Batch Acquisition Function Evaluations and Decouple Optimizer Updates for Faster Bayesian Optimization

Kaichi Irie [1,2,3], **Shuhei Watanabe** [2,3], **Masaki Onishi** [3]

[1]Graduate School of Informatics, Kyoto University
[2]Preferred Networks Inc.
[3]Advanced Industrial Science and Technology (AIST)
irie.kaichi.24x@st.kyoto-u.ac.jp

Abstract

Bayesian optimization (BO) efficiently finds high-performing parameters by maximizing an acquisition function, which models the promise of parameters. A major computational bottleneck arises in acquisition function optimization, where multi-start optimization (MSO) with quasi-Newton (QN) methods is required due to the non-convexity of the acquisition function. BoTorch, a widely used BO library, currently optimizes the summed acquisition function over multiple points, leading to the speedup of MSO owing to PyTorch batching. Nevertheless, this paper empirically demonstrates the suboptimality of this approach in terms of off-diagonal approximation errors in the inverse Hessian of a QN method, slowing down its convergence. To address this problem, we propose to decouple QN updates using a coroutine while batching the acquisition function calls. Our approach not only yields the theoretically identical convergence to the sequential MSO but also drastically reduces the wall-clock time compared to the previous approaches. Our approach is available in `GPSampler` in Optuna, effectively reducing its computational overhead.

Code — https://github.com/Kaichi-Irie/faster-batched-acqf-opt-experiments/tree/submission

1 Introduction

Bayesian optimization (BO) is a prevalent method to reduce trial-and-error iterations in many applications with an expensive objective such as materials discovery (Xue et al. 2016; Li et al. 2017; Vahid et al. 2018), hyperparameter optimization (HPO) of machine learning models (Loshchilov and Hutter 2016; Chen et al. 2018; Feurer and Hutter 2019), and drug discovery (Schneider et al. 2020). Since the acquisition function optimization plays a central role in an efficient search, multi-start optimization (MSO) by a quasi-Newton (QN) method, which is essential to enhance the solution quality, is employed by BoTorch (Balandat et al. 2020) and Optuna (Akiba et al. 2019).

Although MSO is widely regarded as the gold standard, its overhead can be non-negligible when each optimization is run sequentially (denoted SEQ. OPT.), as it multiplies acquisition-function evaluations that each entail an expensive Gaussian-process regressor call. BoTorch optimizes the sum of acquisition-function values across multiple restarts [1] to mitigate this cost. This formulation enables PyTorch to batch acquisition function evaluations, reducing overall overhead while preserving the theoretically exact per-point gradients.

Despite these advantages, this formulation substantially slows down the convergence of QN methods based on our experiments. We empirically show that the slowdown is explained by the approximation error in cross derivatives or off-diagonal blocks, which we call *off-diagonal artifacts*, between dimensions that belong to independent points. Off-diagonal artifacts distort search directions, leading to slower convergence.

As a remedy, we propose **D**ecoupling QN updates per restart while **B**atching acquisition function **E**valuations (D-BE). By contrast, we refer to the BoTorch practice, which **C**ouples QN updates while using **B**atched **E**valuations, as C-BE. To our knowledge, there is no practical block-structure-aware, bound-constrained, which is ubiquitous in BO, QN algorithm that removes these off-diagonal artifacts. D-BE sidesteps this difficulty by keeping per-point states independently while retaining batched evaluations, by combining a coroutine and batching—no new solver is required. In experiments, D-BE eliminates off-diagonal artifacts while leveraging hardware throughput and preserving solution quality, yielding up to $1.5\times$ and $1.1\times$ wall-clock speedups over SEQ. OPT. and C-BE, respectively.

In summary, our contributions are as follows:

1. Present a pitfall of C-BE that slows the convergence, focusing on off-diagonal artifacts,
2. Propose D-BE achieved by a novel combination of a coroutine and batching, which avoids the pitfall without degrading solution quality, and
3. Demonstrate that D-BE is faster than SEQ. OPT. and C-BE in experiments.

Notice that our approach has been successfully merged into `GPSampler` in Optuna, significantly accelerating its runtime.

[1]See "Multiple Random Restarts" in the BoTorch documentation. URL: https://botorch.org/docs/v0.15.0/optimization/#multiple-random-restarts.

