Decreasing the Computing Time of Bayesian Optimization using Generalizable Memory Pruning

Alexander E. Siemenn
Department of Mechanical Engineering
Massachusetts Institute of Technology
Cambridge, MA, USA
asiemenn@mit.edu

Tonio Buonassisi
Department of Mechanical Engineering
Massachusetts Institute of Technology
Cambridge, MA, USA
buonassisi@mit.edu

***Abstract*—Bayesian optimization (BO) suffers from long computing times when processing highly-dimensional or large data sets. These long computing times are a result of the Gaussian process surrogate model having a polynomial time complexity with the number of experiments. Running BO on high-dimensional or massive data sets becomes intractable due to this time complexity scaling, in turn, hindering experimentation. Alternative surrogate models have been developed to reduce the computing utilization of the BO procedure, however, these methods require mathematical alteration of the inherit surrogate function, pigeonholing use into only that function. In this paper, we demonstrate a generalizable BO wrapper of memory pruning and bounded optimization, capable of being used with any surrogate model and acquisition function. Using this memory pruning approach, we show a decrease in wall-clock computing times per experiment of BO from a polynomially increasing pattern to a sawtooth pattern that has a non-increasing trend without sacrificing convergence performance. Furthermore, we illustrate the generalizability of the approach across two unique data sets, two unique surrogate models, and four unique acquisition functions. All model implementations are run on the MIT Supercloud state-of-the-art computing hardware.**

***Index Terms*—efficient computing, bounded search, time complexity scaling, generalizable optimization, data pruning**

I. Introduction

Bayesian optimization (BO) is a data-based global optimization tool that discovers optima without an analytical model of the response function [1]–[3]. A standard BO procedure consists of two primary steps: (1) using a surrogate model to estimate the topology of the target response function given a collection of input data and (2) acquiring new suggested experimental conditions to run based on the estimated surrogate model means and variances [4], [5]. For the first step, a common surrogate model used in BO is a Gaussian Process (GP) regression. GPs model complex, multi-dimensional input-output response relationships using a mixture of kernel functions that interpolate the missing space between collected experiments [6]–[8]. For the second step, a mathematical figure of merit called an acquisition function (AF), acquires new experimental conditions to run, governed by balancing the exploitation of regions of low predicted response function means (for a minimization problem) and the exploration of regions of high predicted response function variances [4], [6], [9]. The interleaving steps of response function estimation *via* surrogate model computation and acquisition of new experiments leverage the estimation power of the surrogate model to discover the optima of challenging experimental problems where it may be otherwise intractable to develop an analytical model representative of the response function [10]–[12].

However, as the complexity or dimensionality of the response function increases, more experimental data points, N, are required for accurate estimation of the response function's surrogate model [13], [14]. This increased data requirement of the surrogate model becomes problematic because the time required to compute a GP regression increases polynomially following the scaling law $O(N^3)$ [7], [8], [15]–[17]. Both *in silico* and *in situ* optimization experiments can be significantly bottlenecked by this unfavorable scaling law if large volumes of data are being collected, hence, by selectively processing subsets of this data in tandem with bounded optimization, the computing times of the BO process can be reduced.

In this study, we explore the use of memory pruning and bounded surrogate models as a method to decrease the number of required experimental data points needed to accurately run an online BO procedure, therefore, decreasing the computing time of optimization. We benchmark the computing times of two surrogate models: (1) a GP and (2) a pre-trained neural network, each with four acquisition functions: (1) expected improvement (EI), (2) lower confidence bound (LCB), (3) EI Abrupt, and (4) LCB Adaptive, all run on the MIT Supercloud, a high-performance supercomputer consisting of Nvidia Volta V100 GPUs [18]. Existing literature on decreasing the computing time of BO conventionally alters the mathematics of a surrogate model to make computation more efficient [5], [7], [9], [19], [20], however, this constrains the user to only this newly developed surrogate for optimization.

In this contribution, we demonstrate the use of a generalized method of memory pruning and search space bounding to efficiently decrease BO computing times without constraining the procedure to a single surrogate model or AF. Furthermore, we demonstrate the reduction of computing times on two relevant problems: (1) optimization of a 6-dimensional ana-

A.E.S. and T.B. thank First Solar and the Acceleration Consortium for their support and fruitful discussions. The authors acknowledge the MIT SuperCloud and Lincoln Laboratory Supercomputing Center for providing HPC resources that have contributed to the research results reported within this paper.

lytical Ackley function to demonstrate relevance for *in silico* experimentation and (2) optimization of a 5-dimensional real-world data set of inorganic crystalline material band gaps to demonstrate relevance for *in situ* experimentation.

