Learning Surrogates for Offline Black-Box Optimization via Gradient Matching

Minh Hoang [1] **Azza Fadhel** [2] **Aryan Deshwal** [2] **Janardhan Rao Doppa** [2] **Trong Nghia Hoang** [2]

Abstract

Offline design optimization problem arises in numerous science and engineering applications including material and chemical design, where expensive online experimentation necessitates the use of *in silico* surrogate functions to predict and maximize the target objective over candidate designs. Although these surrogates can be learned from offline data, their predictions are often inaccurate outside the offline data regime. This challenge raises a fundamental question about the impact of imperfect surrogate model on the performance gap between its optima and the true optima, and to what extent the performance loss can be mitigated. Although prior work developed methods to improve the robustness of surrogate models and their associated optimization processes, a provably quantifiable relationship between an imperfect surrogate and the corresponding performance gap, as well as whether prior methods directly address it, remain elusive. To shed light on this important question, we present a theoretical framework to understand offline black-box optimization, by explicitly bounding the optimization quality based on how well the surrogate matches the latent gradient field that underlines the offline data. Inspired by our theoretical analysis, we propose a principled black-box gradient matching algorithm to create effective surrogate models for offline optimization, improving over prior approaches on various real-world benchmarks.

1. Introduction

Many science and engineering applications involve optimizing an *expensive-to-evaluate black-box objective function* over large design spaces. Some examples include design optimization over candidate molecules, proteins (Nguyen & Daugherty, 2005), drugs, biological sequences, and superconducting materials (Si et al., 2016). To evaluate candidate designs, we need to perform physical lab experiments or computational simulations which are labor-intensive and impractical to do in an online manner. *Offline optimization* (Trabucco et al., 2022; 2021) is a more practical setting where we assume the access to a dataset of input and objective function evaluation pairs, and the overall goal is to use this offline training data to uncover optimal designs.

The prototypical approach (Hutter et al., 2011; Brookes et al., 2019) to solve offline optimization problems is to learn a surrogate model from the given training data which can predict the objective function value for unknown inputs and find optimal input (i.e., maximizer) for this surrogate using gradient-based methods. The key implicit assumption behind this approach is that we can learn an accurate surrogate model over the entire input space using supervised learning. However, this is rarely achievable in practice due to the size and sparsity of the offline training data. In most cases, the surrogate model is only reliable within a constrained neighborhood of the offline data (Fannjiang & Listgarten, 2020) and can be highly erroneous outside this neighborhood. Consequently, there will be a discrepancy between the gradient fields of the target objective function and the surrogate model which will misguide the gradient search towards sub-optimal solutions.

This raises two related questions. *First*, how does the discrepancy in gradient estimation affect the performance gap between the optima of the surrogate model and the target objective function? *Second*, how do we learn surrogate models that closely approximate the gradient field of the target function? Both questions are challenging, given that the target function's gradient field is non-observable even at the offline training data points, and have not been studied by prior work. In fact, we note that while there is an existing literature on random gradient estimation methods (Fu, 2015; Wang et al., 2018), those methods require the ability to actively sample data from the black-box target function which is not possible in the context of offline optimization.

Contributions. The main contributions of this paper include (1) theoretically-sound answers to the above two questions;

[1]Lewis-Sigler Institute of Integrative Genomics, Princeton University, New Jersey, USA [2]School of Electrical Engineering and Computer Science, Washington State University, Pullman, Washington, USA. Correspondence to: Trong Nghia Hoang <trongnghia.hoang@wsu.edu>, Janardhan Rao Doppa <jana.doppa@wsu.edu>.

Proceedings of the 41st International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024.

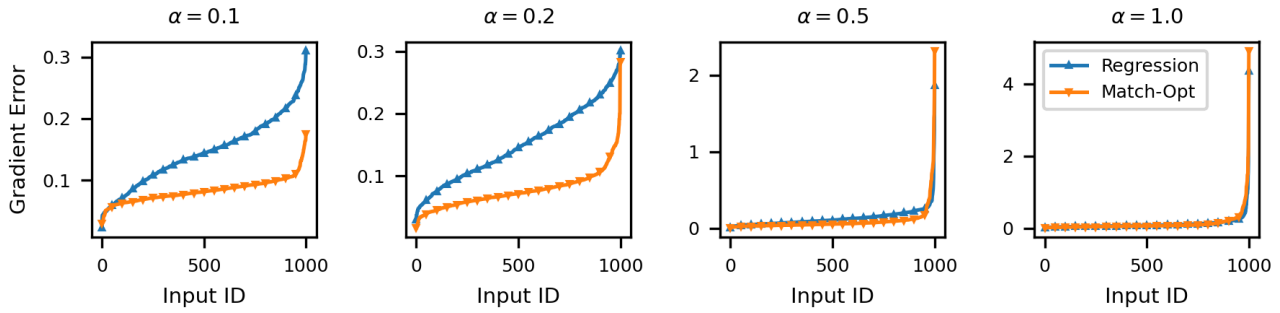


Figure 1. Comparison of gradient estimation error incurred by MATCH-OPT (orange) and standard regression (blue) while learning the gradient field of the Shekel function on 4-dimensional input space at different out-of-distribution (OOD) settings where test inputs were drawn from $\mathbb{N}(0, \alpha\mathbf{I})$ while training inputs were drawn from $\mathbb{N}(0, \mathbf{I})$. Smaller α indicates larger deviation from the offline data regime, which widens the performance gap between MATCH-OPT and standard regression.

and (2) practical demonstration of their significance on real-world offline design optimization problems:

1. To answer the first question, we present a theoretical framework that characterizes the offline optimization performance of gradient-based search guided by a surrogate model. We provably bound the performance gap between the optima of the target function and trained surrogate as a function of how well the surrogate matches the (latent) gradient field of the target function on the offline training data. Our derived bound is non-trivial and yet model-agnostic, making it broadly applicable (Section 3).

2. To answer the second question, we present a principled gradient matching algorithm, referred to as MATCH-OPT, that is inspired by our theoretical analysis. Intuitively, our analysis shows that the worst-case performance of an optimizer following the surrogate gradient is bounded with the gradient gap between the surrogate and target function, and that the bound is tight up to a constant with a sufficiently small learning rate. Hence, a surrogate model trained to directly match gradients will result in good offline optimization performance with gradient search from diverse starting points (referred to as "reliable"). An overview of our algorithm is given in Fig. 2. Our algorithm MATCH-OPT is model-agnostic and allows us to approximate the gradient field that underlies the offline training data using a parametric surrogate (Section 4). In practice, existing offline optimizers exhibit high variance in their performance across diverse optimization tasks. MATCH-OPT is aimed at achieving reliable performance to address this challenge.

To provide an intuition and sanity check to readers, we visualize the reliability of our method's gradient estimation in several out-of-distribution (OOD) settings. We train our method, MATCH-OPT, and a standard regression model on the same set of random inputs drawn from $\mathbb{N}(0, \mathbf{I})$ and their Shekel function (Molga & Smutnicki, 2005) evaluations. Fig. 1 plots the (sorted) gradient estimation error (i.e., the norm difference between predicted and oracle gradients) achieved by the two approaches at 1000 random inputs drawn from different OOD distributions $\mathbb{N}(0, \alpha\mathbf{I})$ parameterized with different values of $\alpha \in [0.1, 0.2, 0.5, 1.0]$. It is observed that (1) when the test and train distributions are the same ($\alpha = 1.0$), the performance of the two approaches are the same; but (2) when α decreases (i.e., larger deviation from the offline data regime), our approach achieves significantly smaller error, suggesting that a direct gradient matching is more reliable in OOD data regimes. While this behaviour does not necessarily translate into better predictive accuracy, our Theorem 3.2 demonstrates that it will indeed minimize the optimization risk as we follow the surrogate's gradient towards the goal of finding the maximum of the target objective function. We note that similar ideas have shown great empirical success in structured prediction where models were learned to guide greedy search in combinatorial spaces (Doppa et al., 2014).

3. Finally, we demonstrate the efficacy of MATCH-OPT on diverse real-world optimization problems from the design-bench benchmark (Trabucco et al., 2022). Our results show that MATCH-OPT consistently shows improved optimization performance over existing baselines, and produces high-quality solutions with gradient search from diverse starting points (Section 5). Our code is publicly available at https://github.com/azzafadhel/MatchOpt.

2. Background and Problem Setup

Offline Black-box Optimization. Suppose $\mathfrak{X}$ is an input space where each $\mathbf{x} \in \mathfrak{X}$ is a candidate input. Let $g : \mathfrak{X} \mapsto \Re$ be an unknown, expensive real-valued objective function which can evaluate any given input $\mathbf{x} \in \mathfrak{X}$ to produce output $z = g(\mathbf{x})$. For example, in material design application, $g(\mathbf{x})$ corresponds to running a physical lab experiment. Our overall goal is to find an optimal input or design $\mathbf{x}_* \in \mathfrak{X}$ that maximizes the output of an experimental process $g(\mathbf{x})$,

$$\mathbf{x}_* \triangleq \operatorname*{argmax}_{\mathbf{x} \in \mathfrak{X}} g(\mathbf{x}) \,. \tag{1}$$

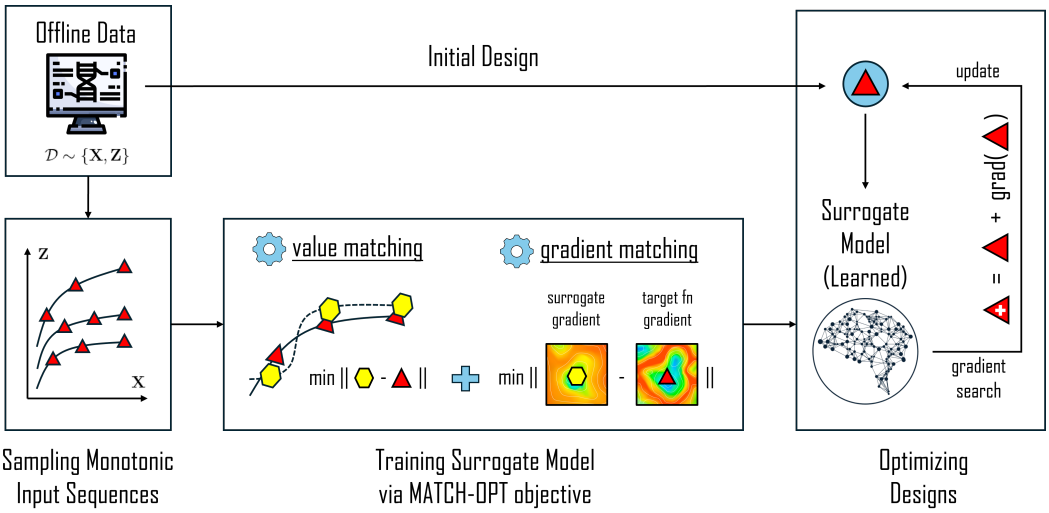


Figure 2. Our approach MATCH-OPT synthesizes input sequences with monotonically increasing target function values from the offline dataset, which are used to train a parametric surrogate model. Our loss function incorporates both standard regression loss (i.e., value matching) and a novel gradient matching loss. We perform gradient search on the trained surrogate to find optimized designs.

We are provided with a dataset of n input-output pairs $\mathfrak{D}=\{(\mathbf{x}_1, z_1), (\mathbf{x}_2, z_2), \cdots, (\mathbf{x}_n, z_n)\}$ collected offline, where $z_i = g(\mathbf{x}_i)$. We do not have access to the target objective function g values on inputs outside the dataset $\mathfrak{D}$.

Surrogate Model. We do not have access to the black-box target function $g(\mathbf{x})$ beyond the offline dataset $\mathfrak{D}$ of n training examples. This allows us to learn a surrogate $g_\phi(\mathbf{x})$ for $g(\mathbf{x})$ via supervised learning.

$$\phi \triangleq \operatorname*{argmin}_{\phi'} \sum_{i=1}^{n} \ell\left(g_{\phi'}\left(\mathbf{x}_i\right),\ z_i\right), \tag{2}$$

where ϕ denotes the parameters of surrogate model and $\ell(z, z')$ denotes the loss of predicting z when the true objective value is z' for a given input $\mathbf{x}$. For example, $\ell(z, z') = (z - z')^2$ and $g_\phi(\mathbf{x}) = \phi^\top \mathbf{x}$.