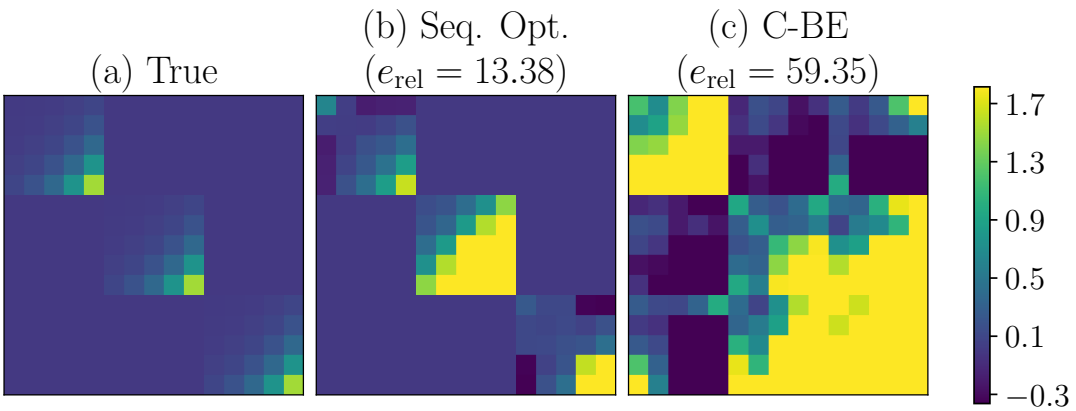

Figure 1: Contour maps of the inverse Hessian (**Left**) and the inverse Hessians approximated by L-BFGS-B with SEQ. OPT. (**Center**) and C-BE (**Right**) on the Rosenbrock function, evaluated near the constrained minimizer ($B = 3, D = 5, \mathbf{x} \in [0, 3]^D$). Each figure has 15×15 tiles, and the (i, j)-th tile corresponds to the colormap for the (i, j)-th element of the (approximated) inverse Hessian. Blue and yellow represent lower and higher values, respectively. Each subtitle reports $e_{\mathrm{rel}}(H) = \|H - H_{\mathrm{true}}\|_F / \|H_{\mathrm{true}}\|_F$. **Left**: The true inverse Hessian exhibits zero at off-diagonal blocks. **Center**: The approximated inverse Hessian by SEQ. OPT.. Off-diagonal blocks show zero everywhere. **Right**: The approximated inverse Hessian by C-BE. Off-diagonal blocks are dense because C-BE does not allow QN methods to be aware of the zero off-diagonal block nature.

2 Background

Bayesian Optimization

Given a function $f(\mathbf{x}) : \mathbb{R}^D \to \mathbb{R}$ to be maximized, Bayesian optimization iteratively suggests a parameter vector $\mathbf{x} \in \mathbb{R}^D$ by maximizing an acquisition function $\alpha : \mathbb{R}^D \to \mathbb{R}$. When the acquisition function is defined on a continuous space, QN methods are widely used for the optimization owing to their fast convergence. Even if the analytical expression of gradients $\nabla\alpha(\mathbf{x})$ is cumbersome to derive, PyTorch enables us to compute the gradients numerically using automatic differentiation (AD) (Ament et al. 2023; Daulton, Balandat, and Bakshy 2020). Although QN methods achieve fast convergence, the non-convexity of acquisition functions necessitates MSO. Namely, the acquisition function is optimized B times independently to enhance the solution quality. The goal of this paper is to propose a method to speed up the B times of independent optimizations without degrading the solution quality.

Quasi-Newton (QN) Methods

QN methods, such as BFGS, L-BFGS, and L-BFGS-B, optimize a function using first-order gradients together with an approximation to the inverse Hessian, i.e., the matrix of second derivatives (Nocedal 1980; Liu and Nocedal 1989; Byrd, Nocedal, and Schnabel 1994; Byrd et al. 1995). In particular, L-BFGS-B is widely used for optimizing acquisition functions because of its fast convergence and ability to handle box constraints.

Multi-Start Optimization: Sequential vs. Batching

Two schemes have been considered for MSO so far, (1) Sequential Optimization (SEQ. OPT.), which performs an independent optimization from each starting point sequentially (see Algorithm 2 in Appendix A for more details), and (2) Coupled Updates with Batched Evaluation (C-BE). More concretely, C-BE optimizes the summation of acquisition function values from independent restarts. Namely, C-BE optimizes the following function with QN methods:

$$\alpha_{\mathrm{sum}}(\mathbf{X}) \coloneqq \sum_{b=1}^{B} \alpha\big(\mathbf{x}^{(b)}\big), \tag{1}$$

where $\mathbf{x}^{(b)} \in \mathbb{R}^D$ is a point in the b-th optimization and $\mathbf{X} = \{\mathbf{x}^{(b)}\}_{b=1}^{B} \in \mathbb{R}^{B \times D}$ is a matrix with each point from the b-th optimization at the b-th column. Since α_{sum} is additively separable, the gradient of α_{sum} with respect to $\mathbf{x}^{(b)}$ matches that of $\alpha(x^{(b)})$, enabling a batched evaluation to get the acquisition function values and their gradients at multiple points simultaneously. In a nutshell, C-BE optimizes a BD-dimensional (additively separable) function instead of solving a D-dimensional function B times independently. BoTorch uses this technique to accelerate MSO.