II. Related Work

Existing literature exists on decreasing the computing time of BO, however, most of this literature requires significant changes to the mathematical structure of the GP or AF. For example, a common method of decreasing the computing time of BO is to implement a Sparse Pseudo-input GP (SPGP) [7], [19]–[21]. A standard GP is non-parametric in nature, meaning that when constructing a prediction, the entire prior training data set is required to compute the response function of a target variable [7]. Instead of using the full number of training data, N, to compute this response function, an SPGP uses a pseudo data set of size $M < N$, such that

$$\mathrm{X}_{\mathrm{SPGP}} = \{\mathrm{X}_{\mathrm{GP}}\}_{m=1}^{M}, \tag{1}$$

where $\mathrm{X}_{\mathrm{SPGP}}$ is the set of input data used to compute the prediction in an SPGP and X_{GP} is the set of input data used to compute the predicted response in a standard GP. Hence, the spacing between pseudo data points is known. Moreover, $||\mathrm{X}_{\mathrm{SPGP}}|| = M$ and $||\mathrm{X}_{\mathrm{GP}}|| = N$. This reforged structure of a GP into an SPGP enables computing time decreases on the order of $O(N^3) \rightarrow O(NM^2)$ since $M < N$.

Another method to decrease the computing time of BO is efficient global optimization (EGO) [5], [9], [22]. Similar to standard BO, EGO implements a surrogate model to generate the input-output response function, however, EGO can acquire a global optimum in fewer online iterations than BO by bounding the derivatives of the acquisition function relative to either the target variable or the surrogate standard error [5]:

$$\begin{aligned} \frac{\delta \mathrm{EI}(\mathrm{X})}{\delta y(\mathrm{X})} &< 0 \quad \text{and} \\ \frac{\delta \mathrm{EI}(\mathrm{X})}{\delta s(\mathrm{X})} &> 0, \end{aligned} \tag{2}$$

where EI is the Expected Improvement acquisition function defined in the next section, y is the target response variable and s is the standard error of the surrogate. Additionally, van Stein *et al.* [23] further decrease the computing time of EGO by parallelizing the computation of the gradients.

In order to decrease the computing time of BO, the methods mentioned above either (1) make significant changes to the surrogate model or (2) rely on computing the gradients of the AF to bound the search space. A downfall of computing the gradients of an AF is immediately constraining the AF of choice to be differentiable. Thus, the EGO studies above are constrained to using only the expected improvement AF and cannot use other AFs, even if it may be more advantageous. Therefore, in this study, we implement methods of decreasing the computing time of BO that do not constrain the user to select a certain surrogate model or AF to run the procedure. Instead, the method used in this paper is a lightweight implementation of a simple space-bounding method, not requiring gradients, from which new experiments are acquired. Furthermore, selective pruning of memory data from outside the bounded search space drives a significant decrease in BO compute time relative to standard BO. Hence, the method described in the next section supports the use of any surrogate model or AF. In this paper, we illustrate the computing times achieved using a GP surrogate model and four different AFs.

III. Methods

In this paper, we extend the implementation of a BO wrapper developed by Siemenn *et al.* [24] that bounds the acquisition and search space while pruning old memory data that lay outside of these computed bounds. This approach is entitled Zooming Memory-Based Initialization (ZoMBI) and is described further in [24] with code publicly available. In brief, for a minimization objective, f, the bounds for each dimension, d, are computed uniquely based on the $\min(\mathrm{X}_d)$ and the $\max(\mathrm{X}_d)$ of the m best-performing memory points, *i.e.*, the points that achieve the m lowest target f values, from the set X. For every loop, all data points that lie outside of the constrained space will be pruned from memory. This is computationally favorable because as the search bounds iteratively zoom in, the target space inside the bounds increases in resolution by the surrogate model while all other space decreases in resolution. A standard GP surrogate model as well as a neural network (NN) surrogate model are used in the ZoMBI optimization procedure to demonstrate its generalizability to several unique surrogate models.

The computing times of four unique AFs implemented using ZoMBI are benchmarked against their standard BO counterparts. These four AFs are expected improvement (EI), lower confidence bound (LCB), EI Abrupt, and LCB Adaptive. Each of these mathematical figures of merit uniquely balances the exploitation of surrogate posterior means and the exploration of surrogate posterior variances.

EI is defined as [25]–[27]:

$$\begin{aligned} a_{\mathrm{EI}}(\mathrm{X}, \mathrm{Y}; \xi, \eta) &= (\mu(\mathrm{X}) - \min(\mathrm{Y}) - \xi)\, \Phi(Z) + \sigma(\mathrm{X})\psi(Z), \\ \text{where} \quad Z &= \frac{\mu(\mathrm{X}) - \min(\mathrm{Y}) - \xi}{\sigma(\mathrm{X})}, \end{aligned} \tag{3}$$

where X is the set of input data $\{x_1, x_2, ...x_N\}$, $x_j \in \mathbb{R}^d$ for d dimensions, Y is the set of corresponding response values $\{y_1, y_2, ...y_N\}$, $y_j \in \mathbb{R}$, ξ is a hyperparameter tuned to favor exploration or exploitation of the surrogate, and $\Phi(\cdot)$ and $\psi(\cdot)$ are the normal cumulative and probability density functions, respectively. EI strikes a balance between exploration and exploitation while considering the prior best-performing response variable of the set, $\min(\mathrm{Y})$.

