**	0.047	0.018	0.011

4.3. UCI Dataset

We further evaluate our algorithm on UCI regression tasks [10], following [11], by optimizing the regularization coefficient λ_1 and kernel bandwidth λ_2 of Gaussian kernel ridge regression [13] over $(\log \lambda_1, \log \lambda_2) \in [-3, 5] \times [-2, 2]$ on a 20×20 grid[3]. Results for concrete compressive strength and wine quality are shown in Table 2, with full results in Table 4 (Appendix E.2). Our method explored more efficiently than DIRECT and AdaLIPO, and, while not statistically significant, reached performance close to TPE and GP-EI.

Table 2: Simple Regret for UCI Regression Tasks

Dataset	Concrete Compressive Strength $[10^{-2}]$				Wine Quality $[10^{-2}]$			
#Iteration	25	50	75	100	25	50	75	100
Random Search [3]	1.908	0.659	0.411	0.311	4.391	2.462	1.833	1.564
CMA-ES [8]	2.510	1.206	0.842	0.689	2.493	1.275	0.722	0.538
TPE [4]	0.283	0.063	0.009	0.005	**1.922**	0.748	**0.196**	**0.019**
GP-EI [12]	1.740	0.039	0.036	0.036	3.871	0.602	0.476	0.476
DIRECT [9]	7.172	7.172	7.172	7.172	7.066	7.066	7.066	7.066
AdaLIPO [11]	0.445	0.096	0.026	0.015	2.944	1.370	0.928	0.588
AdaLIPO w/ EPMR	0.268	0.043	0.009	0.002	2.655	1.146	0.538	0.217

5. Conclusion

We proposed a novel acquisition strategy for Lipschitz optimization that prioritizes candidates by their expected reduction of potential maximizers. Inspired by MILE, we defined an informativeness measure (EPMR) and integrated it into AdaLIPO via weighted sampling and mixing exploitation strategy. The algorithm preserves the convergence guarantee as AdaLIPO. while accelerating convergence in practice. Experiments on benchmark functions and hyperparameter optimization tasks using UCI datasets confirmed its effectiveness over baselines. Future directions include extensions to constrained and multi-objective settings, and adaptive exploration-exploitation balancing.

3. CMA-ES cannot directly handle the discrete search space; suggestions are mapped to the nearest grid point.

References

[1] M. O. Ahmed, S. Vaswani, and M. Schmidt. Combining bayesian optimization and lipschitz optimization. *Machine Learning*, 109, January 2020. ISSN 1573-0565.

[2] T. Akiba, S. Sano, T. Yanase, T. Ohta, and M. Koyama. Optuna: A next-generation hyper-parameter optimization framework. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2019.

[3] J. Bergstra and Y. Bengio. Random search for hyper-parameter optimization. *Journal of Machine Learning Research*, 13:281–305, 2012.

[4] J. Bergstra, R. Bardenet, Y. Bengio, and B. Kégl. Algorithms for hyper-parameter optimization. In *Proceedings of the 25th International Conference on Neural Information Processing Systems*, NIPS'11, pages 2546–2554, 2011.

[5] D. D'Agostino. An efficient global optimization algorithm with adaptive estimates of the local lipschitz constants, 2024. URL https://arxiv.org/abs/2211.04129.

[6] J. Demšar. Statistical comparisons of classifiers over multiple data sets. *Journal of Machine Learning Research*, 7(1):1–30, 2006. URL http://jmlr.org/papers/v7/demsar06a.html.

[7] P. I. Frazier. A tutorial on bayesian optimization, 2018. URL https://arxiv.org/abs/1807.02811.

[8] N. Hansen. *The CMA Evolution Strategy: A Comparing Review*, pages 75–102. Springer Berlin Heidelberg, 2006.

[9] D. R. Jones, C. D. Perttunen, and B. E. Stuckman. Lipschitzian optimization without the lipschitz constant. *Journal of Optimization Theory and Applications*, 79, October 1993. ISSN 1573-2878.

[10] M. Kelly, R. Longjohn, and K. Nottingham. The uci machine learning repository. URL https://archive.ics.uci.edu.

[11] C. Malherbe and N. Vayatis. Global optimization of lipschitz functions. ICML'17, pages 2314–2323. JMLR.org, 2017.

[12] J. Mockus, V. Tiesis, and A. Zilinskas. *The application of Bayesian methods for seeking the extremum*. Towards Global Optimisation 2, 1978.