3 Pitfall of C-BE: Off-Diagonal Artifacts

Although C-BE has been used to accelerate Bayesian optimization, it can in fact slow down each BO iteration or degrade the solution quality when it is used with QN methods.

When using first-order (gradient-based) methods, each block of $\nabla_{\mathbf{X}}\alpha_{\mathrm{sum}}(\mathbf{X})$ equals the per-restart gradient $\nabla\alpha(\mathbf{x}^{(b)})$, making the convergence of C-BE comparable to that of SEQ. OPT. By contrast, QN methods approximate the inverse Hessian, and the accuracy of this approximation strongly affects the convergence rate. Importantly, the Hessian (and its inverse) of the summed acquisition over multiple restarts has zero cross-partial derivatives between variables associated with different restarts. More precisely, the

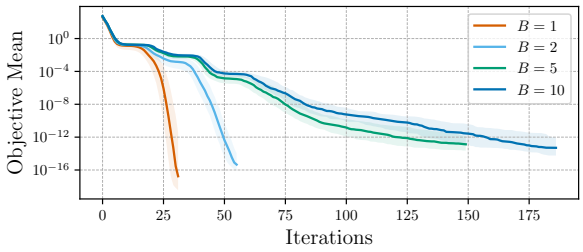

Figure 2: Convergence speed of C-BE when using L-BFGS-B with the memory size $m = 10$ on the Rosenbrock function ($D = 5, \mathbf{x} \in [0, 3]^D$). The figure shows the objective mean over B restarts at each iteration. Each optimization is repeated $1000/B$ times. Each solid line and weak-color band represents the median and the $\pm$ IQR of the objective mean over $1000/B$ runs, respectively. SEQ. OPT. corresponds to $B = 1$. As the number of restarts B increases, the convergence of C-BE requires substantially more iterations.

true Hessian of α_{sum} is block-diagonal:

$$H := \nabla^2_{\mathbf{X}} \alpha_{\mathrm{sum}}(\mathbf{X}) = \begin{bmatrix} \nabla^2 \alpha(\mathbf{x}^{(1)}) & & \mathbf{0} \\ & \ddots & \\ \mathbf{0} & & \nabla^2 \alpha(\mathbf{x}^{(B)}) \end{bmatrix}. \tag{2}$$

However, structure-oblivious QN updates in the BD-dimensional space typically maintain a dense inverse-Hessian approximation, injecting non-zero values into off-diagonal blocks and distorting search directions. We refer to this phenomenon as *off-diagonal artifacts*. Figure 1 visualizes the inverse Hessian and its L-BFGS-B approximation (memory size $m = 10$) under SEQ. OPT. and C-BE. The figure intuitively shows C-BE induces off-diagonal artifacts.

The next question is: "How much do these off-diagonal artifacts affect the convergence speed?". Figure 2 answers this question using the Rosenbrock function. Looking at the medians, the optimization reaches an objective value of 10^{-12} in $\sim$ 30 iterations for SEQ. OPT. ($B = 1$), whereas C-BE with $B = 2$ and $B \in \{5, 10\}$ requires $\sim$ 50 and more than 120 iterations, respectively. Overall, MSO from 10 initial points would require evaluations at about 300 ($= 30 \times 10$) points for $B = 1$ and more than $1200 = (120 \times 10)$ for $B = 10$, concluding that a naïve C-BE weakens the speedup effect. This is also the case for BFGS as shown in Figures 3–5 of Appendix B. It is worth noting that there is no principled way for L-BFGS-B to handle this phenomenon, although BFGS and L-BFGS have been extended to do so by Griewank and Toint and Bigeon, Orban, and Raynaud, respectively. Namely, no work has addressed this issue for the acquisition function optimization, where each parameter has its bounds. The next section introduces a simple yet novel remedy that enables block-diagonal-structure-aware speedups without requiring new QN solvers.