Gradient-based Search Procedure. Once learned, ϕ is fixed and we can use $g_\phi(\mathbf{x})$ as a surrogate to approximate the optimal design as:

$$\mathbf{x}_\phi \simeq \mathbf{x}_\phi^m \text{ where } \mathbf{x}_\phi^{k+1} \triangleq \mathbf{x}_\phi^k + \lambda \cdot \nabla g_\phi\left(\mathbf{x}_\phi^k\right) \tag{3}$$

which is defined recursively for $0 \leq k \leq m-1$ via a m-step gradient ascent process starting from an initial solution $\mathbf{x}_\phi^0 = \mathbf{x}^0$ with a fixed learning rate $\lambda > 0$. The final iterate $\mathbf{x}_\phi^m$ is referred to as the solution of gradient search guided by the surrogate. The use of a differentiable surrogate to find the optimal design also imposes the following implicit assumption on the unknown black-box target function.

Assumption 2.1. The target function is either differentiable or sufficiently close to a differentiable (black-box) proxy.

This is not an unreasonable assumption because otherwise, offline optimization and more broadly, machine learning is an ill-posed problem following a simple thought experiment: *Suppose the target function is not sufficiently close to any differentiable proxy functions, no algorithm that uses a differentiable surrogate can learn a good approximation of the target function.*

To see this, suppose there exists an algorithm that uses a differentiable surrogate that can approximate well the target function. This means there exists a differentiable function that is sufficiently close to the target function, which contradicts the premise of the experiment. Thus, we argue that the offline optimization task is only meaningful within the set of target functions that can be characterized sufficiently accurately using a (black-box) differentiable proxy.

As such, we would want the above surrogate-guided solution x_ϕ^m to match the solution of gradient search guided by the target function's derivative, or the derivative of a differentiable proxy function that is closest to the target function g. Henceforth, we will refer to this as the target function's gradient, and the solution guided by the gradient of the target function is defined as:

$$\mathbf{x}_* \simeq \mathbf{x}_*^m \text{ where } \mathbf{x}_*^{k+1} \triangleq \mathbf{x}_*^k + \lambda \cdot \nabla g\left(\mathbf{x}_*^k\right) \tag{4}$$

which forms a similar gradient search of m steps with the same initial solution $\mathbf{x}_*^0 = \mathbf{x}_\phi^0 = \mathbf{x}^0$ and learning rate $\lambda > 0$. As such, a discrepancy between target function gradients and surrogate gradients can result in a performance gap between the objective function values of $\mathbf{x}_*^m$ and $\mathbf{x}_\phi^m$. This paper therefore studies two related questions in the context of surrogate-guided gradient search for offline optimization.

Q1. How does the discrepancy between target function and surrogate gradients impact the quality of uncovered solutions? This will be discussed in Section 3.

Q2. How to learn surrogate models that can closely approxi-

mate target function gradients using the offline training data $\mathfrak{D}$? This will be discussed in Section 4.

3. Theoretical Analysis

This section provides a rigorous theoretical analysis to answer **Q1**. Explicitly, we derive an upper-bound for the performance gap between gradient search guided by the target function and the trained surrogate, which is characterized explicitly in terms of how well the surrogate's gradient field fits with the offline data.

Performance Gap. First, we define the performance of the solution found via m steps of gradient ascent on $g_\phi(\mathbf{x})$ starting from $\mathbf{x}_\phi^0 = \mathbf{x}^0$ via

$$\begin{aligned} \mathfrak{R}_{g_\phi}^m\left(\mathbf{x}_\phi^0\right) &= \mathfrak{R}_{g_\phi}^m\left(\mathbf{x}^0\right) &= g(\mathbf{x}_*) - g(\mathbf{x}_\phi) \\ &&= g(\mathbf{x}_*) - g(\mathbf{x}_\phi^m)\,. \end{aligned} \quad (5)$$

where $\mathbf{x}_\phi^m$ is defined in (3). Similarly, we have $\mathfrak{R}_g^m(\mathbf{x}_*^0) = \mathfrak{R}_g^m(\mathbf{x}^0) = g(\mathbf{x}_*) - g(\mathbf{x}_*^m) \geq 0$. Note that we are distinguishing between the solution $\mathbf{x}_*^m$ and $\mathbf{x}_*$ here because, often, finding $\mathbf{x}_*$ is intractable even with access to the target function $g(\mathbf{x})$ (e.g., combinatorial spaces). Thus, it is more practical to compare the surrogate solution with the solution found via following the target function's gradient, rather than the true optima. We can now define the performance gap and state our main result.

Definition 3.1. For a fixed gradient ascent process starting from $\mathbf{x}$ with m update steps and learning rate $\lambda > 0$, the performance gap between the surrogate solution $\mathbf{x}_\phi^m$ and the target function solution $\mathbf{x}_*^m$, $\mathfrak{G}_{m,\lambda}(\mathbf{x})$, is given by:

$$\mathfrak{G}_{m,\lambda}(\mathbf{x}) \triangleq \left\| \mathfrak{R}_g^m(\mathbf{x}) - \mathfrak{R}_{g_\phi}^m(\mathbf{x}) \right\|, \quad (6)$$

where $\mathfrak{R}_g$ and $\mathfrak{R}_{g_\phi}$ are as defined above.

Theorem 3.2 (Worst-case optimization risk bound in terms of gradient estimation error)**.** *Suppose $g(\mathbf{x})$ is a ℓ-Lipschitz continuous and μ-Lipschitz smooth function. The worst-case performance gap, $\mathfrak{G}_{m,\lambda} \triangleq \max_{\mathbf{x}} \mathfrak{G}_{m,\lambda}(\mathbf{x})$, between g and some arbitrary surrogate g_ϕ is upper-bounded by:*

$$\begin{aligned} \mathfrak{G}_{m,\lambda} &\leq m\lambda\ell\Big(1+\lambda\mu\Big)^{m-1} \\ &\times \max_{\mathbf{x}} \left\| \nabla g(\mathbf{x}) - \nabla g_\phi(\mathbf{x}) \right\|. \end{aligned} \quad (7)$$

Note that despite the exponential dependence on m, the bound becomes tight and independent of m when the learning rate $\lambda \leq 1/m$, which is the case in all our experiments. See Appendix A for a detailed derivation.

Theorem 3.2 establishes that the worst-case performance gap between the surrogate and target function solutions is upper-bounded by the maximum norm difference between the surrogate and target function gradients over the input space. This provides a direct quantification of optimization quality as a function of gradient discrepancies. In addition, the result of Theorem 3.2 also characterizes a balance between the risk and potential of gradient search in terms of the learning rate and the number of update steps.

As the learning rate λ or the number of search steps m approaches zero, the bound in Theorem 3.2 also approaches zero. This means an extremely conservative gradient search (one that barely moves) would minimize the gap between $\mathfrak{R}_{g_\phi}$ and $\mathfrak{R}_g$, making the performance of the surrogate solution arbitrarily close to that of the target function solution. However, such a conservative strategy would also widen the gap between the target function solution and the true optima, and thus will ultimately deteriorate the overall performance of offline optimization. Conversely, an explorative search that uses larger λ and m would bring $\mathfrak{R}_g$ closer to zero, making the target function solution arbitrarily close to the true optima. Simultaneously, it also widens the gap between the surrogate and target function solution, again reducing the performance of offline optimization. Furthermore, as the bound in Eq. (7) holds for all possible choices of g_ϕ, we can tighten it with respect to g_ϕ. That is:

$$\begin{aligned} \mathfrak{G}_{m,\lambda} &\leq m\lambda\ell\Big(1+\lambda\mu\Big)^{m-1} \\ &\times \min_{\phi}\max_{\mathbf{x}} \left\| \nabla g(\mathbf{x}) - \nabla g_\phi(\mathbf{x}) \right\|. \end{aligned} \quad (8)$$

For a fixed gradient based search configuration (m, λ), the offline optimization task is therefore reduced to solving a minimax program,

$$\phi_* = \operatorname*{argmin}_{\phi}\max_{\mathbf{x}} \left\| \nabla g(\mathbf{x}) - \nabla g_\phi(\mathbf{x}) \right\|, \quad (9)$$

which is non-trivial since we do not have direct access to $\nabla g(\mathbf{x})$. Instead, we only have the value of $g(\mathbf{x}_i)$ at a finite number of inputs $\{\mathbf{x}_i\}_{i=1}^n$. Fortunately, this can be circumvented via matching the gradient $\nabla g_\phi(\mathbf{x})$ to that of the observational data, which approximately represents the target function. This is detailed in Section 4 below.

Remark. As mentioned in the statement of the theorem, the exponential dependence on m of the above bound can be mitigated by choosing $\lambda \leq 1/m$. To see this, note that

$$(1+\lambda\cdot\mu)^{m-1} \leq \left(1+\frac{\mu}{m}\right)^{m-1} < \left(1+\frac{\mu}{m}\right)^{m} \quad (10)$$

which will approach e^μ in the limit of m. Here, we use the known fact that $\lim_{m\to\infty}(1+\mu/m)^m = e^\mu$ with $\mu > 0$. As such, when m is sufficiently large, the bound in Theorem 3.2 is upper-bounded with $m\cdot\lambda\cdot\ell\cdot(1+\lambda\cdot\mu)^{m-1}\cdot$ *gradient-gap* $\simeq \ell\cdot e^\mu\cdot$ *gradient-gap* which asserts that the worst-case performance gap of our offline optimizer is approaching (in the limit of m) $\ell\cdot e^\mu\cdot\max_{\mathbf{x}}\|\nabla g(\mathbf{x}) - \nabla g_\phi(\mathbf{x})\| = \mathbf{O}(\max_{\mathbf{x}}\|\nabla g(\mathbf{x}) - \nabla g_\phi(\mathbf{x})\|)$ which is not dependent on the no. of gradient steps.

Algorithm 1 MATCH-OPT: Black-Box Gradient Matching from Offline Training Data

Input: Dataset $\mathfrak{D} = \{(\mathbf{x}_i, z_i)\}_{i=1}^n$, initial surrogate model parameters ϕ, length of monotonic synthetic paths m, number of iterations τ, learning rate $\lambda > 0$
Output: Surrogate g_ϕ with parameters $\phi^{(\tau)}$

Generate monotonic trajectories $\mathcal{C}^m$ via strategy from Krishnamoorthy et al. (2023b), Kumar et al. (2019)
$\phi^{(0)} \leftarrow \phi$ // initialize parameters of surrogate model
for $t \leftarrow 0 : \tau - 1$ **do**
  $\mathfrak{L} \leftarrow 0$ // initialize the average loss
  for $\zeta = (\mathbf{x}_1, \ldots, \mathbf{x}_m) \in \mathcal{C}^m$ **do**
    $\mathfrak{L}_g^\zeta \leftarrow$ gradient matching loss using Eq. 13 with $\phi = \phi^{(t)}$
    $\mathfrak{L}_r^\zeta \leftarrow \alpha \cdot \sum_{i=1}^{m} \Big(g(\mathbf{x}_i) - g_\phi(\mathbf{x}_i) \Big)^2$ // using $\phi = \phi^{(t)}$
    $\mathfrak{L} \leftarrow \mathfrak{L} + \Big|\mathcal{C}^m\Big|^{-1} \Big(\mathfrak{L}_g^\zeta + \mathfrak{L}_r^\zeta \Big)$ // update average loss
  end for
  $\phi^{(t+1)} \leftarrow \phi^{(t)} + \lambda \cdot \nabla_\phi \mathfrak{L} \Big|_{\phi = \phi^{(t)}}$
end for
return the learned surrogate model g_ϕ with $\phi = \phi^{(\tau)}$

4. Practical Algorithm: MATCH-OPT

This section answers **Q2** by providing a principled algorithm for black-box gradient matching, which we name MATCH-OPT. The crux of solving Eq. (9) lies with how we approximate the target function's gradient field when we are given evaluations of the target function at a fixed set of inputs (i.e., offline dataset). Previous approaches often address this by sampling perturbed values around a chosen input and use the finite difference method to approximate its gradient (Fu, 2015; Wang et al., 2018). However, these methods require the ability to query the target function for evaluations of perturbed data points, which is not possible in the offline optimization setting. To overcome this challenge, we leverage the fundamental theorem for line integrals, which states that for any two inputs $\mathbf{x}$ and $\mathbf{x}'$ with corresponding values $z = g(\mathbf{x})$ and $z' = g(\mathbf{x}')$:

$$\begin{aligned} \Delta z \triangleq z - z' &= \Big(\mathbf{x}' - \mathbf{x}\Big)^\top \int_0^1 \Big[\nabla g\Big(h(t)\Big)\Big] \mathrm{d}t \\ &\simeq \Big(\mathbf{x}' - \mathbf{x}\Big)^\top \int_0^1 \Big[\nabla g_\phi\Big(h(t)\Big)\Big] \mathrm{d}t \,, \quad (11) \end{aligned}$$

where $h(t) = \mathbf{x} \cdot (1-t) + \mathbf{x}' \cdot t$. The above approximation holds when ∇g_ϕ closely estimates ∇g. To enforce this, we therefore need to find ϕ such that the averaged difference between the LHS and RHS of (11) is minimized. This is achieved by solving $\phi^* = \operatorname{argmin}_\phi \mathfrak{L}_g(\phi)$, where

$$\mathfrak{L}_g(\phi) \triangleq \mathbb{E}_{\mathbf{x},\mathbf{x}' \in \mathfrak{D}} \left(\Delta z - \Delta \mathbf{x}^\top \int_0^1 \nabla g_\phi\Big(h(t)\Big) \mathrm{d}t \right)^2 . \quad (12)$$

Eq. (12) provides a tractable learning objective via taking the empirical expectation over random inputs sampled from the offline training dataset $\mathfrak{D}$. It can also be shown that minimizing Eq. (12) will indeed decrease the gradient gap between the surrogate and the black-box target function (see Appendix B for a detailed derivation). Thus, by virtue of Theorem 3.2, minimizing Eq. (12) has the effect of decreasing the upper-bound on the worst-case performance of offline optimization using the learned surrogate's gradient.

