Cliqueformer: Model-Based Optimization with Structured Transformers

Jakub Grudzien Kuba[1], **Pieter Abbeel**[1], **Sergey Levine**[1]

[1]BAIR, UC Berkeley
{kuba, pabbeel, sergey.levine}@berkeley.edu

Abstract

Large neural networks excel at prediction tasks, but their application to design problems, such as protein engineering or materials discovery, requires solving offline model-based optimization (MBO) problems. While predictive models may not directly translate to effective design, recent MBO algorithms incorporate reinforcement learning and generative modeling approaches. Meanwhile, theoretical work suggests that exploiting the target function's structure can enhance MBO performance. We present Cliqueformer, a transformer-based architecture that learns the black-box function's structure through functional graphical models (FGM), addressing distribution shift without relying on explicit conservative approaches. Across various domains, including chemical and genetic design tasks, Cliqueformer demonstrates superior performance compared to existing methods.

1 Introduction

Most of the common use cases of deep learning (DL) so far have taken the form of prediction tasks (Hochreiter and Schmidhuber 1997; He et al. 2016; Krizhevsky, Sutskever, and Hinton 2017). However, in many applications, such as protein synthesis or chip design, we might want to use powerful models to instead solve *optimization* problems. Clearly, accurate predictions of a target score of an object could be used to find a design of the object that maximizes that score. Such a methodology is particularly useful in engineering problems in which evaluating solution candidates comes with big risk. For example, synthesizing a proposed protein requires a series of wet lab experiments and induces extra cost and human effort (Gómez-Bombarelli et al. 2018; Brookes, Park, and Listgarten 2019). Thus, to enable proposing *de-novo* generation of strong solution candidates, researchers have drawn their attention to offline *model-based optimization* (MBO). In this paradigm, first, a surrogate model of the score is learned from offline data. Next, a collection of designs is trained to maximize the surrogate, and then proposed as candidates for maximizers of the target score (Gómez-Bombarelli et al. 2018; Kumar et al. 2021).

Unfortunately, MBO introduces unique challenges not encountered in classical prediction tasks. The most significant issue arises from the incomplete coverage of the design space by the data distribution. This limitation leads to a phenomenon known as *distribution shift*, where optimized designs drift away from the original data distribution. Consequently, this results in poor proposals with significantly overestimated scores (Trabucco et al. 2022; Geng 2023). To address it, popular MBO algorithms have been employing techniques from offline reinforcement learning (Kumar et al. 2020; Trabucco et al. 2021) and generative modeling (Kumar and Levine 2020; Mashkaria, Krishnamoorthy, and Grover 2023) to enforce the in-distribution constraint. Meanwhile, much of the recent success of DL has been driven by domain-specific neural networks that, when scaled together with the amount of data, lead to increasingly better performance. While researchers have managed to establish such models in several fields, it is not immediately clear how to do it in MBO. Recent theoretical work, however, has shown that MBO methods can benefit from information about the target function's *structure* expressed by its *functional graphical model* (Grudzien et al. 2024, FGM). This insight opens up new possibilities for developing more effective MBO models by injecting such structure into their architecture. However, how to integrate such decompositions into scalable neural networks remains an open question, and addressing this challenge is the focus of this work.

In contrast to previous works, in our paper, we develop a scalable model that tackles MBO by learning the structure of the black-box function through the formalism of functional graphical models. Our architecture aims to solve MBO by *1)* decomposing the predictions over the *cliques* of the function's FGM, and *2)* enforcing the cliques' marginals to have wide coverage with our novel form of the variational bottleneck (Kingma and Welling 2013; Alemi et al. 2016). However, building upon our Theorem 2, we do not follow (Grudzien et al. 2024) during the FGM discovery step, and instead subsume it in the learning algorithm. To enable scaling to high-dimensional problems and large datasets, we employ the transformer backbone (Vaswani et al. 2017). Empirically, we show that our model, *Cliqueformer*, inherits the scalability guarantees of MBO with FGM. We further demonstrate its superiority to baselines in a suite of tasks with latent radial-basis functions (Grudzien et al. 2024) and real-world chemical (Hamidieh 2018) and DNA design tasks (Trabucco et al. 2022; Uehara et al. 2024a).

2 Preliminaries

In this section, we provide the necessary background on offline model-based optimization. Additionally, we cover the basics of functional graphical models on top of which we build Cliqueformer.