Algorithm 1: MSO: C-BE vs. D-BE

Inputs: α (Acquisition function), B (number of restarts), $\mathbf{X}_0 = \{\mathbf{x}_0^{(b)}\}_{b=1}^B$ (initial points)
Output: $\mathbf{x}^\star \in \mathrm{argmax}_{\mathbf{x}} \alpha(\mathbf{x})$

$\mathbf{X} \leftarrow \mathbf{X}_0$
[C-BE] Initialize a QN optimizer O_{CBE} on $\mathbf{X} \in \mathbb{R}^{B \times D}$
[D-BE] Initialize independent QN optimizers $O_1, \ldots, O_B$
while not converged **do**
  ▷ Batched Evaluation
  $\mathbf{a} \leftarrow \{\alpha(\mathbf{x}^{(b)})\}_{b=1}^B$, $\mathbf{g} \leftarrow \{\nabla\alpha(\mathbf{x}^{(b)})\}_{b=1}^B$
  ▷ QN Updates
  [C-BE]
  $\alpha_{\mathrm{sum}} \leftarrow \sum_{b=1}^B \alpha^{(b)} := \sum_{b=1}^B \alpha(\mathbf{x}^{(b)})$
  $\mathbf{X} \leftarrow O_{\mathrm{CBE}}(\mathbf{X}, \alpha_{\mathrm{sum}}, \mathbf{g})$
  [D-BE]
  for $b = 1, \ldots, B$ **do**
    $\mathbf{x}^{(b)} \leftarrow O_b(\mathbf{x}^{(b)}, \alpha^{(b)}, \mathbf{g}^{(b)})$
return $\mathbf{x}^\star = \mathrm{argmax}_{\mathbf{x} \in \{\mathbf{x}^{(b)}\}_{b=1}^B} \alpha(\mathbf{x})$

4 Decoupling Updates Converge Faster without Difficult Mathematics

Why Is D-BE Better Than C-BE?

Algorithm 1 highlights the differences between D-BE (our proposal) and C-BE. D-BE is characterized by independent initialization and per-restart updates. Concretely, after each batched evaluation, D-BE performs B independent L-BFGS-B updates—one per restart—using only its own history, thereby avoiding the off-diagonal artifacts present in C-BE.

Batching benefits not only acquisition-function evaluations but also the L-BFGS-B updates; however, the gain for the latter is limited. We now explain why. For example, a single evaluation of the expected improvement using a Gaussian process regressor with n training data points costs $\mathcal{O}(n^2 + nD)$, so evaluating B points costs $\mathcal{O}(B(n^2 + nD))$. By contrast, the L-BFGS-B update with the memory size m for B independent optimizations costs $\mathcal{O}(BmD)$. In regimes where $n \gg m$ (typically after tens to hundreds of trials, with $m \in [5, 20]$), it is cost-effective to batch only evaluations while keeping each optimizer independent. As a result, D-BE avoids the off-diagonal artifacts induced by C-BE.

Most importantly, each restart in D-BE theoretically reproduces the per-restart update trajectory as in SEQ. OPT. under the identical initialization and termination policies. Notice, however, that AD yields slightly different gradients between a batched evaluation and a single evaluation due to modulo floating-point nondeterminism, so the trajectory may not be identical. The general form of D-BE maintains a set of ongoing optimization indices $\mathcal{A} \subseteq \{1, \ldots, B\}$, and prunes the converged optimizations. This mechanism allows D-BE to progressively shrink the batch size, reducing additional computational overhead. Note that Algorithm 1 does not detail the point-wise termination owing to brevity. In contrast, C-BE employs a QN optimizer with a shared QN

state over all the restarts. Since detaching the converged optimizations is not feasible in C-BE, the overall runtime unnecessarily inflates.

Decouple L-BFGS-B Updates by Coroutine

We have discussed the benefit of using D-BE instead of C-BE. However, decoupling the L-BFGS-B updates often requires modifying the L-BFGS-B implementation directly. For example, SciPy provides L-BFGS-B while per-QN-iteration hooks are not exposed, making batched evaluations for independent optimizations non-trivial. Even in this scenario, we do not have to implement our own QN methods if we use a coroutine library. A coroutine is a special type of function that can pause its execution and be resumed later from the same point, allowing for cooperative multitasking and simplifying asynchronous programming. D-BE can be achieved by using a coroutine, which is composed of (1) one batch evaluator, and (2) B workers for each independent optimization. In the coroutine, the evaluator performs a batched evaluation, it dispatches $(\alpha^{(b)}, g^{(b)})$ to each worker, and each worker updates its QN state. Repeating this coroutine enables D-BE without any new solvers.