[13] K. P. Murphy. *Machine Learning: A Probabilistic Perspective*. The MIT Press, 2012.

[14] S. A. Piyavskii. An algorithm for finding the absolute extremum of a function. *USSR Computational Mathematics and Mathematical Physics*, 12(4):57–67, 1972.

[15] C. E. Rasmussen and C. K. I. Williams. *Gaussian Processes for Machine Learning*. The MIT Press, 2005.

[16] B. O. Shubert. A sequential method seeking the global maximum of a function. *SIAM Journal on Numerical Analysis*, 9(3):379–388, 1972.

[17] 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 27th International Conference on International Conference on Machine Learning*, ICML'10, pages 1015–1022, 2010.

[18] S. Surjanovic and D. Bingham. Virtual library of simulation experiments: Test functions and datasets. URL http://www.sfu.ca/~ssurjano.

[19] A. Zanette, J. Zhang, and M. J. Kochenderfer. Robust super-level set estimation using gaussian processes, 2018. URL https://arxiv.org/abs/1811.09977.

Appendix A. Pseudocode of AdaLIPO

Algorithm 1: AdaLIPO [11]

Input : $N_{init}, N_{iter}, q, \mathcal{X}, f$

Output: $\boldsymbol{x}^* = \underset{i=1,\ldots,N_{init}}{\arg\max}\ f(\boldsymbol{x}_i)$

Initialization: Query f at N_{init} points $\boldsymbol{x}_i \sim \mathcal{U}_{\mathcal{X}}(\boldsymbol{x})$, and set $\mathcal{D}_{N_{\text{init}}} = \{(\boldsymbol{x}_i, f(\boldsymbol{x}_i))\}_{i=1}^{N_{\text{init}}}$, $t \leftarrow N_{\text{init}}$

for $t = N_{init}$ **to** N_{iter} **do**

- Sample $b_t \sim \mathcal{B}(q)$
- **if** $b_t = 1$ *(Exploration)* **then**
  - Sample $\boldsymbol{x}_{t+1} \sim \mathcal{U}_{\mathcal{X}}(\boldsymbol{x})$
- **end**
- **if** $b_t = 0$ *(Exploitation)* **then**
  - Estimate Lipschitz constant $\hat{L}_t$ (Eq. 3)
  - Identify the set of potential maximizers $\mathcal{X}_{\hat{L}_t,t}$ (Eq. 2)
  - Sample $\boldsymbol{x}_{t+1} \sim \mathcal{U}_{\mathcal{X}_{\hat{L}_t,t}}(\boldsymbol{x})$
- **end**
- Evaluate $y_{t+1} = f(\boldsymbol{x}_{t+1})$
- Update $\mathcal{D}_t \leftarrow \mathcal{D}_t \cup (\boldsymbol{x}_{t+1}, y_{t+1})$, $t \leftarrow t+1$

end

Here, $\mathcal{U}_{\mathcal{X}}(\boldsymbol{x})$ denotes the uniform distribution over the search space $\mathcal{X}$ and $\mathcal{B}(q)$ denotes the Bernoulli distribution with success probability q.

Appendix B. Bayesian Optimization and Level-Set Estimation

Bayesian optimization (BO) is a framework for optimizing black-box functions by leveraging a probabilistic surrogate model, typically a Gaussian process (GP).

B.1. Gaussian Process Regression

Let $f : \mathcal{X} \to \mathbb{R}$ be the objective function, and let $\mathcal{D}_t = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^t$ be the observed data, where $y_i = f(\boldsymbol{x}_i) + \varepsilon_i$ and $\varepsilon_i \sim \mathcal{N}(0, \sigma_\epsilon^2)$ is i.i.d. Gaussian noise. We model f using a GP prior with mean function $\mu_0(\boldsymbol{x})$ and kernel function $k(\boldsymbol{x}, \boldsymbol{x}')$:

$$f(\boldsymbol{x}) \sim \mathcal{GP}(\mu_0(\boldsymbol{x}), k(\boldsymbol{x}, \boldsymbol{x}')).$$