LCB is defined as [28], [29]:

$$a_{\mathrm{LCB}}(\mathrm{X}; \beta) = \mu(\mathrm{X}) - \beta\sigma(\mathrm{X}), \tag{4}$$

where β is a hyperparameter tuned to factor exploration or exploitation of the surrogate means, μ, and variances σ. A

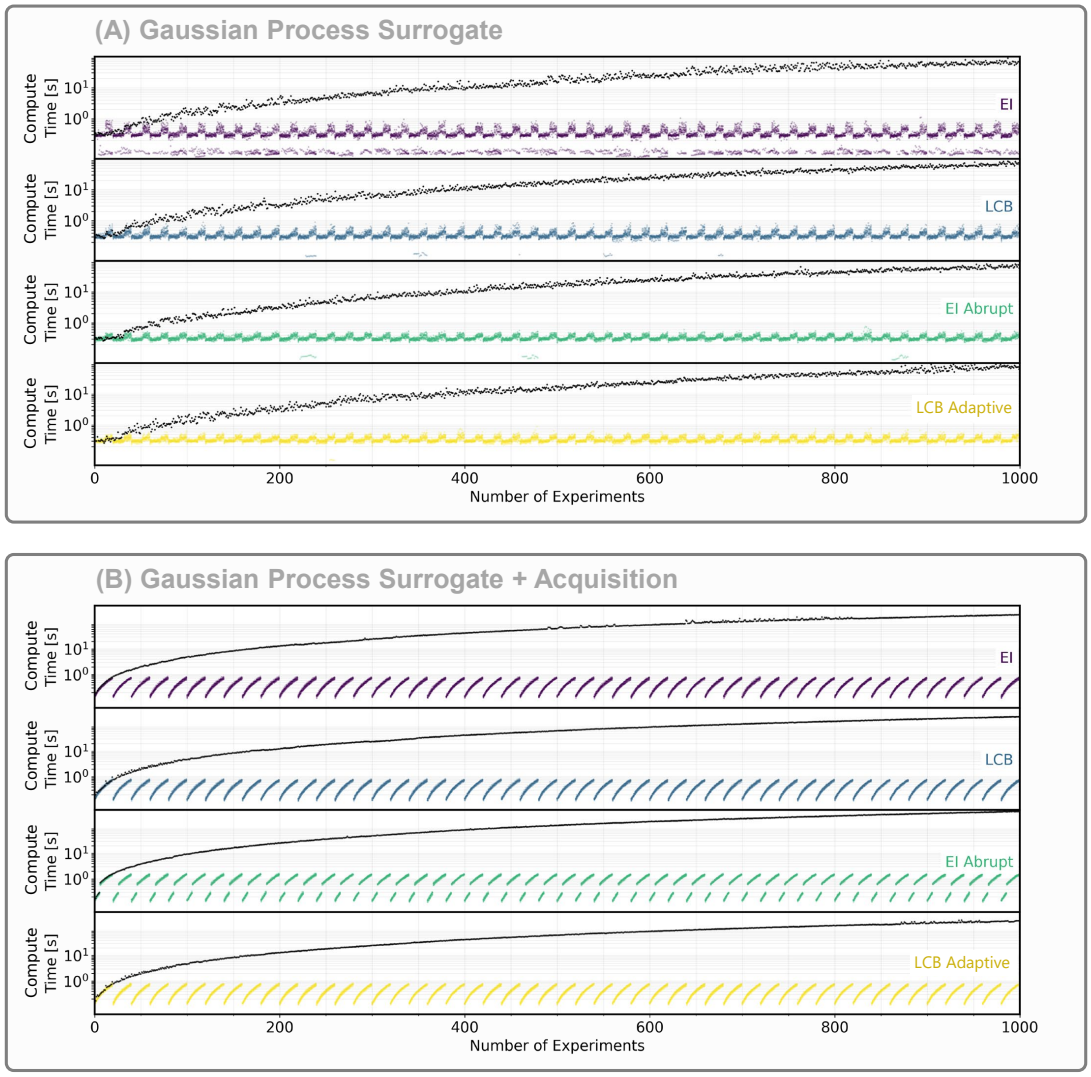

Fig. 1: Computing times of a Bayesian optimization procedure with a Gaussian process surrogate model. (A) Wall-clock computing times per experiment for only the GP computation component across N data points on a 6D analytical Ackley function. (B) Wall-clock computing times per experiment for the acquisition of new data from a GP computed across a mesh grid of 10k data points on a 6D analytical Ackley function. Each panel represents the computing times of four AFs, from top to bottom: EI, LCB, EI Abrupt, and LCB Adaptive. The colored scatter points represent the compute times per experiment of twelve independent optimization procedures using the memory pruning ZoMBI method for each AF. The black scatter points represent the benchmark compute times per experiment of one independent optimization using standard BO for each AF. For ZoMBI, $N \leq 20$ *via* memory pruning and for standard BO, N is the number of experiments. All compute times are wall-clock compute times measured from the MIT Supercloud Nvidia Volta V100 GPUs. The y-axes are shown in log scale.

higher β favors exploration of surrogate variances while a lower β favors exploitation of surrogate means.