5 Benchmarking Experiments

We finally conduct the speedup effect and the solution quality check of D-BE against SEQ. OPT. and C-BE through a BO application. The Rastrigin function from COCO (BBOB) suite (Hansen et al. 2009, 2021) available in OptunaHub (Ozaki, Watanabe, and Yanase 2025) is taken as an objective function for BO. The benchmarking results on other functions are available in Table 2 of Appendix. Note that the optimization is performed in the minimization direction in this section. Each BO iteration first fits a Gaussian process regressor with a Matérn-ν=$5/2$ kernel on a set of observed pairs of a parameter vector and the corresponding objective value. Then, the next parameter vector is determined based on log expected improvement (LogEI). LogEI is optimized by MSO using L-BFGS-B in each BO iteration.

Table 1 shows the benchmarking results. Based on the results, all methods achieved comparable (median) final objective values under the shared iteration cap. As can be seen in the Iters. column, the L-BFGS-B iteration count drastically increased in C-BE. For example, C-BE required substantially more L-BFGS-B iterations than SEQ. OPT. as D grew, e.g., $68.8/21.5 \approx 3.2$ for $D = 20$ and $87.0/26.5 \approx 3.3$ for D=40. This observation is consistent with the discussion regarding the off-diagonal artifacts. These extra iterations can offset batching gains, making C-BE's wall-clock time comparable to—and on some objectives worse than—SEQ. OPT. By contrast, D-BE matches the iteration counts of the SEQ. OPT. baseline while exploiting batched gradient evaluations, having reduced the runtime of the acquisition function optimization ($\sim 1.5\times$ faster than SEQ. OPT.). The similar pattern holds on *Sphere*, *Attractive Sector* and *Step Ellipsoidal* from the COCO (BBOB) suite (Hansen et al. 2009, 2021); See Appendix C (Table 2) for more details. Note that while C-BE occasionally outperformed SEQ. OPT. for $D \in \{10, 20\}$, we attribute this to the difference in the convergence behavior rather than a systematic benefit of coupling. More specifically, the acquisition function optimization for C-BE is mostly terminated by the gradient tolerance but not the iteration count, meaning that the optimization results for C-BE are not premature. This leads to very different optimization results between C-BE and SEQ. OPT., making it possible for C-BE to sometimes outperform SEQ. OPT.

D	Method	Best Value ↓	Runtime (s) ↓	Iters. ↓
5	SEQ. OPT.	10.94	92.7	**11.0**
	C-BE	11.28	62.0	35.0
	D-BE	**10.85**	**59.2**	**11.0**
10	SEQ. OPT.	28.24	96.5	**14.8**
	C-BE	24.71	73.7	54.2
	D-BE	**23.89**	**67.3**	15.0
20	SEQ. OPT.	59.08	129.4	21.5
	C-BE	57.04	95.4	68.8
	D-BE	**55.81**	**86.4**	**20.0**
40	SEQ. OPT.	88.31	169.2	26.5
	C-BE	100.36	120.6	87.0
	D-BE	**76.48**	**108.2**	**25.8**

Table 1: The benchmarking results of BO with 300 trials using L-BFGS-B with the memory size of $m = 10$ and $B = 10$ restarts on the D-dimensional Rastrigin function ($D = 5, 10, 20, 40$). The termination criterion of L-BFGS-B is 200 iterations or $\|\nabla\alpha(\mathbf{x})\|_\infty \leq 10^{-2}$. Each column shows the median over 20 independent runs, each with a different random seed. **Best Value**: the minimum objective value among the 300 trials minus the best objective value over all runs. **Runtime (s)**: the total wall-clock time for BO. **Iters.**: the median of the L-BFGS-B iteration count over 300 Trials $\times 10$ restarts. Lower is better.

6 Conclusion

This paper discussed the speedup effect of BO in light of the acquisition function optimization. Although the optimization of the summed acquisition function at multiple points has been known to be a fast implementation for MSO, we empirically showed that either the convergence speed or the solution quality is degraded owing to off-diagonal artifacts. In principle, C-BE degrades the solution quality when the iteration count is capped, or it takes significantly longer when we wait until the solution quality is satisfied. Mathematically speaking, a variant of L-BFGS-B can optimize efficiently as is, but such an extension is challenging. To this end, we proposed an explicit decoupling approach to circumvent the off-diagonal artifacts. In general, such decoupling requires L-BFGS-B to take a specific design especially when the BO implementation relies on PyTorch owing to its incompatibility with multi-processing. Meanwhile, our proposition only requires a coroutine to realize the decoupling. The experiments demonstrated that our approach accelerated BO while keeping the original performance.

References

Akiba, T.; Sano, S.; Yanase, T.; Ohta, T.; and Koyama, M. 2019. Optuna: A next-generation hyperparameter optimization framework. In *International Conference on Knowledge Discovery & Data Mining*.