Furthermore, we note that in the ideal scenario, Eq. (12) can be indirectly solved using a regression approach (i.e., value matching) because the gradient fields of g and g_ϕ must be the same when $g_\phi(\mathbf{x})$ accurately estimates $g(\mathbf{x})$ for every input $\mathbf{x}$. However, as long as there are discrepancies, it is unclear which surrogate gradient (among surrogate candidates that approximate the target function equally well) would minimize the gradient discrepancy. As such, we argue that a direct gradient matching approach is more preferable in this case. This statement is supported by both our synthetic experiment (see Fig. 1) and real-world experiments presented in Section 5.3.

Practical Considerations. A naïve optimization of Eq. (12) requires enumerating over all pairs of training inputs, which is more expensive than a standard regression algorithm. To avoid this overhead, we adopt the strategy of Krishnamoorthy et al. (2023b) and Kumar & Levine (2020), which organizes training data into trajectories of monotonically increasing target function values. These trajectories mimic realistic optimization paths and thus encourages the model to learn the behavior of a gradient-based optimization algorithm, and thus allows the gradient matching algorithm to focus more on strategic input pairs that are more relevant for gradient estimation.

Specifically, let $\mathcal{C}^m$ denote a finite set of m-hop synthetic input paths with increasing objective function values, such that if $\zeta = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_m\} \in \mathcal{C}^m$, we have $g(\mathbf{x}_{i+1}) \geq g(\mathbf{x}_i)$. To sample trajectories from this set, we first bin the offline inputs based on their percentiles in the dataset, and subsequently sample one input from each bin to form a trajectory with monotonically increasing function values. We adapt the loss function in Eq. (12) to optimize along the sampled paths, and thus focus on estimating gradient information that is relevant to the downstream search procedure. That is, we aim to minimize $\mathfrak{L}_g(\phi; \mathcal{C}^m) \triangleq \mathbb{E}_{\zeta \in \mathcal{C}^m}[\mathfrak{L}_g(\phi; \zeta)]$, where:

$$\begin{aligned} \mathfrak{L}_g(\phi; \zeta) &\triangleq \sum_{i=1}^{m-1} \left(\Delta z - \Delta \mathbf{x}^\top \int_0^1 \nabla g_\phi\Big(h_i(t)\Big) \mathrm{d}t \right)^2 \\ &\simeq \sum_{i=1}^{m-1} \left(\Delta z - \frac{1}{2\kappa} \sum_{u=1}^{\kappa} \Big(\Delta \mathbf{x}^\top \Big(\mathbf{r}_i(u) \Big) \Big) \right)^2 \quad (13) \end{aligned}$$

with $\mathbf{r}_i(u) = \nabla g_\phi(h_i((u-1)/\kappa)) + \nabla g_\phi(h_i(u/\kappa))$ and $h_i(t) = \mathbf{x}_i \cdot (1-t) + \mathbf{x}_{i+1} \cdot t$. Here, Eq. (13) takes empirical expectation over the successive pairs along the synthesized trajectories $\zeta \in \mathcal{C}^m$, $\Delta z \triangleq g(\mathbf{x}_{i+1}) - g(\mathbf{x}_i)$, and

$\Delta\mathbf{x} \triangleq \mathbf{x}_{i+1} - \mathbf{x}_i$. In addition, the integral inside the expectation on the RHS of Eq. (13) is approximated via a discretization of $(0, 1)$ into κ intervals with equal lengths. Our empirical investigations suggest that setting $\kappa = 5$ works best in practice. Finally, our ultimate loss function $\mathfrak{L}(\phi)$ combines Eq. (13) with the regression loss along the synthetic trajectory to achieve the best of both worlds:

$$\mathfrak{L}(\phi) \triangleq \mathfrak{L}_{g,\mathcal{C}^m}(\phi) + \mathbb{E}_{\zeta\in\mathcal{C}^m}\sum_{i=1}^{m}\Big(g(\mathbf{x}_i) - g_\phi(\mathbf{x}_i)\Big)^2 \quad (14)$$

See Algorithm 1 for a complete pseudo-code of our method. Interestingly, Eq. (14) is also motivated by a modification of the above theoretical analysis in Section 3, which characterizes a condition under which the worst-case optimization risk in Theorem 3.2 has a tighter bound. It can be shown that such a bound will depend on both the (worst-case) gradient- and value-matching quantities, which inspires the addition of the above regression loss in Eq. (14) to the original loss in Eq. (13). This is formalized in Theorem 4.1 below.

Theorem 4.1 (Generalized worst-case optimization risk bound)**.** *Suppose the target objective function $g(\mathbf{x})$ is a ℓ-Lipschitz continuous and μ-Lipschitz smooth function. For all $a \in (0, 1)$, the worst-case performance gap, $\mathfrak{G}_{m,\lambda} \triangleq \max_{\mathbf{x}} \mathfrak{G}_{m,\lambda}(\mathbf{x})$, between g and some arbitrary surrogate g_ϕ with Lipschitz constant ℓ_ϕ is upper-bounded by:*

$$\begin{aligned}\mathfrak{G}_{m,\lambda} \;&\leq\; m\cdot 2a\cdot\max_{\mathbf{x}}\Big\|g(\mathbf{x}) - g_\phi(\mathbf{x})\Big\| \\ &+\; m\cdot\Big(\ell + a\cdot(\ell_\phi - \ell)\Big)\cdot\Big(1+\lambda\mu\Big)^{m-1} \\ &\times\; \max_{\mathbf{x}}\Big\|\nabla_{\mathbf{x}}g(\mathbf{x}) - \nabla_{\mathbf{x}}g_\phi(\mathbf{x})\Big\|, \end{aligned} \quad (15)$$

which is tighter than the bound in Theorem 3.2 when the Lipschitz constant ℓ_ϕ of the surrogate satisfies:

$$\ell_\phi \;\leq\; \lambda\ell - \frac{2\cdot\max\Big\|g(\mathbf{x}) - g_\phi(\mathbf{x})\Big\|}{\Big(1+\lambda\mu\Big)^{m-1}\cdot\max\Big\|\nabla_{\mathbf{x}}g(\mathbf{x}) - \nabla_{\mathbf{x}}g_\phi(\mathbf{x})\Big\|}$$

The detailed proof of this result is deferred to Appendix D.

Although it has not been investigated how to further condition the training loss in Eq. (14) so that ℓ_ϕ satisfies the above, we are able to empirically demonstrate the benefit of adding the regression loss along the synthetic monotonic trajectories via an ablation study in Section 5.3.

Complexity Analysis. Given a m-hop synthetic sequence ζ of d-dimensional inputs, each step of the inner loop in Algorithm 1 will require a linear scan over m segments. For each segment, the algorithm needs to compute (1) the gradient matching loss, which costs $\mathcal{O}(dm\kappa|\phi|)$ where κ is the granularity of the discretization in (13) and $|\phi|$ is the number of parameters of the surrogate model, and (2) the regression regularizer on this path, which costs $\mathcal{O}(m|\phi|)$. Thus, suppose $p = |\mathcal{C}^m|$ synthetic input sequences/paths were generated for our algorithm, the entire inner loop of Algorithm 1 will incur a total cost of $\mathcal{O}(p\cdot(dm\kappa|\phi| + m|\phi|)) = \mathcal{O}(p\cdot dm\kappa|\phi|)$. This is the complexity per training iteration. For τ iterations, the total complexity of Algorithm 1 will be $\mathcal{O}(\tau\cdot p\cdot dm\kappa|\phi|)$.

5. Experiments

This section describes the set of benchmark tasks used to evaluate and compare the performance of `MATCH-OPT` with those of other baselines (Section 5.1), the configurations of both our proposed algorithm and those baselines (Section 5.2), as well as their reported results (Section 5.3).

5.1. Benchmarks

Our empirical studies are conducted on six benchmark tasks from a diverse set engineering domains. Each task comprises a black-box target function and an offline training dataset, which is a small subset of a much larger dataset used to train the target function. Each participating algorithm only has access to the offline dataset. The target function is only used to evaluate the performance of the final inputs recommended by those offline optimizers. The specifics of these datasets and their target functions are further provided in the design baseline package (Trabucco et al., 2022). Four tasks are defined over continuous input spaces, whereas the other two are discrete, as summarized below.

1 & 2. The Ant Morphology (Brockman et al., 2016) (ANT) and D'Kitty Morphology dataset (Ahn et al., 2020) (DKITTY) collect morphological observations of two robots and their corresponding rewards in moving as fast as possible, or towards a specific location. The parameters of the robot is defined over a 60/56-dimensional continuous space.

3. The Hopper Controller dataset (Ahn et al., 2020) (HOPPER) collects observations of a neural network policy weights and their rewards on the Hopper-v2 locomotion task in OpenAI Gym (Brockman et al., 2016). The search space is defined over 5126-dimensional continuous space.

4. The Superconductor dataset (Brookes et al., 2019) (SCON) collects observations of superconductor molecules and their critical temperatures. Each molecule is represented by a 86-dimensional continuous vector.

5 & 6. The TF-Bind-8 (TF8) and TF-Bind-10 (TF10) datasets (Barrera et al., 2016) collect the binding activity scores between a given human transcription factor and various DNA sequences of length 8 and 10. The goal of these *discrete* tasks is to find a DNA sequence that maximizes the binding score with the given transcription factor.

METHOD	ANT	DKITTY	HOPPER	SCON	TF8	TF10	MNR
GA	0.271	0.895	0.780	0.699	0.954	0.966	0.600
ENS-MEAN	0.517	0.899	1.524	0.716	0.926	**0.968**	0.500
ENS-MIN	0.536	0.908	1.420	0.734	0.959	0.959	0.467
CMA-ES	**0.974**	0.722	0.620	**0.757**	**0.978**	0.966	0.367
MINS	0.910	0.939	0.150	0.690	0.900	0.759	0.700
CBAS	0.842	0.879	0.150	0.659	0.916	0.928	0.733
ROMA	0.832	0.880	2.026	0.704	0.664	0.820	0.667
BONET	0.927	0.954	0.395	0.500	0.911	0.756	0.683
COMS	0.885	0.953	**2.270**	0.565	0.968	0.873	0.467
MATCH-OPT	0.931 (2)	**0.957 (1)**	1.572 (3)	0.732 (3)	0.977 (2)	0.924 (6)	**0.283 (1)**

Table 1. Performance of MATCH-OPT and other baselines at 100^{th} percentile level. The last column shows the mean normalized rank (MNR) computed across all tasks (smaller is better). The individual rank of MATCH-OPT on each task is included next to its performance.

METHOD	ANT	DKITTY	HOPPER	SCON	TF8	TF10	MNR
GA	0.130	0.742	0.089	0.641	0.510	0.794	0.600
ENS-MEAN	0.192	0.791	0.209	0.644	0.529	**0.796**	0.433
ENS-MIN	0.190	0.803	0.166	**0.672**	0.490	0.794	0.500
CMA-ES	-0.049	0.482	-0.033	0.590	0.592	0.786	0.683
MINS	0.614	0.889	0.088	0.414	0.420	0.465	0.650
CBAS	0.376	0.757	0.013	0.099	0.442	0.613	0.817
ROMA	0.448	0.760	0.370	0.420	0.560	0.780	0.533
BONET	**0.620**	**0.897**	0.390	0.470	0.505	0.465	0.417
COMS	0.557	0.879	0.379	0.414	**0.652**	0.606	0.467
MATCH-OPT	0.611 (3)	0.887 (3)	**0.393 (1)**	0.439 (6)	0.594 (2)	0.720 (6)	**0.350 (1)**

Table 2. Performance of MATCH-OPT and other baselines at 50^{th} percentile level. The last column shows the mean normalized rank (MNR) computed across all tasks (smaller is better). The individual rank of MATCH-OPT on each task is included next to its performance.