EI Abrupt is defined as [24]:

$$a_{\text{EI Abrupt}}(\mathrm{X}, \mathrm{Y}; \beta, \xi, \eta) = \begin{cases} a_{\text{EI}}(\mathrm{X}, \mathrm{Y}; \xi, \eta), & \text{if } |\Delta\{y_{N-3\ldots N}\}| \leq \eta \\ a_{\text{LCB}}(\mathrm{X}; \beta), & \text{otherwise} \end{cases} \quad (5)$$

where the mode of acquisition is abruptly switched between EI and LCB depending on if the finite difference between the $\{y_{N-3\ldots N}\}$ previous response values is below a hyperparameter threshold, η. EI Abrupt provides another level of tunable exploration-exploitation by actively swapping between these modes as more data is collected.

LCB Adaptive is defined as [24], [30], [31]:

$$a_{\text{LCB Adaptive}}(\mathrm{X}, N; \beta, \epsilon) = \mu(\mathrm{X}) - \epsilon^{N}\beta\sigma(\mathrm{X}), \quad (6)$$

where the hyperparameter, β, is actively tuned as the number of collected data points, $N = ||\mathrm{X}||$, increases. LCB Adaptive exponentially decays from being more explorative to then becoming more exploitative as N increases. Since ZoMBI

actively prunes the set X, which decreases N until more experiments are collected, LCB Adaptive is always switching acquisition modes throughout the optimization procedure.

In this paper, we demonstrate the computing times of each of these AFs implemented with the ZoMBI bounding and pruning method as well as implemented with just standard BO. The computing times are further bifurcated into (1) the surrogate model compute times per experiment alone on N data points and (2) the surrogate + AF compute times per experiment on a mesh grid of 10k data points since acquisition of new data points requires the computation of the surrogate across a mesh grid of points in the space.

First, the computing times of each AF with a GP surrogate model are measured on a 6-dimensional analytical Ackley function [32] for 1000 experiments for both ZoMBI and standard BO implementations. Second, the computing times of just the ZoMBI AF implementations with a NN surrogate model for 200 experiments are measured on a real-world 5-dimensional data set of inorganic crystalline material band gaps, available as open-access from Materials Project [33]. Both of these experiments are run on the high-performance supercomputer, MIT Supercloud to measure the wall-clock computing times of both the surrogate models and the acquisition functions [18].

IV. Results

A. Gaussian Process Surrogate on a 6D Analytical Data Set

In this section, we demonstrate a decrease in BO computing times, relative to standard BO, using the ZoMBI method of memory pruning on an *in silico* optimization experiment of an analytical 6-dimensional Ackley function [24], [32]. This *in silico* optimization experiment is run on the MIT Supercloud Nvidia Volta V100 GPU [18].

Figure 1 illustrates (A) the time to compute a GP surrogate model for each iterative experiment and (B) the time to acquire new data points by computing the GP surrogate across a mesh grid for each iterative experiment. One operation is included in the measurement of GP compute time: (1) the fitting of training data, X, to a GP model using a mixture of kernel functions, in this case, Matern 5/2 kernels. Two operations are included in the measurement of GP + acquisition compute time: (1) the prediction and storage of the response values Y from the GP for a mesh grid of 10k data points from a bounded set of X and (2) the computation of the acquisition figure of merit from one of Equations 3–6, hence, GP + acquisition computing times are higher than just GP computing times alone.

In Figure 1(A), a square wave pattern in ZoMBI computing times is shown by the colored scatter points. This is a result of the ZoMBI process selecting the top performing acquired experiments every $N = 20$ experiments and pruning the rest from the memory to bound the surrogate mesh grid computation. Hence, for every 20 experiments, a drop in computing times is noted for ZoMBI, whereas computing times for standard BO continue to increase polynomially per experiment as additional experiments are collected to calculate the GP. As a result of this memory pruning, ZoMBI computing times per experiment demonstrate a non-increasing trend, even after 1000 experiments are acquired. Therefore, the memory pruning procedure significantly decreases the computing time per experiment relative to standard BO. Furthermore, a low spread between scatter points plotted from each of the twelve independent trials demonstrates the high reproducibility of results using the MIT Supercloud GPUs.