Ament, S.; Daulton, S.; Eriksson, D.; Balandat, M.; and Bakshy, E. 2023. Unexpected improvements to expected improvement for Bayesian optimization. *Advances in Neural Information Processing Systems*.

Balandat, M.; Karrer, B.; Jiang, D.; Daulton, S.; Letham, B.; Wilson, A.; and Bakshy, E. 2020. BoTorch: A framework for efficient Monte-Carlo Bayesian optimization. *Advances in neural information processing systems*.

Bigeon, J.; Orban, D.; and Raynaud, P. 2023. A framework around limited-memory partitioned quasi-Newton methods. *Les Cahiers du GERAD ISSN*, 711.

Byrd, R.; Lu, P.; Nocedal, J.; and Zhu, C. 1995. A limited memory algorithm for bound constrained optimization. *SIAM Journal on scientific computing*, 16.

Byrd, R.; Nocedal, J.; and Schnabel, R. 1994. Representations of quasi-Newton matrices and their use in limited memory methods. *Mathematical Programming*, 63.

Chen, Y.; Huang, A.; Wang, Z.; Antonoglou, I.; Schrittwieser, J.; Silver, D.; and de Freitas, N. 2018. Bayesian optimization in AlphaGo. *arXiv:1812.06855*.

Daulton, S.; Balandat, M.; and Bakshy, E. 2020. Differentiable expected hypervolume improvement for parallel multi-objective Bayesian optimization. *Advances in neural information processing systems*.

Feurer, M.; and Hutter, F. 2019. Automated machine learning. *Cham: Springer*.

Griewank, A.; and Toint, P. 1982. Partitioned variable metric updates for large structured optimization problems. *Numerische Mathematik*, 39.

Hansen, N.; Auger, A.; Ros, R.; Mersmann, O.; Tušar, T.; and Brockhoff, D. 2021. COCO: A platform for comparing continuous optimizers in a black-box setting. *Optimization Methods and Software*, 36.

Hansen, N.; Finck, S.; Ros, R.; and Auger, A. 2009. *Real-parameter black-box optimization benchmarking 2009: Noiseless functions definitions*. Ph.D. thesis, INRIA.

Li, C.; de Celis Leal, D. R.; Rana, S.; Gupta, S.; Sutti, A.; Greenhill, S.; Slezak, T.; Height, M.; and Venkatesh, S. 2017. Rapid Bayesian optimisation for synthesis of short polymer fiber materials. *Scientific reports*, 7.

Liu, D.; and Nocedal, J. 1989. On the limited memory BFGS method for large scale optimization. *Mathematical programming*, 45.

Loshchilov, I.; and Hutter, F. 2016. CMA-ES for hyperparameter optimization of deep neural networks. *arXiv:1604.07269*.

Nocedal, J. 1980. Updating quasi-Newton matrices with limited storage. *Mathematics of computation*, 35.

Ozaki, Y.; Watanabe, S.; and Yanase, T. 2025. OptunaHub: A Platform for Black-Box Optimization. *arXiv:2510.02798*.

Schneider, P.; Walters, W.; Plowright, A.; Sieroka, N.; Listgarten, J.; Jr, R. G.; Fisher, J.; Jansen, J.; Duca, J.; Rush, T.; et al. 2020. Rethinking drug design in the artificial intelligence era. *Nature reviews drug discovery*, 19.

Vahid, A.; Rana, S.; Gupta, S.; Vellanki, P.; Venkatesh, S.; and Dorin, T. 2018. New Bayesian-optimization-based design of high-strength 7xxx-series alloys from recycled aluminum. *Jom*, 70.

Xue, D.; Balachandran, P.; Hogden, J.; Theiler, J.; Xue, D.; and Lookman, T. 2016. Accelerated search for materials with targeted properties by adaptive design. *Nature Communications*, 7.

Appendix

A Sequential Optimization (SEQ. OPT.)

Algorithm 2 presents the SEQ. OPT. algorithm, which serves as a baseline. Each restart maintains its own optimizer state, and no batching is performed across restarts. As a result, no cross-restart information is shared, and the per-restart curvature is preserved by construction.

B Off-Diagonal Artifact Effect on BFGS

This section verifies that the observed degradation under C-BE does not stem from limiting the memory of BFGS. The setup in this section follows Section 3. Figures 3 and 4 show that C-BE injects off-diagonal mass into the inverse Hessian approximation, which should be zero, while SEQ. OPT. preserves the block-diagonal structure. Consistently, iteration counts increase markedly under C-BE relative to SEQ. OPT. as seen in Figure 5. These observations confirm that the off-diagonal artifacts are caused by coupling QN updates for a separable problem rather than by limiting the memory.