Commonly, mean of GP prior is set to the constant function to zero, $\mu_0(\boldsymbol{x}) = 0$. Given a query point $\boldsymbol{x} \in \mathcal{X}$, the predictive distribution under the GP posterior is Gaussian:

$$f(\boldsymbol{x}) \mid \mathcal{D}_t \sim \mathcal{N}(\mu_t(\boldsymbol{x}), \sigma_t^2(\boldsymbol{x})),$$

where

$$\mu_t(\boldsymbol{x}) = \boldsymbol{k}_t(\boldsymbol{x})^\top \left(\boldsymbol{K}_t + \sigma_\epsilon^2 \boldsymbol{I}\right)^{-1} \boldsymbol{y}_t,$$
$$\sigma_t^2(\boldsymbol{x}) = k(\boldsymbol{x}, \boldsymbol{x}) - \boldsymbol{k}_t(\boldsymbol{x})^\top \left(\boldsymbol{K}_t + \sigma_\epsilon^2 \boldsymbol{I}\right)^{-1} \boldsymbol{k}_t(\boldsymbol{x}).$$

Here, - $\boldsymbol{K}_t \in \mathbb{R}^{t \times t}$ is the kernel matrix with $(\boldsymbol{K}_t)_{ij} = k(\boldsymbol{x}_i, \boldsymbol{x}_j)$, - $\boldsymbol{k}_t(\boldsymbol{x}) = [k(\boldsymbol{x}_1, \boldsymbol{x}), \ldots, k(\boldsymbol{x}_t, \boldsymbol{x})]^\top$, - $\boldsymbol{y}_t = [y_1, \ldots, y_t]^\top$.

B.2. Acquisition Function

In Bayesian optimization, various acquisition functions are used to balance exploration and exploitation. Below are typical examples with their analytical forms, assuming a Gaussian process posterior with predictive mean $\mu_t(\boldsymbol{x})$ and standard deviation $\sigma_t(\boldsymbol{x})$, and the best observed value $y_t^* = \max_{i=1,\dots,t} y_i$.

Expected Improvement (EI)

The Expected Improvement (EI) function quantifies the expected amount of improvement over the current best value. It encourages both exploring uncertain regions and exploiting promising areas.

$$\begin{aligned}\alpha_{\mathrm{EI}}(\boldsymbol{x}) &= \mathbb{E}[\max(0, f(\boldsymbol{x}) - y_t^*)], \\ &= (\mu_t(\boldsymbol{x}) - y_t^*)\Phi(z) + \sigma_t(\boldsymbol{x})\phi(z), \\ z &= \frac{\mu_t(\boldsymbol{x}) - y_t^*}{\sigma_t(\boldsymbol{x})}.\end{aligned}$$

Here, $\Phi(\cdot)$ is the cumulative distribution function (CDF) of the standard normal distribution, representing the probability of improvement. $\phi(\cdot)$ is the probability density function (PDF), representing the spread of the distribution. The first term encourages exploitation by focusing on areas with high mean predictions, while the second term encourages exploration by considering uncertainty.

Probability of Improvement (PI)

The Probability of Improvement (PI) simply computes the probability that a query point improves upon the current best. It is straightforward and purely probabilistic but may ignore the magnitude of improvement:

$$\alpha_{\mathrm{PI}}(\boldsymbol{x}) = \Phi\left(\frac{\mu_t(\boldsymbol{x}) - y_t^*}{\sigma_t(\boldsymbol{x})}\right).$$

Upper Confidence Bound (UCB)

The Upper Confidence Bound (UCB) acquisition function balances exploration and exploitation by combining the predictive mean and the standard deviation. The parameter β_t adjusts the degree of exploration:

$$\alpha_{\mathrm{UCB}}(\boldsymbol{x}) = \mu_t(\boldsymbol{x}) + \beta_t \sigma_t(\boldsymbol{x}).$$

Larger values of β_t lead to more exploratory behavior, while smaller values encourage exploitation.

These acquisition functions rely solely on the GP posterior at the current step. In contrast, our proposed informativeness measure (EPMR) considers the expected reduction of the potential maximizer region, providing a different perspective on exploration.