Offline Model-based Optimization

We consider an offline model-based optimization (MBO for short) problem, where we are given a dataset $\mathcal{D} = \{\mathbf{x}^i, \mathrm{y}^i\}_{i=1}^N$ of examples $\mathbf{x}$ from a *design space* $\mathcal{X}$, following distribution $p(\mathbf{x})$, and their values $\mathrm{y} = f(\mathbf{x}) \in \mathbb{R}$ under an unkown (black-box) function $f(\mathbf{x})$. Our goal is to optimize this function **offline**. That is, to find its maximizer

$$\mathbf{x}^\star = \arg\max_{\mathbf{x} \in \mathcal{X}} f(\mathbf{x}) \tag{1}$$

by only using information provided in $\mathcal{D}$ (Kumar and Levine 2020). Sometimes, a more general objective in terms of a *policy* over $\pi(\mathbf{x})$ is also used, $\eta(\mathbf{x}) = \mathbb{E}_{\mathbf{x}\sim\pi}[f(\mathbf{x})]$. In either form, unlike in Bayesian optimization, we cannot make additional queries to the black-box function (Brochu, Cora, and De Freitas 2010; Kumar and Levine 2020). This formulation represents settings in which obtaining such queries is prohibitively costly, such as tests of new chemical molecules or of new hardware architectures (Kim et al. 2016; Kumar et al. 2021; Yang et al. 2024).

To solve MBO, it is typical to learn a model $f_\theta(\mathbf{x})$ of $f(\mathbf{x})$ parameterized by a vector θ with a regression method and data from $\mathcal{D}$,

$$L(\theta) = \mathbb{E}_{(\mathbf{x},\mathrm{y})\sim\mathcal{D}}\left[\left(f_\theta(\mathbf{x}) - \mathrm{y}\right)^2\right] + \mathrm{Reg}(\theta) \tag{2}$$

where $\mathrm{Reg}(\theta)$ is an optional regularizer. Classical methods choose $\mathrm{Reg}(\theta)$ to be identically zero, while conservative methos use the regularizer to bring the values of examples out of $\mathcal{D}$ down. For example, the regularizer of Conservative Objective Model's (Trabucco et al. 2021, COMs) is

$$\mathrm{Reg}_{\mathrm{com}}(\theta) = \alpha\big(\mathbb{E}_{\mathbf{x}\sim\mu_{\theta_\perp}}[f_\theta(\mathbf{x})] - \mathbb{E}_{\mathbf{x}\sim\mathcal{D}}[f_\theta(\mathbf{x})]\big), \quad \alpha > 0,$$

where $(\cdot)_\perp$ is the stop-gradient operator and $\mu_{\theta_\perp}(\mathbf{x})$ is the distribution obtained with a few gradient ascent steps on $\mathbf{x}$ initialized from $\mathcal{D}$. This distribution depends on the value of θ but is not differentiated through while computing the loss's gradient, and thus we denote it by $\theta_\perp$.

Unfortunately, in addition to the extra computational cost that the inner-loop gradient ascent induces, COMs's regularizer limits the amount of improvement that it allows its designs to make. This is particularly frustrating since recent work on functional graphical models (Grudzien et al. 2024, FGM) delivered a premise of large improvements in the case when the black-box function's graph can be discovered, as we explain in the next subsection.

Functional Graphical Models

An FGM of a high-dimensional function $f(\mathbf{x})$ is a graph over components of $\mathbf{x}$ that separates $\mathrm{x}_i, \mathrm{x}_j \in \mathbf{x}$ if their contributions to $f(\mathbf{x})$ are independent of each other. Knowing such a structure of f one can eliminate interactions between independent variables from a model that approximates it, and thus prevent an MBO algorithm from adversarially exploiting them. We summarize basic properties of FGMs below. In what follows, we denote $\mathcal{X}_{-i}$ as the design (input) space without the i^{th} subspace, and $\mathbf{x}_{-i}$ as its element [1].