5.2. Algorithm Configuration and Evaluation

Baselines. We evaluate and compare the performance of MATCH-OPT against those of multiple state-of-the-art baselines including COMS (Trabucco et al., 2021), ROMA (Yu et al., 2021), BONET (Krishnamoorthy et al., 2023b). Several other baselines from the design bench benchmark (Trabucco et al., 2022) including Gradient Ascent (GA), Gradient Ascent Ensemble Mean (ENS-MEAN), Gradient Ascent Ensemble Min (ENS-MIN), covariance matrix adaptation evolution strategy (CMA-ES) (Hansen, 2006), model inversion networks (MINS) (Kumar & Levine, 2020), conditioning by adaptive sampling (CBAS) (Brookes et al., 2019) are also included for a thorough comparison. The same neural network architecture is used for all baselines. Our implementation of the MATCH-OPT framework is released at https://github.com/azzafadhel/MatchOpt. Other details of our experiments are deferred to Appendix C.

Evaluation Methodology. We follow the widely adopted evaluation methodology introduced by Trabucco et al. (2022). That is, each algorithm starts the search from the same initial set of $n = 128$ offline inputs and generates the corresponding set of solution candidates which are evaluated by the oracle function. For each algorithm, these (128) solutions are then sorted in increasing order, and the corresponding values at the 100^{th} percentile (maximum solution) and 50^{th} (median solution) are reported in Table 1 and Table 2 below. All target function values are normalized using the maximum and minimum values from a larger unobserved dataset (that was used to train the target function). We run each algorithm on each task four times and report the mean. We report their standard deviations in Appendix E.

Comparison Metrics. The overall performance of a baseline against other methods across different optimization tasks can be assessed using (a) their mean (normalized) performance; and (b) their mean (normalized) performance rank. While the first metric has often been used in prior work, it does not account for the variation in performance among tasks. For example, normalized performance are often close to 1 for easy tasks, whereas for harder tasks, they can be closer to 0. The mean performance metric therefore might favor algorithms that do well on easy tasks, but poorly on other hard tasks. To mitigate such biased assessment, we

consider the mean normalized rank (MNR) metric that is agnostic to such variation of performance:

$$\text{MNR}(\mathcal{A}) \triangleq \frac{1}{p}\sum_{i=1}^{p}\frac{\text{rank}(\mathcal{A};\text{task}_i)}{\#\text{ algorithms}} \tag{16}$$

where p is the number of tasks and $\text{rank}(\mathcal{A};\text{task}_i) = q$ means $\mathcal{A}$ is the q-best algorithm for the i-th task. To scale the MNR to the same range of $(0, 1)$ (for convenience), we also normalize the rank by the number of participating algorithms in the ranking order. An algorithm with low MNR therefore has more reliable performance across tasks, and is preferable to other methods with higher MNR.

5.3. Results and Discussion

To demonstrate the effectiveness of MATCH-OPT, we report the 100^{th} and 50^{th} percentile results of all methods in Table 1 and Table 2. Other than the algorithm's individual performance reported for each task, we calculate its mean normalized rank (see Eq. (16)) to account for the reliability of its performance (across tasks) in the comparison.

Mean Rank Comparison. Overall, no algorithm performs best in more than two task domains due to the diverse and challenging nature of the benchmark tasks. In fact, for the 100-percentile performance reported in Table 1, each algorithm only performs best in at most one task. Among these, MATCH-OPT performs best on the DKITTY dataset, and second best on ANT and TF8 datasets. MATCH-OPT is consistently among the top-3 performers on four out of six task domains, which is an evidence of its reliable performance. In fact, this is best reflected in terms of the mean normalized rank metric (MNR) which averages the normalized rank of each baseline across all six tasks (see Eq. (16)). Among all algorithms, MATCH-OPT achieves the lowest MNR, which is markedly lower than that of the second lowest MNR of COMS. At 50^{th} percentile, Table 2 shows that MATCH-OPT achieves the best MNR among the baselines.

Reliability Assessment. To further demonstrate the consistent reliability of MATCH-OPT as previously alluded to in the introduction section, we also plot the MNRs of all competing baselines at every solution percentile level in Fig. 3a. As expected, MATCH-OPT achieves the lowest MNR at almost every percentile, averaging at approximately 0.35 which is again markedly lower than the second lowest MNR. In addition, we also plot the mean performance of the tested algorithms across all percentile level in Fig. 3b, which also show that MATCH-OPT is the best performer (on average) between 0- and 80-percentile. Above that, between 80- and 100-percentile level, MATCH-OPT is the second best performer. The above observations (both MNR and mean performance) suggest that MATCH-OPT is consistently the most reliable among all optimizers. We also refer the readers to Appendix F which further visualizes the entire rank distribution of the tested algorithm across different percentile level. All observations are consistent with our above observations in Fig. 3.

Ablation Studies for Regression Regularizer. To demonstrate the effectiveness of our practical consideration mentioned in Section 4, we conduct an ablation study comparing two versions of MATCH-OPT using the original gradient matching loss in (12) (referred as MATCH-OPT (no-regularizer)) and an augmented version with regression regularizer along a set of sampled synthetic input sequences in (14) (referred to as MATCH-OPT (with-regularizer)). Table 3 and 4 below reports the performance of these ablated methods at the 100^{th} and 50^{th} percentile of solutions respectively. Overall, we observe that MATCH-OPT (with-regularizer) outperforms MATCH-OPT (no-regularizer) on 4/6 tasks for both the 100^{th}-percentile and 50^{th}-percentile metric, thus confirming that it is an effective strategy to prioritize optimizing the gradient matching loss along critical trajectories of inputs.

6. Related Work

Black-box optimization problems were studied using derivative-free methods, such as random gradient estimation (Wang et al., 2018) or Bayesian optimization (Snoek et al., 2012; Wang et al., 2013; Eriksson et al., 2019). These methods require online evaluation of the target function to approximate its derivative or learn its surrogate model. In many practical applications, this can be very expensive (e.g., testing new protein or drug design), or even dangerous (e.g., test-driving autonomous vehicles in a real physical environment). To avoid this, *offline optimization* approaches tackle this problem via utilizing an existing dataset that records target function evaluations for a fixed set of inputs. These approaches can be categorized into two main families:

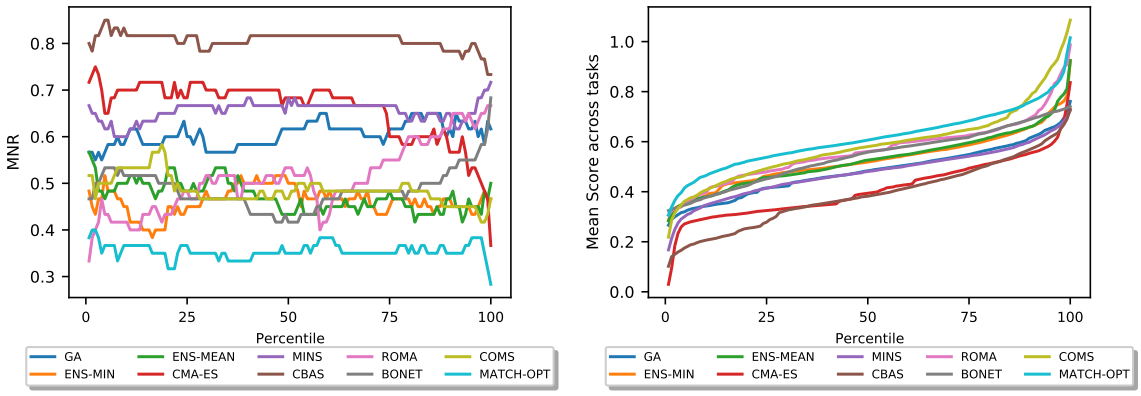
Conditioning Search Model. Existing approaches in this direction are grounded in the framework of density estimation, which aims to learn a probabilistic prior over the input space. The search model is treated as a probability distribution conditioned on the rare event of achieving a high target function score, and is estimated using different approaches, such as adaptive trust-region based strategies (Brookes et al., 2019), adaptive step-size in gradient update via reinforcement learning (Chemingui et al., 2024) or zero-sum game (Fannjiang & Listgarten, 2020), or autoregressive modeling (Krishnamoorthy et al., 2023b), (Krishnamoorthy et al., 2023a). Kumar & Levine (2020) learns an inverse mapping of the target function evaluations to inputs and uses it as a search model that predicts which regions will most likely have high-performing designs. These approaches are often sensitive to the accuracy of the conditioning at out-of-distribution input regimes and/or require learning a computationally expensive generative model of the input space. The robustness of

METHOD	ANT	DKITTY	HOPPER	SCON	TF8	TF10
MATCH-OPT (no-regularizer)	0.924	0.945	1.172	**0.739**	0.941	**0.954**
MATCH-OPT (with-regularizer)	**0.931**	**0.957**	**1.572**	0.732	**0.977**	0.924

Table 3. Performance comparison between versions of MATCH-OPT with and without regression regularizer at the 100^{th} performance percentile (i.e., maximum solution) generated by each method.

METHOD	ANT	DKITTY	HOPPER	SCON	TF8	TF10
MATCH-OPT (no-regularizer)	0.572	0.876	0.372	**0.471**	0.551	**0.768**
MATCH-OPT (with-regularizer)	**0.611**	**0.887**	**0.393**	0.439	**0.594**	0.720

Table 4. Performance comparison between versions of MATCH-OPT with and without regression regularizer at the 50^{th} performance percentile (i.e., maximum solution) generated by each method.



(a) Mean Normalized Rank (b) Mean Performance

Figure 3. Plots of (a) mean normalized ranks (MNRs); and (b) mean (normalized) performance of baselines at all performance percentiles.

these conditioning algorithms has neither been defined, nor investigated.

Conditioning Surrogate Model. Approaches in this direction tend to fix the search methodology and focus on conditioning the surrogate model to improve the likelihood of finding a good design. This is generally achieved via adopting different forms of regularization on the predicted values of OOD inputs based on the learned surrogate. For example, Yu et al. (2021) uses robust model pre-training and adaptation to ensure local smoothness, whereas Fu & Levine (2021) maximizes data likelihood to reduce the uncertainty in OOD prediction. Alternatively, Trabucco et al. (2021) penalizes high-value predictions for OOD examples, and Dao et al. (2024) penalizes surrogate candidates with high prediction sensitivity over the offline data to avoid overestimation. These approaches are only justified empirically through practical demonstrations. From a theoretical perspective, the extent of effectiveness of these algorithms, as well as the fundamental question regarding when to trust a surrogate function both remain unclear.

7. Conclusion

This paper presents a new theoretical perspective on offline black-box optimization which established the first upper bound on the performance gap between the solutions guided by a trained surrogate and the target function. The bound reveals that such performance gap depends on how well the surrogate model matches the gradient field of the target function on the offline dataset. Inspired by this theory, we studied a novel algorithm for creating surrogate models based on gradient matching and demonstrated improved solutions on diverse real-world benchmarks. Although our theory and algorithm is grounded in the context of offline optimization, the developed principles can be broadly applied to related sub-areas including safe Bayesian optimization and safe reinforcement learning in online learning scenarios.

Impact Statement

This paper introduces a new theoretical perspective to understand and analyze the offline optimization problem, which is a cost-effective alternative to the traditional online experimentation approach to material or experimental design. The methodological improvements and new understanding gained in the paper can lead to improvements in many science and engineering applications including design optimization of hardware, materials, and molecules. Our empirical studies only use publicly available dataset. We do not anticipate any negative ethical or societal impact.

References

Ahn, M., Zhu, H., Hartikainen, K., Ponte, H., Gupta, A., Levine, S., and Kumar, V. Robel: Robotics benchmarks for learning with low-cost robots. In *Conference on robot learning*, pp. 1300–1313. PMLR, 2020.

Barrera, L. A., Vedenko, A., Kurland, J. V., Rogers, J. M., Gisselbrecht, S. S., Rossin, E. J., Woodard, J., Mariani, L., Kock, K. H., Inukai, S., et al. Survey of variation in human transcription factors reveals prevalent dna binding changes. *Science*, 351(6280):1450–1454, 2016.

Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., and Zaremba, W. Openai gym. *arXiv preprint arXiv:1606.01540*, 2016.

Brookes, D., Park, H., and Listgarten, J. Conditioning by adaptive sampling for robust design. In *International conference on machine learning*, pp. 773–782. PMLR, 2019.

Chemingui, Y., Deshwal, A., Hoang, T. N., and Doppa, J. R. Offline model-based optimization via policy-guided gradient search. In *AAAI Conference on Artificial Intelligence*, 2024.

Chen, C., Zhang, Y., Fu, J., Liu, X., and Coates, M. Bidirectional learning for offline infinite-width model-based optimization. In *Thirty-Sixth Conference on Neural Information Processing Systems*, 2022. URL https://openreview.net/forum?id=_j8yVIyp27Q.