B.3. Level-Set Estimation and MILE

The Maximum Improvement for Level-set Estimation (MILE) selects the next query point by estimating the expected gap of the volume of level-set $\mathcal{I}_t = \{\boldsymbol{x} \in \mathcal{X} \mid P(f(\boldsymbol{x}) > \tau) > 1 - \delta\}$ after observing $(\boldsymbol{x}, y)$. Its acquisition function is defined as:

$$\alpha_{\text{MILE}}(\boldsymbol{x}) = \mathbb{E}_y \left[|\mathcal{I}_{t|\boldsymbol{x}}| \right] - |\mathcal{I}_t|,$$

where $\mathcal{I}_{t|\boldsymbol{x}}$ denotes the updated estimate after observing $(\boldsymbol{x}, f(\boldsymbol{x}))$. The second term is a constant that does not depend on $\boldsymbol{x}$. The first term can be analytically calculated as below:

$$\mathbb{E}_y \left[|\mathcal{I}_{t|\boldsymbol{x}}| \right] = \sum_{\boldsymbol{x}' \in \mathcal{X}} \Phi \left(\frac{\sqrt{\sigma_t^2(\boldsymbol{x}) + \sigma_\epsilon^2}}{|\text{Cov}_t(f(\boldsymbol{x}'), f(\boldsymbol{x}))|} \times \left(\mu_t(\boldsymbol{x}') - \beta \sigma_{t|\boldsymbol{x}}(\boldsymbol{x}') - \tau \right) \right), \tag{11}$$

where $\sigma^2_{t|\boldsymbol{x}}(\boldsymbol{x}')$ is the variance of posterior distribution of GP after observing $\boldsymbol{x}$, which is analytically calculated as below:

$$\sigma^2_{t|\boldsymbol{x}}(\boldsymbol{x}') = \sigma_t^2(\boldsymbol{x}') - \frac{\text{Cov}_t^2(f(\boldsymbol{x}'), f(\boldsymbol{x}))}{\sigma_t^2(\boldsymbol{x}') + \sigma_\epsilon^2}.$$

Here, $\text{Cov}_t(f(\boldsymbol{x}'), f(\boldsymbol{x}))$ denotes the posterior covariance between $\boldsymbol{x}'$ and $\boldsymbol{x}$ under the GP model. It can be computed using the kernel matrix $\boldsymbol{K}_t$ as:

$$\text{Cov}_t = \boldsymbol{K}_t(\boldsymbol{X}, \boldsymbol{X}) - \boldsymbol{K}_t(\boldsymbol{X}, \boldsymbol{X}_t) \left(\boldsymbol{K}_t(\boldsymbol{X}_t, \boldsymbol{X}_t) + \sigma_\epsilon \boldsymbol{I} \right)^{-1} \boldsymbol{K}_t(\boldsymbol{X}_t, \boldsymbol{X}').$$

where $\boldsymbol{K}_t(\boldsymbol{X}, \boldsymbol{X}) \in \mathbb{R}^{|\mathcal{X}| \times |\mathcal{X}|}$ and $\boldsymbol{K}_t(\boldsymbol{X}, \boldsymbol{X}_t) \in \mathbb{R}^{|\mathcal{X}| \times t}$ denotes the kernel matrix. The detail derivation of Eq. 11 can be found in [19].

In practice, in the case of $\mathcal{X}$ is a continuous space, the expectation is approximated using Monte Carlo sampling. MILE is particularly useful when the goal is not to find the maximum itself but to characterize regions exceeding a given threshold, which also aligns with identifying potential maximizers under the Lipschitz assumption.

Appendix C. Expected Potential Maximizers Reduction

This appendix provides the proof of Theorem 2 presented in the main text. The theorem derives an analytical expression for the expected reduction in the number of potential maximizers after querying a new point $\boldsymbol{x}$.

Proof

Let $\mathcal{S}_{L,t} \subset \mathcal{X}_{L,t}$ be a subset potential maximizers under the Lipschitz assumption. We consider the condition under which $\boldsymbol{x}'$ is excluded from $\mathcal{S}_{L,t}$ after observing $f(\boldsymbol{x}) = y$. There are two cases:

- **Case 1:** The updated upper bound becomes lower than both the previous upper bound and the updated current best value:

$$f^u_{L,t|\boldsymbol{x}}(\boldsymbol{x}') < f^u_{L,t}(\boldsymbol{x}'), \quad f^u_{L,t|\boldsymbol{x}}(\boldsymbol{x}') < \max\{y_t^*, y\}.$$

- **Case 2:** The updated upper bound remains unchanged ($f^u_{L,t|\boldsymbol{x}}(\boldsymbol{x}') = f^u_{L,t}(\boldsymbol{x}')$), but $y > y_t^*$ and $f^u_{L,t}(\boldsymbol{x}') < y$, which violates the condition for being a potential maximizer.