C Full Benchmarking Results

Table 2 provides the complete results for *Sphere*, *Attractive Sector*, *Step Ellipsoidal*, and *Rastrigin*. Note that the setup follows Section 5. The results show that C-BE degrades notably in higher dimensions both in solution quality and in optimizer iterations. For example, C-BE yields larger **Best Value** in $40D$ than D-BE by $+25.4\%$ on *Attractive Sector*, $+16.5\%$ on *Step Ellipsoidal*, $+31.3\%$ on *Rastrigin*, and $+2.8\%$ on Sphere [2]. The iteration counts of C-BE also inflate sharply at D=40, requiring about $3.3\times \sim 29\times$ more L-BFGS-B iterations than D-BE. The largest gap is observed on *Attractive Sector* (102.2 vs. 3.5).

By contrast, D-BE matches SEQ. OPT. in solution quality and iteration counts while achieving the best wall-clock time overall. D-BE attains the shortest median runtime in almost all objective-dimension pairs (14/16), and is up to $1.76\times$ faster than SEQ. OPT. ($5D$ *Attractive Sector*; 127.8 seconds vs. 72.5 seconds), and $1.29\times$ faster than C-BE($40D$ *Attractive Sector*; 139.5 seconds vs. 108.1 seconds).

[2] Percentages computed as (C-BE − D-BE)/D-BE using Table 2.

Algorithm 2: MSO: SEQ. OPT.

Inputs: α (Acquisition function), B (number of restarts), $\mathbf{x}_0 = \{\mathbf{x}_0^{(b)}\}_{b=1}^{B}$ (initial points)
Output: $\mathbf{x}^\star \in \operatorname{argmax}_{\mathbf{x}} \alpha(\mathbf{x})$

$\mathbf{x} \leftarrow \mathbf{x}_0$
for $b = 1, \ldots, B$ **do**
  Initialize QN optimizer O_b; set $\mathbf{x}^{(b)} \leftarrow \mathbf{x}_0^{(b)}$
  while restart b not converged **do**
    ▷ Evaluation (no batching across restarts)
    $a^{(b)} \leftarrow \alpha(\mathbf{x}^{(b)})$; $\mathbf{g}^{(b)} \leftarrow \nabla\alpha(\mathbf{x}^{(b)})$
    ▷ QN Updates
    $\mathbf{x}^{(b)} \leftarrow O_b\big(\mathbf{x}^{(b)}, a^{(b)}, \mathbf{g}^{(b)}\big)$
return $\mathbf{x}^\star = \operatorname{argmax}_{\{\mathbf{x}^{(b)}\}_{b=1}^{B}} \alpha(\mathbf{x})$

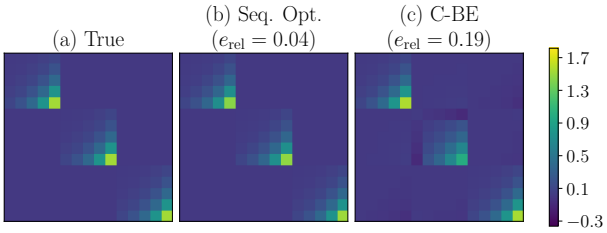

Figure 3: Contour maps of the inverse Hessian (**Left**) and its approximations by SEQ. OPT. (**Center**) and C-BE (**Right**). This figure follows the same setup as Figure 1 except that BFGS, i.e., the memory is not limited, is used instead of L-BFGS-B.

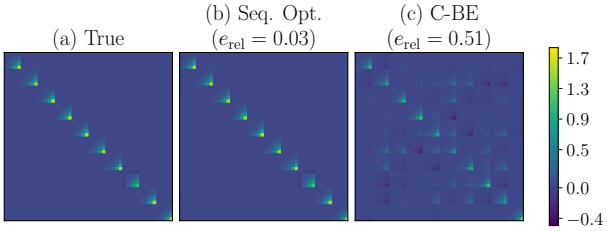

Figure 4: Contour maps of the inverse Hessian (**Left**) and its approximations by SEQ. OPT. (**Center**) and C-BE (**Right**). This figure follows the same setup as Figure 1 except that BFGS, i.e., the memory is not limited, is used instead of L-BFGS-B, and $B = 10$ is used instead of $B = 3$. Off-diagonal artifacts are more prominent for a large B.