Dao, M. C., Nguyen, P. L., Truong, T. N., and Hoang, T. N. Boosting offline optimizers with surrogate sensitivity. In *ICML*, 2024.

Doppa, J. R., Fern, A., and Tadepalli, P. Structured prediction via output space search. *Journal of Machine Learning Research*, 15(38):1317–1350, 2014. URL http://jmlr.org/papers/v15/doppa14a.html.

Eriksson, D., Pearce, M., Gardner, J., Turner, R. D., and Poloczek, M. Scalable global optimization via local bayesian optimization. *Advances in Neural Information Processing Systems*, 32, 2019.

Fannjiang, C. and Listgarten, J. Autofocused oracles for model-based design. *Advances in Neural Information Processing Systems*, 33:12945–12956, 2020.

Fu, J. and Levine, S. Offline model-based optimization via normalized maximum likelihood estimation. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=FmMKSO4e8JK.

Fu, M. C. *Stochastic gradient estimation*. Springer, 2015.

Hansen, N. The CMA evolution strategy: a comparing review. *Towards a new evolutionary computation: Advances in the estimation of distribution algorithms*, pp. 75–102, 2006.

Hutter, F., Hoos, H. H., and Leyton-Brown, K. Sequential model-based optimization for general algorithm configuration. In *Learning and Intelligent Optimization: 5th International Conference, LION*, pp. 507–523. Springer, 2011.

Krishnamoorthy, S., Mashkaria, S. M., and Grover, A. Diffusion models for black-box optimization, 2023a.

Krishnamoorthy, S., Mashkaria, S. M., and Grover, A. Generative pretraining for black-box optimization. In *Internation Conference on Machine Learning*, 2023b.

Kumar, A. and Levine, S. Model inversion networks for model-based optimization. *Advances in Neural Information Processing Systems*, 33:5126–5137, 2020.

Kumar, A., Fu, J., Soh, M., Tucker, G., and Levine, S. Stabilizing off-policy q-learning via bootstrapping error reduction. *Advances in Neural Information Processing Systems*, 32, 2019.

Molga, M. and Smutnicki, C. Test functions for optimization needs. *Test functions for optimization needs*, 101:48, 2005.

Nguyen, A. W. and Daugherty, P. S. Evolutionary optimization of fluorescent proteins for intracellular fret. *Nature Biotechnology*, 23(3):355–360, 2005.

Si, Q., Yu, R., and Abrahams, E. High-temperature superconductivity in iron pnictides and chalcogenides. *Nature Reviews Materials*, 1(4):1–15, 2016.

Snoek, J., Hugo, L., and Adams, R. P. Practical Bayesian optimization of machine learning algorithms. In *Advances in Neural Information Processing Systems*, pp. 2960–2968, 2012.

Trabucco, B., Kumar, A., Geng, X., and Levine, S. Conservative objective models for effective offline model-based optimization. In *International Conference on Machine Learning*, pp. 10358–10368. PMLR, 2021.

Trabucco, B., Geng, X., Kumar, A., and Levine, S. Design-bench: Benchmarks for data-driven offline model-based optimization. In *International Conference on Machine Learning*, pp. 21658–21676. PMLR, 2022.

Wang, Y., Du, S., Balakrishnan, S., and Singh, A. Stochastic zeroth-order optimization in high dimensions. In *International conference on artificial intelligence and statistics*, pp. 1356–1365. PMLR, 2018.

Wang, Z., Zoghi, M., Hutter, F., Matheson, D., and de Freitas, N. Bayesian optimization in high dimensions via random embeddings. In *International Joint Conference on Artificial Intelligence*, pp. 1778–1784, 2013.

Wilson, J. T., Moriconi, R., Hutter, F., and Deisenroth, M. P. The reparameterization trick for acquisition functions, 2017.

Yu, S., Ahn, S., Song, L., and Shin, J. Roma: Robust model adaptation for offline model-based optimization. *Advances in Neural Information Processing Systems*, 34: 4619–4631, 2021.

A. Proof of Theorem 3.2

Theorem A.1 (Worst-case optimization risk in terms of gradient estimation error)**.** *Suppose $g(\mathbf{x})$ is a continuous function with Lipschitz and smooth constants, ℓ and μ. Then, we have*

$$\mathfrak{G}_{m,\lambda} \;\triangleq\; \max_{\mathbf{x}} \mathfrak{G}_{m,\lambda}\Big(\mathbf{x}\Big) \;\leq\; m\lambda\ell\Big(1+\lambda\mu\Big)^{m-1} \cdot \max_{\mathbf{x}} \Big\|\nabla g(\mathbf{x}) - \nabla g_{\phi}(\mathbf{x})\Big\|$$

which characterizes the upper-bound of the worst-case performance gap in terms of the maximum norm difference between the surrogate and oracle gradient over the input space.

Proof. We first note that the performance of the m-step oracle solution starting at $\mathbf{x}_*^0 = \mathbf{x}^0$ is exactly the (m-1)-step oracle solution starting at $\mathbf{x}_*^1$. Likewise, the performance of the m-step surrogate solution starting at $\mathbf{x}_\phi^0 = \mathbf{x}^0$ is exactly the (m-1)-step surrogate solution starting at $\mathbf{x}_\phi^1$. That is:

$$\begin{aligned}
\mathfrak{R}_g^m\left(\mathbf{x}^0\right) \;&=\; \mathfrak{R}_g^m\left(\mathbf{x}_*^0\right) \;=\; \mathfrak{R}_g^{m-1}\left(\mathbf{x}_*^1\right) \quad \text{where} \quad \mathbf{x}_*^1 \;=\; \mathbf{x}_*^0 \;+\; \lambda\nabla g\left(\mathbf{x}_*^0\right) \\
\mathfrak{R}_g^m\left(\mathbf{x}^0\right) \;&=\; \mathfrak{R}_{g_\phi}^m\left(\mathbf{x}_\phi^0\right) \;=\; \mathfrak{R}_{g_\phi}^{m-1}\left(\mathbf{x}_\phi^1\right) \quad \text{where} \quad \mathbf{x}_\phi^1 \;=\; \mathbf{x}_\phi^0 \;+\; \lambda\nabla g_\phi\left(\mathbf{x}_\phi^0\right)
\end{aligned} \tag{17}$$

Consequently, for each initial point $\mathbf{x}^0$, we can bound the performance gap as follows:

$$\begin{aligned}
\mathfrak{G}_{m,\lambda}\Big(\mathbf{x}^0\Big) \;&\triangleq\; \Big\|\mathfrak{R}_{g_\phi}^m\left(\mathbf{x}^0\right) \;-\; \mathfrak{R}_g^m\left(\mathbf{x}^0\right)\Big\| \;=\; \Big\|\mathfrak{R}_{g_\phi}^{m-1}\left(\mathbf{x}_\phi^1\right) - \mathfrak{R}_g^{m-1}\left(\mathbf{x}_*^1\right)\Big\| \\
&=\; \Big\|\mathfrak{R}_{g_\phi}^{m-1}\left(\mathbf{x}_\phi^1\right) \;-\; \mathfrak{R}_g^{m-1}\left(\mathbf{x}_\phi^1\right) \;+\; \mathfrak{R}_g^{m-1}\left(\mathbf{x}_\phi^1\right) \;-\; \mathfrak{R}_g^{m-1}\left(\mathbf{x}_*^1\right)\Big\| \\
&\leq\; \Big\|\mathfrak{R}_{g_\phi}^{m-1}\left(\mathbf{x}_\phi^1\right) \;-\; \mathfrak{R}_g^{m-1}\left(\mathbf{x}_\phi^1\right)\Big\| \;+\; \Big\|\mathfrak{R}_g^{m-1}\left(\mathbf{x}_\phi^1\right) \;-\; \mathfrak{R}_g^{m-1}\left(\mathbf{x}_*^1\right)\Big\| \\
&=\; \mathfrak{G}_{m-1,\lambda}\left(\mathbf{x}_\phi^1\right) \;+\; \Big\|\mathfrak{R}_g^{m-1}\left(\mathbf{x}_\phi^1\right) \;-\; \mathfrak{R}_g^{m-1}\left(\mathbf{x}_*^1\right)\Big\| .
\end{aligned} \tag{18}$$

Thus, let $\mathcal{E}_{m-1}(\mathbf{x}_\phi^1, \mathbf{x}_*^1) \triangleq \|\mathfrak{R}_g^{m-1}(\mathbf{x}_\phi^1) \;-\; \mathfrak{R}_g^{m-1}(\mathbf{x}_*^1)\|$, we have $\mathfrak{G}_{m,\lambda}(\mathbf{x}^0) \leq \mathfrak{G}_{m-1,\lambda}(\mathbf{x}_\phi^1) + \mathcal{E}_{m-1}(\mathbf{x}_\phi^1, \mathbf{x}_*^1)$. To bound the term $\mathcal{E}_{m-1}(\mathbf{x}_\phi^1, \mathbf{x}_*^1)$, we will prove the following intermediate results.

Lemma A.2. *For any $k \in [1, m]$ and two different starting points $\mathbf{u}^0$ and $\mathbf{v}^0$, the performance gap between the k-step oracle solutions respectively starting from $\mathbf{u}^0$ and $\mathbf{v}^0$ is bounded by the norm distance between the starting points:*

$$\mathcal{E}_k\Big(\mathbf{u}^0, \mathbf{v}^0\Big) \;=\; \left\|\mathfrak{R}_g^k\left(\mathbf{u}^0\right) \;-\; \mathfrak{R}_g^k\left(\mathbf{v}^0\right)\right\| \;\leq\; \ell(1+\lambda\mu)^k \left\|\mathbf{u}^0 \;-\; \mathbf{v}^0\right\| . \tag{19}$$

Proof. Let us first define the respective oracle search trajectories using the same gradient ascent formalism. That is, the respective candidate solutions at some intermediate step $\kappa \in [1, k]$ are given by $\mathbf{u}^\kappa = \mathbf{u}^{\kappa-1} + \lambda\nabla g(\mathbf{u}^{\kappa-1})$ and $\mathbf{v}^\kappa = \mathbf{v}^{\kappa-1} + \lambda\nabla g(\mathbf{v}^{\kappa-1})$. We can then make use of the Lipschitz continuous assumption to achieve the following bound:

$$\begin{aligned}
\left\|\mathfrak{R}_g^k\left(\mathbf{u}^0\right) - \mathfrak{R}_g^k\left(\mathbf{v}^0\right)\right\| \;&=\; \left\|g\left(\mathbf{x}_*\right) - g\left(\mathbf{u}^k\right) - g\left(\mathbf{x}_*\right) \;+\; g\left(\mathbf{v}^k\right)\right\| \\
&=\; \left\|g\left(\mathbf{v}^k\right) - g\left(\mathbf{u}^k\right)\right\| \;\leq\; \ell \cdot \left\|\mathbf{v}^k - \mathbf{u}^k\right\| .
\end{aligned} \tag{20}$$

We subsequently bound the distance between the candidate solutions at step k in terms of the distance at step $k-1$ using the smoothness conditions:

$$\begin{aligned}
\left\|\mathbf{v}^k - \mathbf{u}^k\right\| \;&=\; \left\|\mathbf{v}^{k-1} - \mathbf{u}^{k-1} + \lambda\nabla g\left(\mathbf{v}^{k-1}\right) \;-\; \lambda\nabla g\left(\mathbf{u}^{k-1}\right)\right\| \\
&\leq\; \left\|\mathbf{v}^{k-1} - \mathbf{u}^{k-1}\right\| \;+\; \lambda\Big\|\nabla g\left(\mathbf{v}^{k-1}\right) \;-\; \nabla g\left(\mathbf{u}^{k-1}\right)\Big\| \\
&\leq\; (1+\lambda\mu)\left\|\mathbf{v}^{k-1} - \mathbf{u}^{k-1}\right\| .
\end{aligned} \tag{21}$$

Applying this bound recursively yields $\|\mathbf{v}^k - \mathbf{u}^k\| \leq (1+\lambda\mu)^k\|\mathbf{v}^0 - \mathbf{u}^0\|$. We finally substitute the above into (20) to arrive at the final bound $\|\mathfrak{R}_g^k(\mathbf{u}^0) - \mathfrak{R}_g^k(\mathbf{v}^0)\| \leq \ell(1+\lambda\mu)^k\|\mathbf{v}^0 - \mathbf{u}^0\|$. □