We first examine Case 1. The updated upper bound is:

$$f^u_{L,t}(\boldsymbol{x}') = \min_{i=1,\dots,t} \left[y_i + L\|\boldsymbol{x}' - \boldsymbol{x}_i\| \right],$$
$$f^u_{L,t|\boldsymbol{x}}(\boldsymbol{x}') = \min \left\{ f^u_{L,t}(\boldsymbol{x}'),\ y + L\|\boldsymbol{x}' - \boldsymbol{x}\| \right\}.$$

To decrease the upper bound, the following must hold:

$$y + L\|\boldsymbol{x}' - \boldsymbol{x}\| < f^u_{L,t}(\boldsymbol{x}) \quad \Rightarrow \quad y < f^u_{L,t}(\boldsymbol{x}) - L\|\boldsymbol{x}' - \boldsymbol{x}\|.$$

Additionally, to exclude $\boldsymbol{x}$ from $\mathcal{S}_{L,t|\boldsymbol{x}}$, it must also be:

$$y + L\|\boldsymbol{x}' - \boldsymbol{x}\| < y^*_t \quad \Rightarrow y < y^*_t - L\|\boldsymbol{x}' - \boldsymbol{x}\|. \tag{12}$$

Since $f^u_{L,t}(\boldsymbol{x}') \geq y^*_t$ for all $\boldsymbol{x}' \in \mathcal{S}_{L,t}$, the latter condition suffices for exclusion. Now consider Case 2, the following must hold:

$$y > f^u_{L,t}(\boldsymbol{x}'). \tag{13}$$

Combining both cases Eq. 12 and Eq. 13, the expected reduction in the number of potential maximizers is:

$$\begin{aligned}
\alpha_{\text{EPMR}}(\boldsymbol{x}) &= \int_{f^l_{L,t}(\boldsymbol{x})}^{f^u_{L,t}(\boldsymbol{x})} p_t(y \mid \boldsymbol{x}) \cdot \left(|\mathcal{S}_{L,t}| - |\mathcal{S}_{L,t|\boldsymbol{x}}| \right) dy \\
&= \sum_{\boldsymbol{x}' \in \mathcal{S}_{L,t}} \Big[P_t(f^l_{L,t}(\boldsymbol{x}) < y < y^*_t - L\|\boldsymbol{x}' - \boldsymbol{x}\|) \\
&\quad + P_t(f^u_{L,t}(\boldsymbol{x}') < y < f^u_{L,t}(\boldsymbol{x})) \Big].
\end{aligned}$$

Using the Gaussian posterior predictive distribution at $\boldsymbol{x}$, we obtain:

$$\begin{aligned}
\alpha_{\text{EPMR}}(\boldsymbol{x}) = \sum_{\boldsymbol{x}' \in \mathcal{S}_{L,t}} \Bigg[&\max \left\{ \Phi\left(\frac{y^*_t - L\|\boldsymbol{x}' - \boldsymbol{x}\| - \mu_t(\boldsymbol{x})}{\sigma_t(\boldsymbol{x})} \right) - \Phi\left(\frac{f^l_{L,t}(\boldsymbol{x}) - \mu_t(\boldsymbol{x})}{\sigma_t(\boldsymbol{x})} \right),\ 0 \right\} \\
&+ \max \left\{ \Phi\left(\frac{f^u_{L,t}(\boldsymbol{x}) - \mu_t(\boldsymbol{x})}{\sigma_t(\boldsymbol{x})} \right) - \Phi\left(\frac{f^u_{L,t}(\boldsymbol{x}') - \mu_t(\boldsymbol{x})}{\sigma_t(\boldsymbol{x})} \right),\ 0 \right\} \Bigg],
\end{aligned} \tag{14}$$

which completes the proof. ■

Appendix D. Pseudocode of AdaLIPO with Expected Potential Maximizers Reduction

Algorithm 2: AdaLIPO w/ EPMR (Ours)