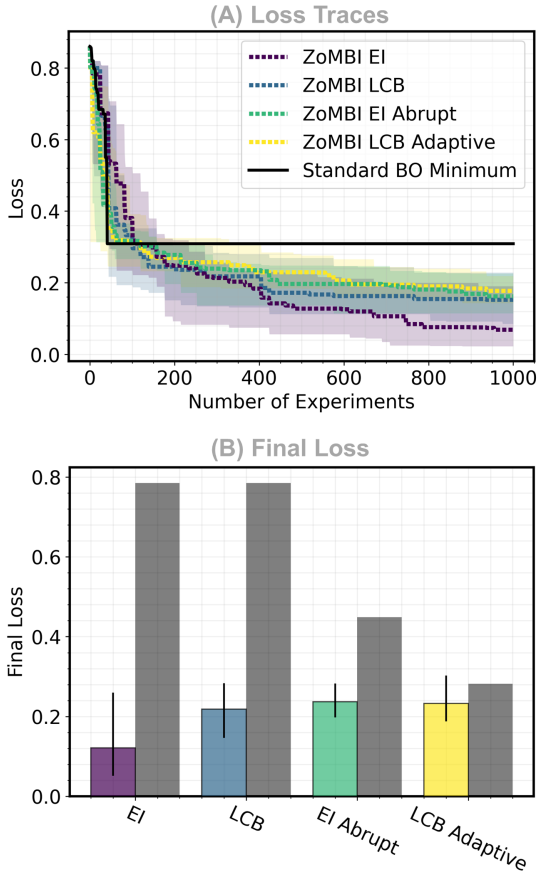

Fig. 2: Loss of standard and ZoMBI Bayesian optimization on a 6-dimensional Ackley function. (A) Loss traces over the 1000-experiment optimization procedure from Figure 1. Only the minimum standard BO loss trace is shown for clarity. (B) Final loss values after 1000 sampled experiments. The colored bars and traces illustrate the median minimum values discovered by twelve independent trials of the memory pruning ZoMBI method for each AF with the 5th and 95th percentile indicated by (A) the shaded region and (B) the error bars. The grey bars illustrate the minimum values discovered by one independent trial of standard BO for each AF.

Similar to the GP surrogate computing time results, a significant decrease in computing times using memory pruning is demonstrated for the acquisition of new data points, shown in Figure 1(B). A log-transformed sawtooth pattern is shown between memory pruning steps where a drop in compute time occurs. Again, low variance between the twelve independent trials is demonstrated due to the high overlap between scatter