Applying Lemma A.2 with $k = m-1$, $\mathbf{u}^0 = \mathbf{x}^1_\phi$, and $\mathbf{v}^0 = \mathbf{x}^1_*$ subsequently allows us to bound $\mathcal{E}_{m-1}(\mathbf{x}^1_\phi, \mathbf{x}^1_*)$ as follows:

$$\begin{aligned}\mathcal{E}_{m-1}\left(\mathbf{x}^1_\phi, \mathbf{x}^1_*\right) &\leq \ell(1+\lambda\mu)^{m-1}\left\|\mathbf{x}^1_\phi - \mathbf{x}^1_*\right\| \\ &= \ell(1+\lambda\mu)^{m-1}\left\|\mathbf{x}^0 + \lambda\nabla g_\phi(\mathbf{x}^0) - \mathbf{x}^0 - \lambda\nabla g(\mathbf{x}^0)\right\| \\ &= \lambda\ell(1+\lambda\mu)^{m-1}\left\|\nabla g_\phi(\mathbf{x}^0) - \nabla g(\mathbf{x}^0)\right\| \triangleq \mathcal{Q}\left(\mathbf{x}^0\right) . \end{aligned} \tag{22}$$

We will now use this result to complete our bound for the performance gap. That is:

$$\begin{aligned}\mathfrak{G}_{m,\lambda} &\triangleq \max_{\mathbf{x}^0}\mathfrak{G}_{m,\lambda}\Big(\mathbf{x}^0\Big) \leq \max_{\mathbf{x}^0}\mathfrak{G}_{m-1,\lambda}\Big(\mathbf{x}^1_\phi\Big) + \max_{\mathbf{x}^0}\mathcal{Q}\Big(\mathbf{x}^0\Big) \\ &\leq \mathfrak{G}_{m-1,\lambda} + \max_{\mathbf{x}^0}\mathcal{Q}\Big(\mathbf{x}^0\Big) \\ &\leq \mathfrak{G}_{0,\lambda} + m\cdot\max_{\mathbf{x}^0}\mathcal{Q}\Big(\mathbf{x}^0\Big) , \end{aligned} \tag{23}$$

where the last inequality is obtained via recursively applying the previous inequality m times. Substituting $\mathfrak{G}_{0,\lambda} = 0$ and the upper-bound for $\mathcal{Q}\Big(\mathbf{x}^0\Big)$ above into (23) gives:

$$\mathfrak{G}_{m,\lambda} \leq m\lambda\ell(1+\lambda\mu)^{m-1}\cdot\max_{\mathbf{x}^0}\left\|\nabla g_\phi\left(\mathbf{x}^0\right) - \nabla g\left(\mathbf{x}^0\right)\right\| , \tag{24}$$

which completes our proof for Theorem 3.2. □

Tightness of the bound. Note that despite the exponential dependence on m of the above bound, its tightness can be controlled by choosing a sufficiently small value for λ. For example, if we choose $\lambda \leq 1/m$, it will follow that

$$(1+\lambda\cdot\mu)^{m-1} \leq \left(1+\frac{\mu}{m}\right)^{m-1} < \left(1+\frac{\mu}{m}\right)^{m} \tag{25}$$

which will approach e^μ in the limit of m. Here, we use the known fact that $\lim_{m\to\infty}(1+\mu/m)^m = e^\mu$ with $\mu > 0$. As such, when m is sufficiently large the bound in Theorem 3.2 is upper-bounded with $m\cdot\lambda\cdot\ell\cdot(1+\lambda\cdot\mu)^{m-1}\cdot$ gradient-gap $\simeq \ell\cdot e^\mu\cdot$ gradient-gap which asserts that the worst-case performance gap of our offline optimizer is approaching (in the limit of m) $\ell\cdot e^\mu\cdot\max_{\mathbf{x}}\|\nabla g(\mathbf{x}) - \nabla g_\phi(\mathbf{x})\| = \mathbf{O}(\max_{\mathbf{x}}\|\nabla g(\mathbf{x}) - \nabla g_\phi(\mathbf{x})\|)$ which is not dependent on the no. of gradient steps.

B. Minimizing Eq. (12) Reduces Gradient Gap.

Intuitively, minimizing Eq. 12 will reduce the gradient gap. To formalize this intuition rigorously, we will show below that (1) in the limit of optimization if a parameterization ϕ can be found that zeroes out the loss in Eq. 12 over the entire input space, the gradient gap is zero; and (2) in more practical cases, where the loss in Eq. 12 is not zero, the gradient gap is still guaranteed to be upper-bound by the Lipschitz constant of the function gap, which decreases as we optimize the loss function in Eq. 12. These are detailed below.

A. The minimized loss in Eq. 12 is zero. In this case, let us define:

$$\mathbf{F}_\phi(\mathbf{x}, \mathbf{x}') \triangleq \int_0^1 \nabla g(t\mathbf{x} + (1-t)\mathbf{x}')\mathrm{d}t - \int_0^1 \nabla g_\phi(t\mathbf{x} + (1-t)\mathbf{x}')\mathrm{d}t . \tag{26}$$

The loss in Eq. 12 can be rewritten as

$$\mathfrak{L}_g(\phi) = \mathbb{E}\left[\left(\mathbf{F}_\phi\Big(\mathbf{x}, \mathbf{x}'\Big)^\top\Big(\mathbf{x} - \mathbf{x}'\Big)\right)^2\right] , \tag{27}$$

where the expectation is over all pairs $(\mathbf{x}, \mathbf{x}')$ from the input space. At the optimal ϕ, since $\mathfrak{L}_g(\phi) = 0$ as assumed,

$$\mathbf{F}_\phi\Big(\mathbf{x}, \mathbf{x}'\Big)^\top\Big(\mathbf{x} - \mathbf{x}'\Big) = 0 \tag{28}$$

for any choice of $(\mathbf{x}, \mathbf{x}')$. Next, by the line integration theorem, we also have

$$g(\mathbf{x}) - g(\mathbf{x}') \quad = \quad (\mathbf{x} - \mathbf{x}')^\top \left(\int_0^1 \nabla g(t\mathbf{x} + (1-t)\mathbf{x}') \mathrm{d}t \right) , \tag{29}$$

$$g_\phi(\mathbf{x}) - g_\phi(\mathbf{x}') \quad = \quad (\mathbf{x} - \mathbf{x}')^\top \left(\int_0^1 \nabla g_\phi(t\mathbf{x} + (1-t)\mathbf{x}') \mathrm{d}t \right) , \tag{30}$$

which together imply that

$$\mathbf{F}_\phi\Big(\mathbf{x}, \mathbf{x}'\Big)^\top \Big(\mathbf{x} - \mathbf{x}'\Big) \quad = \quad g(\mathbf{x}) - g(\mathbf{x}') - g_\phi(\mathbf{x}) + g_\phi(\mathbf{x}') \, . \tag{31}$$

Combining Eq. 28 and Eq. 31 results in

$$g(\mathbf{x}) - g_\phi(\mathbf{x}) \quad = \quad g(\mathbf{x}') - g_\phi(\mathbf{x}') \tag{32}$$

for any choice of $(\mathbf{x}, \mathbf{x}')$. This means there exists a constant c such that

$$g(\mathbf{x}) - g_\phi(\mathbf{x}) \quad = \quad c \tag{33}$$

for all $\mathbf{x}$. Thus, taking the derivative with respect to $\mathbf{x}$ on both sides of the above yields

$$\nabla g(\mathbf{x}) - \nabla g_\phi(\mathbf{x}) \quad = \quad 0 \, , \tag{34}$$

which implies immediately that the gradient gap is zero everywhere. Hence, optimizing Eq. 12 guarantees in principle that the gradient will be perfectly matched in the limit of data (i.e., when we take the expectation over the entire input space rather than over a finite set of offline data points).

B. The minimized loss in Eq. 12 is not zero. In this case, let us define

$$h(\mathbf{x}) \quad = \quad g(\mathbf{x}) - g_\phi(\mathbf{x}) \tag{35}$$

and it will follow that $|\mathbf{F}_\phi(\mathbf{x}, \mathbf{x}')^\top (\mathbf{x} - \mathbf{x}')| = |h(\mathbf{x}) - h(\mathbf{x}')|$ following Eq. 31 above. This means our loss function is working towards minimizing $(h(\mathbf{x}) - h(\mathbf{x}'))^2$ over $(\mathbf{x}, \mathbf{x}')$. This will makes $h(\mathbf{x})$ smoother as the output distance between different inputs are being reduced.

As a result, this process will reduce the Lipschitz constant ϵ of $h(\mathbf{x})$, which is defined to be the minimum value such that

$$|h(\mathbf{x}) - h(\mathbf{x}')| \quad \leq \quad \epsilon \cdot \|\mathbf{x} - \mathbf{x}'\| \, , \tag{36}$$

which implies

$$|(g(\mathbf{x}) - g(\mathbf{x}')) - (g_\phi(\mathbf{x}) - g_\phi(\mathbf{x}'))| \quad \leq \quad \epsilon \cdot \|\mathbf{x} - \mathbf{x}'\| \tag{37}$$

Now, dividing both sides by $\|\mathbf{x} - \mathbf{x}'\|$ yields

$$\left| \frac{g(\mathbf{x}) - g(\mathbf{x}')}{\|\mathbf{x} - \mathbf{x}'\|} - \frac{g_\phi(\mathbf{x}) - g_\phi(\mathbf{x}')}{\|\mathbf{x} - \mathbf{x}'\|} \right| \quad \leq \quad \epsilon \, . \tag{38}$$

Now, suppose we choose $\mathbf{x}' = \mathbf{x} + t \cdot \mathbf{e}_i$ where $\mathbf{e}_i$ is a d-dimensional one-hot vector with the hot component at the i-th position where d denotes the input dimension. So, the above is equivalent to

$$\left| \frac{g(\mathbf{x} + t \cdot \mathbf{e}_i) - g(\mathbf{x})}{t\|\mathbf{e}_i\|} - \frac{g_\phi(\mathbf{x} + t \cdot \mathbf{e}_i) - g_\phi(\mathbf{x})}{t\|\mathbf{e}_i\|} \right| \quad \leq \quad \epsilon \, , \tag{39}$$

or more expressively,

$$-\epsilon \quad \leq \quad \frac{g(\mathbf{x} + t \cdot \mathbf{e}_i) - g(\mathbf{x})}{t\|\mathbf{e}_i\|} - \frac{g_\phi(\mathbf{x} + t \cdot \mathbf{e}_i) - g_\phi(\mathbf{x})}{t\|\mathbf{e}_i\|} \quad \leq \quad \epsilon \tag{40}$$

Taking $\lim_{t\to 0}$ on all parts of the above inequality, the above can be rewritten as

$$-\epsilon \;\leq\; \lim_{t\to 0}\left(\frac{g(\mathbf{x}+t\cdot\mathbf{e}_i)-g(\mathbf{x})}{t\|\mathbf{e}_i\|}\right) - \lim_{t\to 0}\left(\frac{g_\phi(\mathbf{x}+t\cdot\mathbf{e}_i)-g_\phi(\mathbf{x})}{t\|\mathbf{e}_i\|}\right) \;\leq\; \epsilon\,. \tag{41}$$

Next, using the definition of directional gradient

$$\nabla_{\mathbf{r}}\, g(\mathbf{x}) \;\;=\;\; \lim_{t\to 0}\, \frac{1}{t}\Big(g(\mathbf{x}+t\cdot\mathbf{r})-g(\mathbf{x})\Big) \tag{42}$$

and the fact that $\nabla_{\mathbf{r}} g(\mathbf{x}) = \nabla g(\mathbf{x})^\top \mathbf{r}$ on $\mathbf{r}=\mathbf{e}_i$, we have

$$-\epsilon \;\leq\; \frac{1}{\|\mathbf{e}_i\|}\cdot\Big(\nabla g(\mathbf{x})^\top \mathbf{e}_i\Big) - \frac{1}{\|\mathbf{e}_i\|}\cdot\Big(\nabla g_\phi(\mathbf{x})^\top \mathbf{e}_i\Big) \;\leq\; \epsilon\,, \tag{43}$$

which implies that

$$\Big(\nabla g(\mathbf{x}) - \nabla g_\phi(\mathbf{x})\Big)^\top \mathbf{e}_i \;\;\leq\;\; \epsilon\cdot\|\mathbf{e}_i\| \;\;=\;\; \epsilon\,. \tag{44}$$

The last step is true because $\|\mathbf{e}_i\|=1$. Next, repeat the above argument with $\mathbf{r}=\mathbf{e}_i$ for $i=1,2,\ldots,d$ and summing both sides of the resulting inequalities over $i=1,2,\ldots,d$, we have

$$\Big\|\nabla g(\mathbf{x}) - \nabla g_\phi(\mathbf{x})\Big\|_1 \;\;\triangleq\;\; \sum_{i=1}^{d}\left[\Big(\nabla g(\mathbf{x}) - \nabla g_\phi(\mathbf{x})\Big)^\top \mathbf{e}_i\right] \;\;\leq\;\; d\cdot\epsilon \;=\; \mathbf{O}(\epsilon)\,. \tag{45}$$

Finally, we note that

$$\Big\|\nabla g(\mathbf{x}) - \nabla g_\phi(\mathbf{x})\Big\|_2 \;\leq\; \Big\|\nabla g(\mathbf{x}) - \nabla g_\phi(\mathbf{x})\Big\|_1 \;\leq\; \mathbf{O}(\epsilon)\,, \tag{46}$$

which completes our proof and asserts that the gradient gap is indeed bounded by the Lipschitz constant of the gap function $h(\mathbf{x}) = g(\mathbf{x}) - g_\phi(\mathbf{x})$ that decreases as we optimize the training objective.