Input : $N_{init}, N_{iter}, N_{sample}, q, \gamma, \mathcal{X}, f$
Output: $\boldsymbol{x}^* = \underset{i=1,\ldots,N_{init}}{\arg\max}\ f(\boldsymbol{x}_i)$
Initialization: $\mathcal{D}_{N_{init}} = \{(\boldsymbol{x}_i, f(\boldsymbol{x}_i))\}_{i=1}^{N_{init}}$ where $\boldsymbol{x}_i \sim \mathcal{U}_{\mathcal{X}}(\boldsymbol{x})$, $t \leftarrow N_{init}$
for $t = N_{init}$ **to** N_{iter} **do**
  Sample $b_t \sim \mathcal{B}(q)$
  if $b_t = 1$ *(Exploration)* **then**
    Sample $\boldsymbol{x}_{t+1} \sim \mathcal{U}_{\mathcal{X}}(\boldsymbol{x})$
  end
  if $b_t = 0$ *(Exploitation)* **then**
    Estimate Lipschitz constant $\hat{L}_t$ (Eq. 3)
    Identify the set of potential maximizers $\mathcal{X}_{\hat{L}_t,t}$ (Eq. 2)
    Sample subset of potential maximizes $\mathcal{S}_{\hat{L}_t,t} = \{\boldsymbol{x} \in \mathcal{X}_{\hat{L}_t,t} | \boldsymbol{x} \sim \mathcal{U}_{\mathcal{X}_{\hat{L}_t,t}}(\boldsymbol{x})\}$, where $|\mathcal{S}_{\hat{L}_t,t}| = N_{sample}$
    Sample $\boldsymbol{x}_{t+1} \sim Q_t(\boldsymbol{x})$ (Eq. 9)
  end
  Evaluate $y_{t+1} = f(\boldsymbol{x}_{t+1})$
  Update $\mathcal{D}_t \leftarrow \mathcal{D}_t \cup (\boldsymbol{x}_{t+1}, y_{t+1})$, $t \leftarrow t+1$
end

Appendix E. Experiments

For statistical comparison, we applied Wilcoxon signed-rank tests [6] with a significance level of $p < 0.05$ across benchmark functions and query budgets. In the tables, the best-performing method is shown in bold. When our proposed method (AdaLIPO /w EPMR) significantly outperforms other Lipschitz-based baselines (DIRECT and AdaLIPO), its regret values are underlined.

E.1. Benchmark Functions

We conducted experiments on a range of benchmark functions provided in the benchmark repository by Surjanovic and Bingham [18]. In addition to the $5d$-Ackley and $6d$-Hartmann functions reported in the main paper, we evaluated the performance on the following 20 functions. All functions were evaluated in 2 to 5 dimensions depending on their standard definitions. Table 3 summarizes the mean regret values at each iteration.

E.2. UCI Dataset

We performed kernel ridge regression on 6 datasets from the UCI repository [10], following the protocol of Malherbe and Vayatis [11]. The hyperparameters $(\log \lambda_1, \log \lambda_2)$ were optimized in the range $[-3, 5] \times [-2, 2]$. In the main paper, we reported results for Concrete and Wine datasets. Here, we provide results for the remaining 4 datasets: Airfoil, Cancer, Energy and MPG. Table 4 summarizes the mean regret values at each iteration.