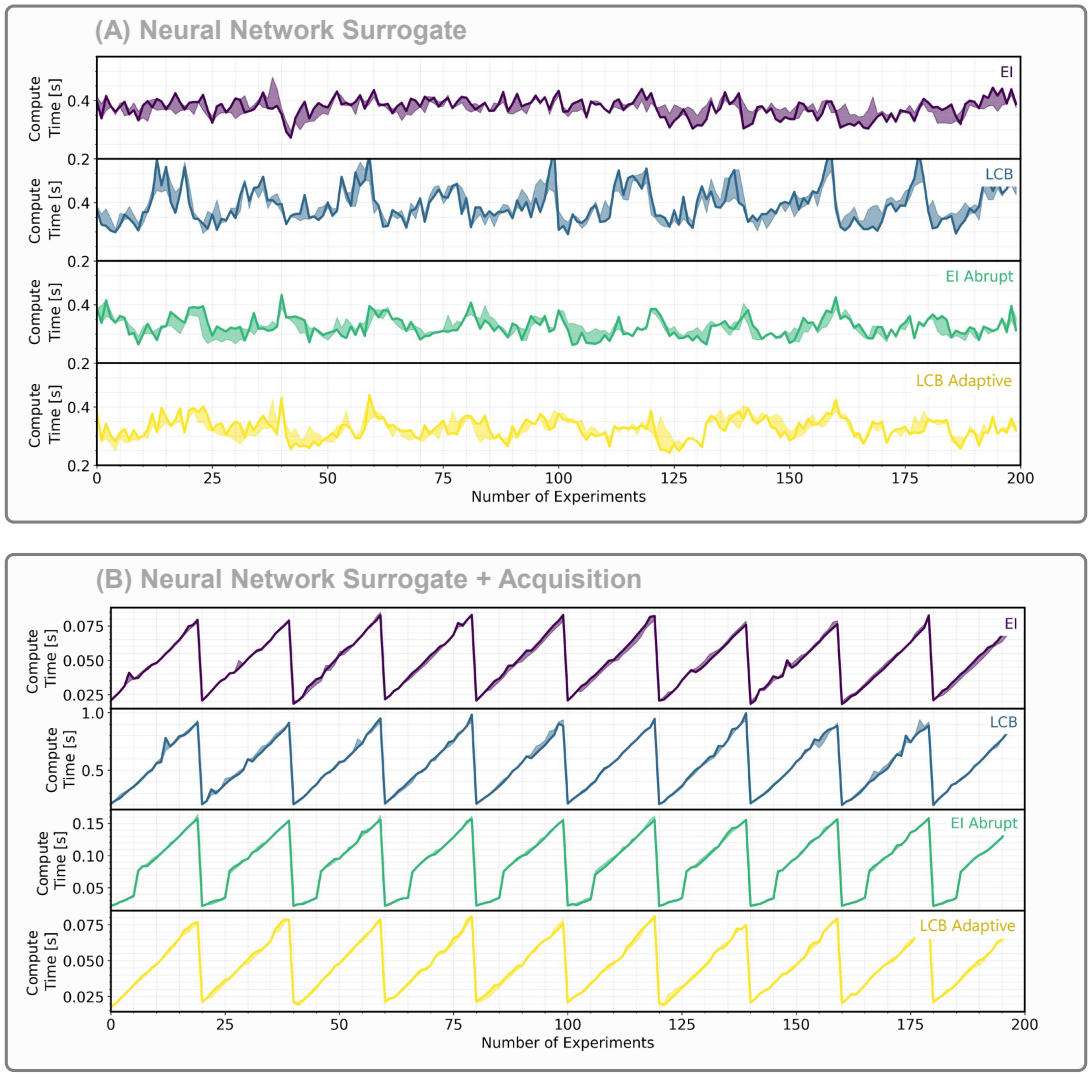

Fig. 3: Computing times of a Bayesian optimization procedure with a pre-trained neural network surrogate model. (A) Wall-clock computing times per experiment for only the NN computation component across N data points on a 5D real-world materials data set. (B) Wall-clock computing times per experiment for the acquisition of new data from a NN computed across a mesh grid of 10k data points on a 5D real-world materials data set. Each panel represents the computing times of four AFs, from top to bottom: EI, LCB, EI Abrupt, and LCB Adaptive. The median values are shown by the solid line and the 5th and 95th percentile range is shown by the shaded region. All compute times are wall-clock compute times measured from the MIT Supercloud Nvidia Volta V100 GPUs.

points. Using ZoMBI, the time to compute the GP surrogate alone is approximately 0.2 seconds per experiment (Figure 1(A)), while the time to compute and acquire new data points from the surrogate model takes approximately 1 second per experiment (Figure 1(B)). This difference arises due to the number of computations being performed: Figure 1(A) computes the GP across only $N \leq 20$ points while Figure 1(B) computes the GP and the acquisition value (Equations 3–6) across 10k points in a mesh grid to acquire new data points. After 1000 experiments are collected, the ZoMBI method still achieves computing times of 1 second per experiment, however, the standard BO method polynomially approaches compute times of 100 seconds per experiment, a factor of 100x slower. Therefore, computing times of BO are significantly reduced using a memory pruning and bounded optimization approach. But, does this pruning and bounding process adversely impact optimization performance?

Figure 2 illustrates the convergence of ZoMBI and standard BO on the global minimum of the 6-dimensional Ackley function. Not only does this memory pruning and bounded optimization procedure not adversely impact optimization performance, but it is also demonstrated to outperform standard BO on the 6-dimensional Ackley function. ZoMBI EI achieves the lowest function values after 1000 experiments with LCB,

then EI Abrupt, then LCB Adaptive following, in that order. The reverse is noted for standard BO. This implies that without the memory pruning and search space bounding features of ZoMBI, the actively adapting acquisition functions, EI Abrupt and LCB Adaptive, perform better than the conventional EI and LCB acquisition functions. Moreover, we note that standard BO, shown as the black trace in Figure 2(A), stops learning after fewer than 50 experiments due to local minima and the sharpness of the Ackley function global minimum [32], [34] while all ZoMBI methods continue to learn by continuously zooming in the search space bounds.

B. Neural Network Surrogate on a 5D Real-world Data Set

In this section, we demonstrate a decrease in BO computing times, relative to standard BO, using the ZoMBI method of memory pruning on an optimization problem translatable to *in situ* experimentation. The data set optimized is a 5-dimensional open-access data set of inorganic crystals with the objective of optimizing the properties density, formation energy, energy above hull, Fermi energy to find a material with 1.4eV band gap [33], [35]. A pre-trained NN is used as the surrogate model instead of the GP to demonstrate the generalizability of the memory pruning method to various surrogate models.

Figure 3 illustrates the time to fit a pre-trained NN to the set X on the MIT Supercloud [18]. Figure 3(A) illustrates fitting the NN surrogate to a maximum of $N = 20$ points using ZoMBI, whereas Figure 3(B) illustrates fitting the NN and computing the respective AF to a mesh grid of 10k points using ZoMBI. The combination of the NN fitting to few data and also being pre-trained produces a noisy trace of computing times in Figure 3(A). However, as the number of fitting points increases from 20 to 10k, a much clearer trend in computing times can be seen in Figure 3(B).

Similar to the GP surrogate results on the 6D Ackley function in Figure 1, a sawtooth pattern, resetting every 20 experiments is shown for the NN + AF computing times in Figure 3 for the 5D real-world data set. Although each of these AFs has a similar structure to their compute time curves, each y-axis has a different scale, and LCB is noted to have the highest computing time. This is likely due to LCB's explorative nature constantly generating a wide search bound which encompasses many more data points when compared to any of the greedier AF methods. Furthermore, an interesting pattern is seen in the EI Abrupt curves where the first rising segment has a different structure than the second rising segment, this is the abrupt switch between EI and LCB sampling modes that changes the bounding and, in turn, changes the number of data points kept in memory.