C. Training and Evaluation Details of `MATCH-OPT`

We used a feed-forward neural network with 4 layers ($512 \to 128 \to 32 \to 1$) activated by the Leaky ReLU function as the surrogate model for `MATCH-OPT`. For each task, we trained the model using Adam optimizer with 1e-4 learning rate and a batch size of 128 for 200 epochs.

During the evaluation, we employed gradient updates for 150 iterations uniformly across all the tasks. This evaluation procedure used an Adam optimizer with a 0.01 learning rate for all discrete tasks, and a 0.001 learning rate for all continuous tasks. We chose a larger learning rate for discrete tasks since the discrete inputs are converted into logits (same as all baselines).

D. Extended Theoretical Analysis to Incorporate the Value Matching Regularizer

This section discusses an extension of the original theoretical analysis in Section 3 to provide a theoretical condition under which the worst-case optimization in Theorem 3.2 has a tighter bound. We will show that this bound is expressed in terms of both the gradient and value matching quantities, which inspires the addition of the value-matching regularizer in Eq. (14) to the original loss in Eq. (13). This is formalized in Theorem D.1 below.

Theorem D.1 (Generalized worst-case optimization risk bound)**.** *Suppose the target objective function $g(\mathbf{x})$ is a ℓ-Lipschitz continuous and μ-Lipschitz smooth function. For all $a\in(0,1)$, the worst-case performance gap, $\mathfrak{G}_{m,\lambda} \triangleq \max_{\mathbf{x}} \mathfrak{G}_{m,\lambda}(\mathbf{x})$, between g and some arbitrary surrogate g_ϕ with Lipschitz constant ℓ_ϕ is upper-bounded by:*

$$\begin{aligned}\mathfrak{G}_{m,\lambda} \;&\leq\; m\cdot 2a\cdot\max_{\mathbf{x}}\Big\|g(\mathbf{x})-g_\phi(\mathbf{x})\Big\| \\ &+\; m\cdot\Big(\ell + a\cdot(\ell_\phi-\ell)\Big)\cdot\Big(1+\lambda\mu\Big)^{m-1}\cdot\max_{\mathbf{x}}\Big\|\nabla_{\mathbf{x}} g(\mathbf{x}) - \nabla_{\mathbf{x}} g_\phi(\mathbf{x})\Big\|\,.\end{aligned} \tag{47}$$

Proof. First, let us start from Eq. (20),

$$\begin{aligned}\left\|\mathfrak{R}_g^k(\mathbf{u}^0)-\mathfrak{R}_g^k(\mathbf{v}^0)\right\| &= \left\|g(\mathbf{v}^k)-g(\mathbf{u}^k)\right\| &= \left\|g(\mathbf{v}^k)-g_\phi(\mathbf{v}^k)+g_\phi(\mathbf{v}^k)-g_\phi(\mathbf{u}^k)+g_\phi(\mathbf{u}^k)-g(\mathbf{u}^k)\right\| \\ &&\leq \left\|g(\mathbf{v}^k)-g_\phi(\mathbf{v}^k)\right\|+\left\|g_\phi(\mathbf{v}^k)-g_\phi(\mathbf{u}^k)\right\|+\left\|g_\phi(\mathbf{u}^k)-g(\mathbf{u}^k)\right\|\end{aligned} \tag{48}$$

Next, similar to the derivation of Lemma A.2, we have:

$$\left\|g_\phi(\mathbf{v}^k)-g_\phi(\mathbf{u}^k)\right\| \leq \ell_\phi\cdot(1+\lambda\mu)^k\cdot\left\|\mathbf{v}^0-\mathbf{u}^0\right\| \tag{49}$$

with ℓ_ϕ denotes the Lipschitz constant of the surrogate $g_\phi(\mathbf{x})$.

Now, let $\mathcal{E}_k(\mathbf{u}^0,\mathbf{v}^0)\triangleq|\mathfrak{R}_g^k(\mathbf{u}^0)-\mathfrak{R}_g^k(\mathbf{v}^0)|$ as defined in Lemma A.2 of Appendix A. It follows that

$$\mathcal{E}_k(\mathbf{u}^0,\mathbf{v}^0) \leq \left\|g(\mathbf{v}^k)-g_\phi(\mathbf{v}^k)\right\|+\left\|g_\phi(\mathbf{u}^k)-g(\mathbf{u}^k)\right\|+\ell_\phi\cdot(1+\lambda\mu)^k\cdot\left\|\mathbf{v}^0-\mathbf{u}^0\right\|. \tag{50}$$

Let $k=m-1$, $\mathbf{u}^0=\mathbf{x}_\phi^1$, $\mathbf{v}^0=\mathbf{x}_*^1$. Similar to Eq. (22), we have

$$\begin{aligned}\mathcal{E}_{m-1}(\mathbf{x}_\phi^1,\mathbf{x}_*^1) &\leq \left\|g(\mathbf{v}^{m-1})-g_\phi(\mathbf{v}^{m-1})\right\|+\left\|g_\phi(\mathbf{u}^{m-1})-g(\mathbf{u}^{m-1})\right\| \\ &+ \ell_\phi\cdot(1+\lambda\mu)^{m-1}\cdot\left\|\nabla_{\mathbf{x}}g(\mathbf{x}^0)-\nabla_{\mathbf{x}}g(\mathbf{x}^0)\right\|.\end{aligned} \tag{51}$$

Additionally, the following was also shown in Lemma A.2:

$$\mathcal{E}_{m-1}(\mathbf{x}_\phi^1,\mathbf{x}_*^1) \leq \lambda\ell\cdot(1+\lambda\mu)^{m-1}\cdot\left\|\nabla_{\mathbf{x}}g(\mathbf{x}^0)-\nabla_{\mathbf{x}}g_\phi(\mathbf{x}^0)\right\|. \tag{52}$$

Now, combining Eq. (51) and Eq. (52) with $a\in(0,1)$,

$$\begin{aligned}\mathcal{E}_{m-1}(\mathbf{x}_\phi^1,\mathbf{x}_*^1) &\leq a\cdot\left(\left\|g(\mathbf{v}^{m-1})-g_\phi(\mathbf{v}^{m-1})\right\|+\left\|g_\phi(\mathbf{u}^{m-1})-g(\mathbf{u}^{m-1})\right\|\right) \\ &+ \Big((1-a)\cdot\lambda\ell+a\cdot\ell_\phi)\Big)\cdot(1+\lambda\mu)^{m-1}\cdot\left\|\nabla_{\mathbf{x}}g(\mathbf{x}^0)-\nabla_{\mathbf{x}}g_\phi(\mathbf{x}^0)\right\|.\end{aligned} \tag{53}$$

Plugging the above new bound to Eq. (23) in Appendix A,

$$\begin{aligned}\mathfrak{G}_{m,\lambda}=\max_{\mathbf{x}^0}\mathfrak{G}_{m,\lambda}(\mathbf{x}^0) &\leq \max_{\mathbf{x}^0}\mathfrak{G}_{m-1,\lambda}(\mathbf{x}^0) \\ &+ \max_{\mathbf{x}^0} a\cdot\left(\left\|g(\mathbf{v}^{m-1})-g_\phi(\mathbf{v}^{m-1})\right\|+\left\|g_\phi(\mathbf{u}^{m-1})-g(\mathbf{u}^{m-1})\right\|\right) \\ &+ \max_{\mathbf{x}^0}\Big((1-a)\cdot\lambda\ell+a\cdot\ell_\phi)\Big)\cdot(1+\lambda\mu)^{m-1}\cdot\left\|\nabla_{\mathbf{x}}g(\mathbf{x}^0)-\nabla_{\mathbf{x}}g_\phi(\mathbf{x}^0)\right\|.\end{aligned} \tag{54}$$

Rearranging the above gives the following upper bound:

$$\begin{aligned}\mathfrak{G}_{m,\lambda}-\mathfrak{G}_{m-1,\lambda} &\leq \max_{\mathbf{x}^0} 2a\cdot\left\|g(\mathbf{x}^0)-g_\phi(\mathbf{x}^0)\right\| \\ &+ \max_{\mathbf{x}^0}\Big((1-a)\cdot\lambda\ell+a\cdot\ell_\phi)\Big)\cdot\Big(1+\lambda\mu\Big)^{m-1}\left\|\nabla_{\mathbf{x}}g(\mathbf{x}^0)-\nabla_{\mathbf{x}}g_\phi(\mathbf{x}^0)\right\|,\end{aligned} \tag{55}$$

where the first term on the RHS is derived from the fact that the maximum of an unconstrained optimization program upper bounds that of a constrained optimization program (the dependency of $\mathbf{u}^{m-1}$ and $\mathbf{v}^{m-1}$ on $\mathbf{x}^0$ can be seen as constraints). Applying this m times consecutively gives:

$$\begin{aligned}\mathfrak{G}_{m,\lambda}-\mathfrak{G}_{0,\lambda} &\leq 2am\cdot\max_{\mathbf{x}}\left\|g(\mathbf{x})-g_\phi(\mathbf{x})\right\| \\ &+ m\cdot\Big((1-a)\cdot\lambda\ell+a\cdot\ell_\phi)\Big)\cdot\Big(1+\lambda\mu\Big)^{m-1}\cdot\max_{\mathbf{x}}\left\|\nabla_{\mathbf{x}}g(\mathbf{x})-\nabla_{\mathbf{x}}g_\phi(\mathbf{x})\right\|.\end{aligned} \tag{56}$$

Since $\mathfrak{G}_{0,\lambda} = 0$, this implies:

$$\begin{aligned}\mathfrak{G}_{m,\lambda} &\leq 2am \cdot \max_{\mathbf{x}} \left\| g(\mathbf{x}) - g_\phi(\mathbf{x}) \right\| \\ &+ m \cdot \Big((1-a) \cdot \lambda\ell + a \cdot \ell_\phi) \Big) \cdot \Big(1 + \lambda\mu \Big)^{m-1} \cdot \max_{\mathbf{x}} \left\| \nabla_{\mathbf{x}} g(\mathbf{x}) - \nabla_{\mathbf{x}} g_\phi(\mathbf{x}) \right\| . \end{aligned} \tag{57}$$

Using the above result, we can compute the difference between the RHS of this bound in Eq. (57) and that of the original bound in Eq. (7), which is $2am \cdot \max_{\mathbf{x}} \|g(\mathbf{x}) - g_\phi(\mathbf{x})\| + am \cdot (\ell_\phi - \lambda\ell) \cdot (1+\lambda\mu)^{m-1} \cdot \max_{\mathbf{x}} \|\nabla_{\mathbf{x}} g(\mathbf{x}) - \nabla_{\mathbf{x}} g_\phi(\mathbf{x})\|$. The second term of this difference could be negative as ℓ_ϕ decreases beyond $\lambda\ell$ (see Appendix B for intuition on why ℓ_ϕ decreases as we train g_ϕ). As a result, when ℓ_ϕ and the ratio $\max_{\mathbf{x}} \|g(\mathbf{x}) - g_\phi(\mathbf{x})\| / \max_{\mathbf{x}} \|\nabla_{\mathbf{x}} g(\mathbf{x}) - \nabla_{\mathbf{x}} g_\phi(\mathbf{x})\|$ is sufficiently small, or such that:

$$\ell_\phi + \frac{2 \cdot \max_{\mathbf{x}} \left\| g(\mathbf{x}) - g_\phi(\mathbf{x}) \right\|}{(1+\lambda\mu)^{m-1} \cdot \max_{\mathbf{x}} \left\| \nabla_{\mathbf{x}} g(\mathbf{x}) - \nabla_{\mathbf{x}} g_\phi(\mathbf{x}) \right\|} \leq \lambda\ell \,, \tag{58}$$

the bound will become tighter, thus justifying the importance of the value matching term. Developing a training algorithm to substantiate the above condition will be part of our follow-up work. □

E. Mean and Standard Deviation Results

As mentioned in the evaluation methodology, we ran each method for 4 different runs. This section reports the mean results from Tables 1 and 2 along with the corresponding standard deviations.