Table 3: Simple Regret for Benchmark Function

Function (d)	Ackley (2)				Beale (2)			
#Iteration	25	50	75	100	25	50	75	100
Random Search	12.14	9.824	8.519	7.801	2.716	1.438	0.909	0.700
CMA-ES	8.353	6.167	5.476	5.002	1.725	0.899	0.588	0.498
TPE	9.779	5.903	4.115	3.124	1.550	**0.404**	**0.198**	**0.148**
GP-EI	13.10	3.826	3.737	3.477	2.830	0.854	0.381	0.255
DIRECT	11.12	9.577	9.550	9.535	3.350	3.174	3.173	3.173
AdaLIPO	8.270	5.435	3.993	3.326	2.061	1.218	0.804	0.654
AdaLIPO w/ EPMR	7.737	**2.759**	**0.949**	**0.727**	1.960	1.167	0.664	0.480
Function (d)	Branin (2)				Bukin (2)			
#Iteration	25	50	75	100	25	50	75	100
Random Search	1.764	0.923	0.684	0.579	31.36	19.25	16.87	14.41
CMA-ES	3.231	2.227	1.540	1.401	27.68	17.88	15.41	12.48
TPE	1.136	0.298	0.124	0.068	19.81	12.17	8.875	7.321
GP-EI	2.130	**0.003**	**0.001**	**0.001**	28.31	13.17	12.49	12.40
DIRECT	2.072	1.596	1.594	1.594	30.20	22.32	21.23	20.96
AdaLIPO	1.020	0.557	0.351	0.263	19.47	11.69	9.152	7.353
AdaLIPO w/ EPMR	0.814	0.316	0.148	0.094	**16.39**	**9.211**	**6.429**	**4.321**
Function (d)	Dixon-Price (2)				Drop-Wave (2)			
#Iteration	25	50	75	100	25	50	75	100
Random Search	14.632	6.565	4.342	3.340	0.373	0.289	0.250	0.212
CMA-ES	**3.841**	1.148	0.664	0.467	**0.229**	0.165	0.136	0.110
TPE	7.285	1.213	0.346	0.144	0.316	0.162	**0.098**	**0.081**
GP-EI	17.00	0.649	0.212	0.136	0.419	0.280	0.262	0.261
DIRECT	23.78	21.47	21.45	21.45	0.438	0.421	0.421	0.420
AdaLIPO	12.08	5.989	3.595	1.933	0.368	0.237	0.216	0.186
AdaLIPO w/ EPMR	8.790	4.197	2.429	1.873	0.363	0.270	0.235	0.211
Function (d)	Eggholder (2)				Griewank (2)			
#Iteration	25	50	75	100	25	50	75	100
Random Search	369.1	257.1	231.1	197.4	4.718	2.826	2.193	1.817
CMA-ES	459.9	449.8	431.8	413.5	1.958	1.109	0.962	0.813
TPE	366.4	273.3	202.4	168.1	2.863	1.153	0.733	0.549
GP-EI	321.4	**203.2**	**144.4**	**105.6**	5.811	**0.407**	**0.381**	**0.381**
DIRECT	351.1	330.3	327.9	327.4	5.481	4.866	4.848	4.844
AdaLIPO	352.2	261.5	208.4	174.3	2.540	1.117	0.788	0.661
AdaLIPO w/ EPMR	354.8	264.4	223.8	193.3	1.540	0.891	0.793	0.740
Function (d)	Holder Table (2)				Levy (2)			
#Iteration	25	50	75	100	25	50	75	100

Continued on next page

Random Search	7.926	5.826	3.753	3.056	0.944	0.608	0.409	0.323
CMA-ES	10.88	10.82	10.79	10.69	0.381	0.180	0.129	0.094
TPE	5.222	2.805	**1.395**	**0.848**	0.511	0.130	0.035	0.018
GP-EI	8.336	3.223	2.247	1.268	1.019	**0.024**	**0.001**	**0.001**
DIRECT	7.087	6.477	6.474	6.474	1.009	0.876	0.876	0.876
AdaLIPO	5.954	2.989	2.093	1.654	0.801	0.422	0.304	0.266
AdaLIPO w/ EPMR	5.913	3.383	2.606	1.747	0.644	0.202	0.113	0.085
Function (d)	Michalewicz (2)				Rastrigin (2)			
#Iteration	25	50	75	100	25	50	75	100
Random Search	0.697	0.620	0.538	0.484	10.481	7.714	6.409	5.400
CMA-ES	0.518	0.370	0.272	0.226	**6.234**	4.093	3.084	2.476
TPE	0.488	0.363	0.204	0.143	9.054	4.234	2.998	2.747
GP-EI	0.669	**0.138**	**0.029**	**0.000**	11.09	5.878	4.632	3.934
DIRECT	0.496	0.425	0.424	0.424	11.39	8.883	8.869	8.867
AdaLIPO	0.559	0.354	0.213	0.140	8.780	6.189	4.728	3.966
AdaLIPO w/ EPMR	0.573	0.351	0.141	0.024	8.600	6.748	5.828	5.394
Function (d)	Rosenbrock (2)				Six-Hump Camel (2)			
#Iteration	25	50	75	100	25	50	75	100
Random Search	47.22	16.30	12.94	8.730	0.572	0.357	0.258	0.181
CMA-ES	25.12	7.611	4.639	2.837	0.342	0.154	0.096	0.079
TPE	26.46	6.136	2.895	1.995	0.326	0.124	0.045	0.021
GP-EI	56.21	**2.674**	**0.658**	**0.367**	0.676	**0.027**	**0.001**	**0.000**
DIRECT	136.6	128.2	128.1	128.1	0.528	0.460	0.459	0.459
AdaLIPO	62.62	21.49	11.90	9.001	0.516	0.307	0.233	0.191
AdaLIPO w/ EPMR	39.09	17.93	8.729	6.110	0.423	0.201	0.132	0.085
Function (d)	Styblinski-Tang (2)				Three-Hump Camel (2)			
#Iteration	25	50	75	100	25	50	75	100
Random Search	10.49	6.408	5.444	4.924	0.672	0.410	0.285	0.244
CMA-ES	13.57	10.67	9.279	8.673	**0.306**	0.123	0.093	0.076
TPE	9.840	6.056	3.134	1.745	0.373	0.174	0.080	0.029
GP-EI	11.83	**0.011**	**0.005**	**0.003**	0.817	**0.074**	**0.010**	**0.005**
DIRECT	9.568	9.107	9.105	9.104	0.766	0.675	0.674	0.674
AdaLIPO	11.71	6.723	4.302	3.184	0.666	0.370	0.297	0.250
AdaLIPO w/ EPMR	10.78	2.946	1.233	0.784	0.654	0.386	0.261	0.227
Function (d)	Powell (4)				Dixon-Price (5)			
#Iteration	25	50	75	100	25	50	75	100