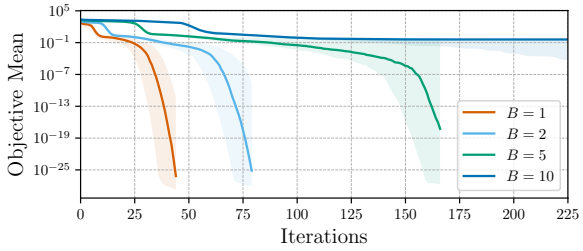

Figure 5: Convergence speed of C-BE when using BFGS. This figure follows the same setup as Figure 2 except that BFGS is used instead of L-BFGS-B. As the number of restarts B increases, the convergence of C-BE requires substantially more iterations.

Objective	Method	Best Value ↓	Runtime (s) ↓	Iters. ↓
$5D$ Sphere	SEQ. OPT.	4.026×10^{-5}	196.9	**7.0**
	C-BE	$\mathbf{1.068 \times 10^{-5}}$	**121.8**	7.5
	D-BE	5.167×10^{-5}	128.2	**7.0**
$10D$ Sphere	SEQ. OPT.	9.632×10^{-4}	213.6	13.0
	C-BE	$\mathbf{1.846 \times 10^{-4}}$	146.8	26.0
	D-BE	4.973×10^{-4}	**141.6**	**12.8**
$20D$ Sphere	SEQ. OPT.	8.471×10^{-3}	225.4	15.5
	C-BE	9.820×10^{-3}	173.1	48.8
	D-BE	$\mathbf{7.246 \times 10^{-3}}$	**147.4**	**15.0**
$40D$ Sphere	SEQ. OPT.	0.04	291.4	**21.2**
	C-BE	0.03	220.9	75.2
	D-BE	**0.03**	**190.2**	22.0
$5D$ AS	SEQ. OPT.	32.53	127.8	**12.0**
	C-BE	**32.38**	**72.2**	37.0
	D-BE	35.90	72.5	12.5
$10D$ AS	SEQ. OPT.	**59.69**	195.8	17.4
	C-BE	61.97	134.9	57.0
	D-BE	60.55	**121.9**	**17.0**
$20D$ AS	SEQ. OPT.	55.75	203.8	**6.2**
	C-BE	**52.47**	183.2	104.0
	D-BE	61.16	**151.6**	9.5
$40D$ AS	SEQ. OPT.	319.00	130.5	**3.0**
	C-BE	388.20	139.5	102.2
	D-BE	**309.60**	**108.1**	3.5
$5D$ SE	SEQ. OPT.	0.09	115.5	**12.5**
	C-BE	**0.02**	70.6	37.0
	D-BE	0.10	**69.4**	**12.5**
$10D$ SE	SEQ. OPT.	2.41	126.5	18.5
	C-BE	**1.40**	89.2	75.0
	D-BE	2.37	**77.4**	**18.2**
$20D$ SE	SEQ. OPT.	17.67	118.1	**14.1**
	C-BE	21.78	95.2	76.0
	D-BE	**17.56**	**81.1**	14.8
$40D$ SE	SEQ. OPT.	**34.32**	293.9	**23.0**
	C-BE	52.46	231.9	122.5
	D-BE	45.02	**184.6**	**23.0**
$5D$ Rastrigin	SEQ. OPT.	10.94	92.7	**11.0**
	C-BE	11.28	62.0	35.0
	D-BE	**10.85**	**59.2**	**11.0**
$10D$ Rastrigin	SEQ. OPT.	28.24	96.5	**14.8**
	C-BE	24.71	73.7	54.2
	D-BE	**23.89**	**67.3**	15.0
$20D$ Rastrigin	SEQ. OPT.	59.08	129.4	21.5
	C-BE	57.04	95.4	68.8
	D-BE	**55.81**	**86.4**	**20.0**
$40D$ Rastrigin	SEQ. OPT.	88.31	169.2	26.5
	C-BE	100.36	120.6	87.0
	D-BE	**76.48**	**108.2**	**25.8**

Table 2: The benchmarking results of BO with 300 trials using L-BFGS-B with the memory size of $m = 10$ and $B = 10$ restarts on the D-dimensional functions ($D = 5, 10, 20, 40$). We picked four objectives: Sphere, Attractive Sector (AS), Step Ellipsoidal (SE), and Rastrigin. The termination criterion of L-BFGS-B is 200 iterations or $\|\nabla\alpha(\mathbf{x})\|_\infty \leq 10^{-2}$. Each column shows the median over 20 independent runs, each with a different random seed. **Best Value**: the minimum objective value among the 300 trials minus the best objective value over all runs. **Runtime (s)**: the total wall-clock time for BO. **Iters.**: the median of the L-BFGS-B iteration count over 300 Trials $\times 10$ restarts. Lower is better.