Overall, the NN surrogate run on the 5D real-world data set produces similar non-increasing computing times per experiment to the GP surrogate run on the 6D analytical Ackley function. Hence, demonstrating the potential for the ZoMBI memory pruning and bounding optimization method to be generalizable to various surrogate models, without modification, to decrease the computing time of BO.

V. Summary & Conclusions

In this paper, we demonstrate the capabilities of search space bounding and memory pruning in Bayesian optimization to significantly decrease the optimization procedure's computing time. We demonstrate this decrease in compute time by up to 100x across two unique data sets, two unique surrogate models, and four unique acquisition functions, all of which are run on the high-performance MIT Supercloud supercomputer [18].

The method of bounding and memory pruning using Zooming Memory-Based Initialization (ZoMBI) [24] implemented in this paper takes the best-performing memory points and uses those values to construct a constrained search region for the acquisition function to sample from. Upon consecutive constraints, prior data points that lay outside of these bounds are pruned from memory, decreasing the number of data points used to fit a surrogate model, in turn, decreasing the time required to compute the surrogate model and its acquisition function.

We demonstrate that this iterative constraining and pruning process achieves a sawtooth computing time pattern per experiment, relative to standard BO that exhibits a polynomially increasing computing time trend following $O(N^3)$ for N experiments. The sawtooth computing time pattern is shown to reset back to near-zero after each memory pruning update, hence, producing a non-increasing computing time trend per experiment. Furthermore, this decreased computing time is shown to persist across analytical and real-world data sets, across Gaussian Process regression and neural network surrogate models, and across four acquisition functions: expected improvement, lower confidence bound, abrupt expected improvement, and adaptive lower confidence bound. The results demonstrated in this paper are also shown to be reproducible with low variance across several independent trials by being run on the MIT Supercloud supercomputer. Hence, in this paper, we demonstrate the reproducibility and generalizability of the proposed ZoMBI memory pruning and bounded optimization method to decrease the computing times of Bayesian optimization across a variety of data sets, surrogate models, and acquisition functions.

References

[1] 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*, vol. 86, pp. 97–106, 3 1964.

[2] S. Greenhill, S. Rana, S. Gupta, P. Vellanki, and S. Venkatesh, "Bayesian optimization for adaptive experimental design: A review," *IEEE Access*, vol. 8, pp. 13 937–13 948, 2020.

[3] B. Shahriari, K. Swersky, Z. Wang, R. P. Adams, and N. D. Freitas, "Taking the human out of the loop: A review of bayesian optimization," *Proceedings of the IEEE*, vol. 104, pp. 148–175, 1 2016.

[4] E. Brochu, V. M. Cora, and N. de Freitas, "A Tutorial on Bayesian Optimization of Expensive Cost Functions, with Application to Active User Modeling and Hierarchical Reinforcement Learning," 2010.

[5] D. R. Jones, M. Schonlau, and W. J. Welch, "Efficient Global Optimization of Expensive Black-Box Functions," *J. Glob. Optim.*, vol. 13, pp. 455–492, 1998.

[6] M. Seeger, "Gaussian processes for machine learning." *Int. J. Neural Syst.*, vol. 14, no. 2, pp. 69–106, 2004.

[7] E. Snelson and Z. Ghahramani, *Sparse Gaussian Processes using Pseudo-inputs*, Y. Weiss, B. Schölkopf, and J. Platt, Eds. MIT Press, 2005, vol. 18.

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

[9] P. Hennig and C. J. Schuler, "Entropy search for information-efficient global optimization," *Journal of Machine Learning Research*, vol. 13, pp. 1809–1837, 2012.

[10] J. R. Deneault, J. Chang, J. Myung, D. Hooper, A. Armstrong, M. Pitt, and B. Maruyama, "Toward autonomous additive manufacturing: Bayesian optimization on a 3d printer," *MRS Bulletin*, vol. 46, pp. 566–575, 7 2021.

[11] A. E. Siemenn, E. Shaulsky, M. Beveridge, T. Buonassisi, S. M. Hashmi, and I. Drori, "A Machine Learning and Computer Vision Approach to Rapidly Optimize Multiscale Droplet Generation," *ACS Appl. Mater. Interfaces*, vol. 14, no. 3, pp. 4668–4679, 2022.

[12] Z. Liu, N. Rolston, A. C. Flick, T. W. Colburn, Z. Ren, R. H. Dauskardt, and T. Buonassisi, "Machine learning with knowledge constraints for process optimization of open-air perovskite solar cell manufacturing," *Joule*, vol. 6, no. 4, pp. 834–849, 2022.

[13] H. Sayama, I. Pestov, J. Schmidt, B. J. Bush, C. Wong, J. Yamanoi, and T. Gross, "Modeling complex systems with adaptive networks," *Computers & Mathematics with Applications*, vol. 65, no. 10, pp. 1645–1664, 2013.

[14] D. H. Wolpert and D. R. Wolf, "Estimating functions of probability distributions from a finite set of samples," *Physical Review E*, vol. 52, no. 6, p. 6841, 1995.