Continued on next page

Random Search	246.0	112.9	92.56	75.20	8636	4177	3257	2177
CMA-ES	**43.25**	**18.82**	13.41	11.36	**514.7**	174.8	120.7	101.3
TPE	83.97	32.97	15.97	8.539	2536	793.1	203.6	70.48
GP-EI	207.9	28.45	15.87	8.741	6830	**73.75**	**43.05**	**31.83**
DIRECT	297.1	166.5	127.7	117.7	9295	4852	3232	2653
AdaLIPO	201.5	110.5	91.31	75.70	7805	3161	2357	1858
AdaLIPO w/ EPMR	132.7	65.46	41.22	33.79	5826	838.5	479.3	349.3
Function (d)	Rosenbrock (5) $[10^3]$				Styblinski Tang (5)			
#Iteration	25	50	75	100	25	50	75	100
Random Search	15.17	10.33	7.725	5.758	10.49	6.408	5.444	4.924
CMA-ES	4.475	2.276	1.507	1.081	13.57	10.67	9.279	8.673
TPE	6.468	1.857	0.727	0.344	9.840	6.056	3.134	1.745
GP-EI	13.81	**1.041**	**0.496**	0.279	11.83	**0.011**	**0.005**	**0.003**
DIRECT	19.34	8.954	6.297	5.514	11.15	10.49	10.49	10.49
AdaLIPO	14.12	8.064	6.164	4.542	11.71	6.723	4.302	3.184
AdaLIPO w/ EPMR	8.652	2.652	1.790	1.511	10.78	2.946	1.233	0.784

Table 4: Regret for UCI Regression Tasks

Dataset	Airfoil Self-Noise $[10^{-2}]$				Breast Cancer Wisconsin $[10^{-2}]$			
#Iteration	25	50	75	100	25	50	75	100
Random Search	5.004	1.990	1.494	1.190	1.094	0.702	0.460	0.379
CMA-ES	5.544	3.261	2.120	1.859	1.099	0.471	0.295	0.246
TPE	1.177	0.261	0.039	**0.000**	0.341	0.092	0.028	**0.000**
GP-EI	4.482	**0.065**	0.065	0.065	1.094	**0.064**	0.046	0.046
DIRECT	10.85	10.85	10.85	10.85	6.312	6.312	6.312	6.312
AdaLIPO	2.107	0.528	0.245	0.197	0.659	0.258	0.112	0.049
AdaLIPO w/ EPMR	1.336	0.246	0.077	0.039	0.407	0.194	0.090	0.035
Dataset	Energy Efficiency $[10^{-2}]$				Auto MPG $[10^{-2}]$			
#Iteration	25	50	75	100	25	50	75	100
Random Search	1.246	0.361	0.194	0.175	0.852	0.440	0.304	0.207
CMA-ES	1.693	1.040	0.716	0.612	0.737	0.283	0.193	0.149
TPE	**0.395**	0.042	0.010	0.004	0.330	0.119	0.032	0.003
GP-EI	1.251	0.053	0.005	0.005	0.794	**0.020**	0.017	0.013
DIRECT	5.889	5.889	5.889	5.889	4.955	4.955	4.955	4.955
AdaLIPO	0.745	0.337	0.073	0.039	0.409	0.166	0.098	0.056
AdaLIPO w/ EPMR	0.686	0.269	0.183	0.144	0.333	0.108	0.056	0.029