METHOD	ANT	DKITTY	HOPPER
GA	0.271 ± 0.013	0.895 ± 0.013	0.780 ± 0.462
ENS-MEAN	0.517 ± 0.039	0.899 ± 0.010	1.524 ± 0.710
ENS-MIN	0.536 ±0.031	0.908 ±0.019	1.42 ±0.645
CMA-ES	**0.974** ± 0.556	0.722 ± 0.001	0.620 ±0.151
MINS	0.910 ± 0.034	0.939 ± 0.003	0.150 ± 0.186
CBAS	0.842 ± 0.015	0.879 ± 0.002	0.150 ± 0.014
ROMA	0.832 ± 0.055	0.880 ± 0.008	2.026 ± 0.225
BONET	0.927 ±0.002	0.954 ± 0.0001	0.395 ± 0.0002
COMS	0.885 ± 0.024	0.953 ± 0.016	**2.270** ±0.237
MATCH-OPT	0.931 ± 0.011 (2)	**0.957 ± 0.014 (1)**	1.572 ± 0.322 (3)
METHOD	SCON	TF8	TF10
GA	0.699 ± 0.054	0.954 ± 0.020	0.966 ± 0.026
ENS-MEAN	0.716 ± 0.065	0.926 ±0.005	**0.968** ± 0.019
ENS-MIN	0.734 ± 0.058	0.959 ± 0.052	0.959 ± 0.021
CMA-ES	**0.757** ± 0.013	**0.978** ± 0.007	0.966 ± 0.007
MINS	0.690 ± 0.024	0.900 ± 0.059	0.759 ±0.031
CBAS	0.659 ±0.086	0.916 ± 0.035	0.928 ±0.013
ROMA	0.704 ±0.032	0.664 ±0.015	0.820 ±0.014
BONET	0.500 ±0.002	0.911 ± 0.005	0.756 ±0.006
COMS	0.565 ± 0.012	0.968 ±0.018	0.873 ±0.053
MATCH-OPT	0.732 ±0.003 (3)	0.977 ±0.004 (2)	0.924 ±0.038 (6)

Table 5. Comparing MATCH-OPT and other baselines based on the 100$^{\text{th}}$ percentile of the solutions (i.e., maximum solution) generated by each method. Each cell shows the mean and standard deviation of the function values found by each method over 4 runs. The individual rank of our method is included next to its reported performance for each benchmark.

METHOD	ANT	DKITTY	HOPPER
GA	0.130 ± 0.029	0.742 ±0.012	0.089 ± 0.07
ENS-MEAN	0.192 ±0.010	0.791 ± 0.019	0.209 ± 0.035
ENS-MIN	0.190 ±0.006	0.803 ± 0.005	0.166 ±0.052
CMA-ES	-0.049 ± 0.003	0.482 ± 0.171	-0.033 ± 0.006
MINS	0.614 ± 0.034	0.889 ± 0.004	0.088 ± 0.170
CBAS	0.376 ± 0.023	0.757 ± 0.005	0.013 ± 0.002
ROMA	0.448 ± 0.013	0.760 ± 0.028	0.370 ± 0.008
BONET	**0.620** ±0.003	**0.897** ± 0.0001	0.390 ± 0.0002
COMS	0.557 ± 0.015	0.879 ± 0.001	0.379 ±0.005
MATCH-OPT	0.611± 0.007 (3)	0.887±0.003 (3)	**0.393** ± 0.005 (1)
METHOD	SCON	TF8	TF10
GA	0.641 ± 0.036	0.510 ± 0.055	0.794 ± 0.013
ENS-MEAN	0.644 ± 0.070	0.529 ± 0.030	**0.796** ± 0.006
ENS-MIN	**0.672** ± 0.042	0.490 ± 0.052	0.794 ± 0.008
CMA-ES	0.590 ± 0.012	0.592 ± 0.015	0.786 ± 0.009
MINS	0.414 ± 0.011	0.420 ± 0.009	0.465 ± 0.016
CBAS	0.099 ± 0.008	0.442 ± 0.038	0.613 ± 0.012
ROMA	0.420 ± 0.030	0.560 ± 0.104	0.780 ± 0.400
BONET	0.470 ± 0.004	0.505 ± 0.004	0.465 ± 0.002
COMS	0.414 ±0.023	**0.652** ± 0.108	0.606 ± 0.027
MATCH-OPT	0.439 ± 0.016(6)	0.594 ± 0.015(2)	0.720 ±0.015 (6)

Table 6. Comparing MATCH-OPT and baselines based on 50^{th} percentile of the solutions (i.e., median solution) generated by each method. Each cell shows the mean and standard deviation of the function values found by each method over 4 runs. The individual rank of our method is included next to its reported performance for each benchmark.

F. Rank Distribution Plots

To further illustrate the reliability of MATCH-OPT, this section visualizes the entire rank distribution of the tested algorithm across different percentile level (i.e., 25, 50, 75 and 100). Overall, we observe that MATCH-OPT (colored in red) consistently achieves lower mean and standard deviation of performance across all datasets at every percentile level, as compared to that of other baselines. This observation corroborates previous results presented in the main text, and confirms our hypothesis regarding the robustness of MATCH-OPT.

G. Additional Experiments

In addition to the results reported in the main text, we have also compared MATCH-OPTwith three additional baselines, which include DDOM (Krishnamoorthy et al., 2023a), BO-qEI (Wilson et al., 2017) and BDI (Chen et al., 2022). The results are reported in Table 7 and Table 8 below.

METHOD	ANT	DKITTY	HOPPER	SCON	TF8	TF10
MATCH-OPT	0.931	0.957	1.572	0.732	0.977	0.924
DDOM	0.768	0.911	-0.261	0.570	0.674	0.538
BDI	0.967	0.940	1.706	0.735	0.973	OOM
BO-qEI	0.812	0.896	0.528	0.576	0.607	0.864

Table 7. Performance comparison between versions of MATCH-OPT with DDOM, BO-qEI and BDI at the 100^{th} performance percentile (i.e., maximum solution). OOM indicates that the method runs out of memory

In both the 50-th and 100-th percentile settings, it appears MATCH-OPT outperforms DDOM in all tasks. Furthermore,

the results also show that MATCH-OPT performs the best in 6 out of 12 cases (across both the 100-th and 50-th percentile settings) while BO-qEI only performs best in 1 out of 12 cases. BDI performs best in 5 out of 12 cases, runs out of memory in 2 out of 12 cases. Overall, MATCH-OPT appears to perform more stable than BDI and is marginally better than BDI. It is also more memory-efficient than BDI as it does run successfully in all cases, while BDI runs out of memory in 2 cases. MATCH-OPT also outperforms BO-qEI significantly in 11 out of 12 cases.

H. Running Time

We also report the running time achieved by all tested algorithms in Table 9.

All reported running times are in seconds. Our algorithm incurs more time than other baselines but its total running time is still affordable in the offline setting: 4785s = 1.32hr. We do, however, want to remark that such complexity comparison is only tangential to our main contribution. Our main focus is on building optimizer with better and more stable performance overall, even at an affordable increase of running time. Furthermore, we want to point out that as some of the baselines (such as BONET) use an entirely different model which has a different number of parameters than ours, the reported running times here might not be comparable on the same compute platform. The computations were performed on a Ubuntu machine with a 3.73GHz AMD EPYC 7313 16-Core Processor (32 cores, 251 GB RAM) and two NVIDIA RTX A6000 GPUs. Each has 48 GB RAM.

I. Limitation

One potential limitation of our approach in comparison to other baselines is that our gradient match algorithm learns from pairs of data points. Thus, the total number of training pairs it needs to consume grows quadratically in the number of offline data points. For example, an offline dataset with N examples will result in a set of $O(N^2)$ training pairs for our algorithm, which increases the training time quadratically. However, an intuition here is that training pairs are not equally informative and, in our experiments, it suffices to get competitive performance by just focusing on pairs of data along the sampled trajectories with monotonically increasing objective function values. This allows us to keep training cost linearly with respect to N.

On the other hand, while it is true that none of the existing baselines (including our algorithm) outperform others on all tasks, we believe that at least on these benchmark datasets, our algorithm tends to perform most stably across all tasks, as measured by the mean averaged rank reported in each of our performance tables. This is a single metric that is computed based on the performance of all baselines across all tasks. The end-user can make a judgment based on such metrics. In practice, by looking at how existing baselines perform overall on a set of benchmark tasks that are similar to a target task, one can decide empirically which baseline is most likely to be best for the target task.

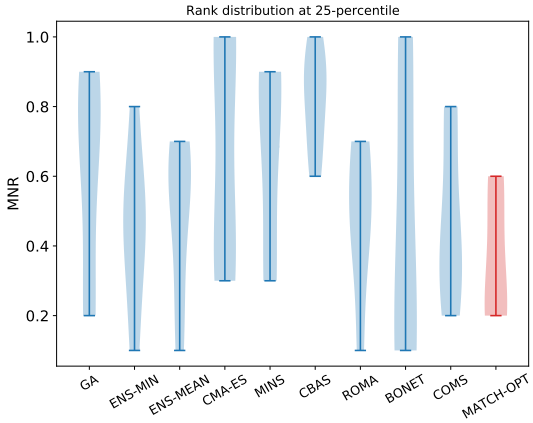
METHOD	ANT	DKITTY	HOPPER	SCON	TF8	TF10
MATCH-OPT	0.611	0.887	0.393	0.439	0.594	0.720
DDOM	0.554	0.868	-0.570	0.390	0.418	0.461
BDI	0.583	0.870	0.400	0.480	0.595	OOM
BO-qEI	0.568	0.883	0.360	0.490	0.439	0.557

Table 8. Performance comparison between versions of MATCH-OPT with DDOM, BO-qEI and BDI at the 50^{th} performance percentile (i.e., maximum solution). OOM indicates that the method runs out of memory

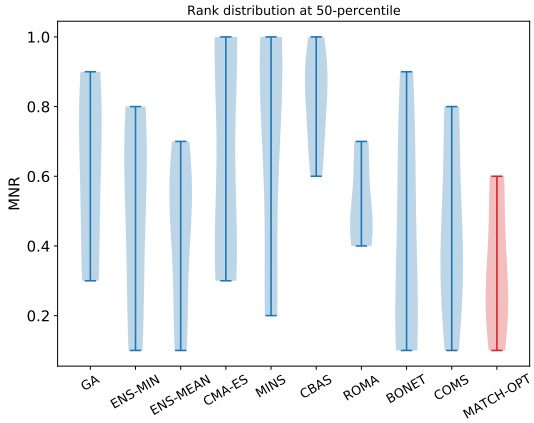
	OURS	BO-qEI	CMA-ES	ROMA	MINS	CBAS
TIME	4785	111	3804	489	359	189

	BONET	GA	ENS-MEAN	ENS-MIN	DDOM
TIME	614	45	179	179	2658

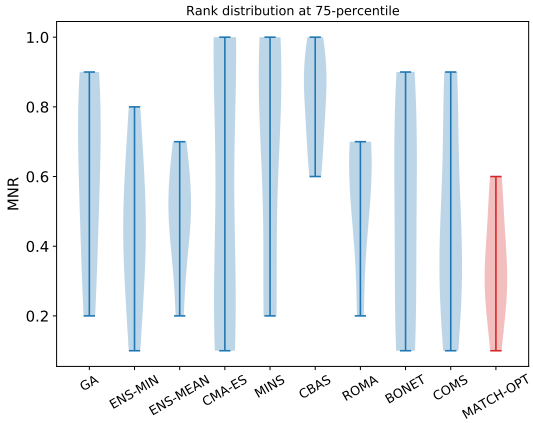
Table 9. Total running time (in seconds) of all tested baselines.



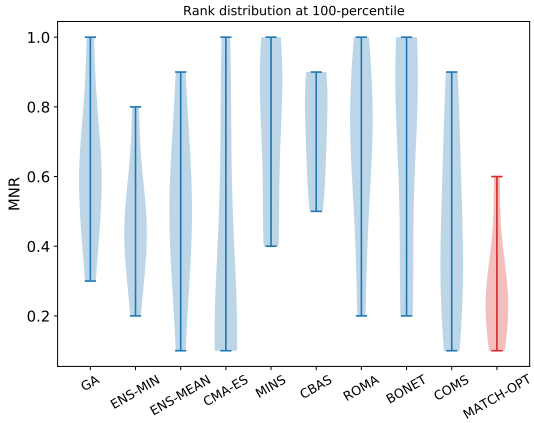
(a) Mean Rank (25th percentile)



(b) Mean Rank (50th percentile)



(a) Mean Rank (75th percentile)



(b) Mean Rank (100th percentile)

Figure 4. Plots of distributions of mean normalized rank (MNR) of the tested algorithms across all tasks at the (a) 25-th, (b) 50-th, (c) 75-th, and (d) 100-th performance percentile levels.