[15] C. Li, S. Gupta, S. Rana, V. Nguyen, S. Venkatesh, and A. Shilton, "High Dimensional Bayesian Optimization Using Dropout," *Proc. 26th Int. Jt. Conf. Artif. Intell. IJCAI*, 2017.

[16] Z. Wang, C. Li, S. Jegelka, and P. Kohli, "Batched High-dimensional Bayesian Optimization via Structural Kernel Learning," *Proc. 34th Int. Conf. Mach. Learn. Sydney, Aust. PMLR*, vol. 70, 2017.

[17] G. Lan, J. M. Tomczak, D. M. Roijers, and A. E. Eiben, "Time Efficiency in Optimization with a Bayesian-Evolutionary Algorithm," 2020.

[18] A. Reuther, J. Kepner, C. Byun, S. Samsi, W. Arcand, D. Bestor, B. Bergeron, V. Gadepally, M. Houle, M. Hubbell, M. Jones, A. Klein, L. Milechin, J. Mullen, A. Prout, A. Rosa, C. Yee, and P. Michaleas, "Interactive supercomputing on 40,000 cores for machine learning and data analysis," *2018 IEEE High Perform. Extrem. Comput. Conf.*, pp. 1–6, 2018.

[19] F. Leibfried, V. Dutordoir, S. T. John, and N. Durrande, "A Tutorial on Sparse Gaussian Processes and Variational Inference," 2021.

[20] M. Titsias, "Variational learning of inducing variables in sparse gaussian processes," *Proc. Mach. Learn. Res.*, vol. 5, pp. 567–574, 16–18 Apr 2009.

[21] M. McIntire, D. Ratner, and S. Ermon, "Sparse gaussian processes for bayesian optimization," in *Conference on Uncertainty in Artificial Intelligence*, 2016.

[22] S. Jeong and S. Obayashi, "Efficient global optimization (ego) for multi-objective problem and data mining," *2005 IEEE Congress on Evolutionary Computation, IEEE CEC 2005. Proceedings*, vol. 3, pp. 2138–2145, 2005.

[23] B. van Stein, H. Wang, and T. Back, "Automatic configuration of deep neural networks with parallel efficient global optimization," *2019 Int. Jt. Conf. Neural Networks*, pp. 1–7, 2019.

[24] A. E. Siemenn, Z. Ren, Q. Li, and T. Buonassisi, "Fast bayesian optimization of needle-in-a-haystack problems using zooming memory-based initialization (zombi)," *npj Computational Materials*, vol. 9, pp. 1–13, 5 2023.

[25] J. Mockus, V. Tiesis, and A. Zilinskas, "The application of bayesian methods for seeking the extremum," pp. 117–129, 1978.

[26] V. R. Šaltenis, "One method of multiextremum optimization," *Automatic Control and Comput. Sci.*, pp. 33–38, 1971.

[27] D. Zhan and H. Xing, "Expected improvement for expensive optimization: a review," *Journal of Global Optimization*, vol. 78, pp. 507–544, 11 2020.

[28] P. Auer, "Using confidence bounds for exploitation-exploration tradeoffs," *J. Mach. Learn. Res.*, vol. 3, pp. 397–422, 2002.

[29] D. D. Cox and S. John, "A statistical method for global optimization," *Conference Proceedings - IEEE International Conference on Systems, Man and Cybernetics*, vol. 1992-January, pp. 1241–1246, 1992.

[30] N. Srinivas, A. Krause, S. Kakade, and M. Seeger, "Gaussian process optimization in the bandit setting: No regret and experimental design," *Proc. 27th Int. Conf. Mach. Learn. Haifa, Isr. 2010*, pp. 1015–1022, 2010.

[31] F. Häse, L. M. Roch, C. Kreisbeck, and A. Aspuru-Guzik, "Phoenics: A bayesian optimizer for chemistry," *ACS Cent. Sci.*, vol. 4, pp. 1134–1145, 2018.

[32] D. H. Ackley, *A connectionist machine for genetic hillclimbing*. Kluwer Academic Publishers, 1987.

[33] A. Jain, S. P. Ong, G. Hautier, W. Chen, W. D. Richards, S. Dacek, S. Cholia, D. Gunter, D. Skinner, G. Ceder, and K. A. Persson, "Commentary: The Materials Project: A materials genome approach to accelerating materials innovation," *APL Mater.*, vol. 1, no. 1, p. 011002, 2013.

[34] G. Merkuryeva and V. Bolshakovs, "Benchmark fitness landscape analysis," *International Journal of Simulation Systems, Science and Technology*, vol. 12, no. 2, pp. 38–45, 2011.

[35] M. De Jong, W. Chen, T. Angsten, A. Jain, R. Notestine, A. Gamst, M. Sluiter, C. K. Ande, S. Van Der Zwaag, J. J. Plata, C. Toher, S. Curtarolo, G. Ceder, K. A. Persson, and M. Asta, "Charting the complete elastic properties of inorganic crystalline compounds," *Sci. Data*, vol. 2, no. 1, pp. 1–13, 2015.