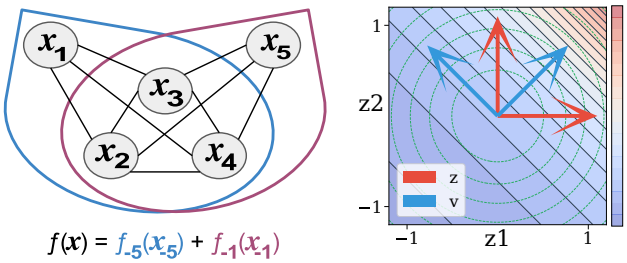


Figure 1: *Left.* An FGM of a 5D function which decomposes as $f(\mathbf{x}) = f_{-5}(\mathbf{x}_{-5}) + f_{-1}(\mathbf{x}_{-1})$. By Definition 1, nodes x_1 and x_5 are not linked. *Right.* Illustration of construction in the proof of Theorem 2 in 2D. Red axes ($\mathbf{z}$) show contour lines of f depending on both coordinates z_1 and z_2. Blue axes ($\mathbf{v}$) show the same contours depending only on v_1 after rotation. The Gaussian distribution (green circles) maintains circular shape in both coordinate systems, demonstrating invariance under rotation.

Definition 1. *Let $\mathbf{x} = (\mathrm{x}_v \mid v \in \mathcal{V})$ be a joint variable with index set $\mathcal{V}$, and $f(\mathbf{x})$ be a real-valued function. An FGM $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ of $f(\mathbf{x})$ is a graph where the edge set $\mathcal{E} \subset \mathcal{V}^2$ is such that,*

$$\exists\, f_{-i} : \mathcal{X}_{-i} \to \mathbb{R} \text{ and } f_{-j} : \mathcal{X}_{-j} \to \mathbb{R}, \text{ with } f(\mathbf{x}) = f_{-i}(\mathbf{x}_{-i}) + f_{-j}(\mathbf{x}_{-j}), \text{implies } (i,j) \notin \mathcal{E}.$$

See Figure 1 for illustration.

Crucially, FGMs enable decomposing the target function into sub-functions with smaller inputs defined by the set of maximal cliques $\mathcal{C}$ of the FGM's graph,

$$f(\mathbf{x}) = \sum_{C \in \mathcal{C}} f_C(\mathbf{x}_C). \tag{3}$$

Intuitively, the decomposition enables more efficient learning of the target function since it can be constructed by adding together functions defined on smaller inputs, which are easier to learn. This, in turn, allows for more efficient MBO since the joint solution $\mathbf{x}^\star$ can be recovered by *stitching* individual solutions $\mathbf{x}^\star_C$ to smaller problems. This intuition is formalized by the following theorem.

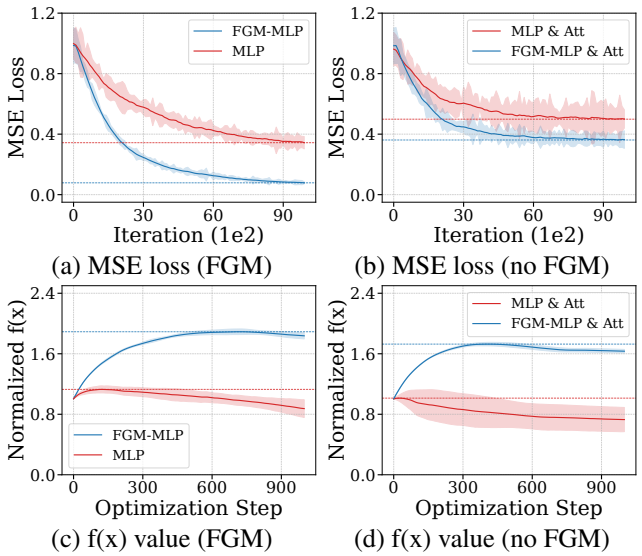
Theorem 1 ((Grudzien et al. 2024)). *For a real-valued function $f(\mathbf{x})$ with FGM with maximal cliques $\mathcal{C}$, and policy class Π, the regret of MBO with FGM information satisfies*

$$\eta(\pi^\star) - \eta(\hat{\pi}_{FGM}) \leq C_{stat} C_{cpx} \max_{\pi\in\Pi, \mathbf{x}\in\mathcal{X}, C\in\mathcal{C}} \frac{\pi(\mathbf{x}_C)}{p_C(\mathbf{x}_C)}$$

where C_{stat}, C_{cpx} are distribution and complexity constants defined in Appendix B.

This theorem implies that a function approximator equipped with FGM does not require the dataset to cover

[1] For example, if $\mathcal{X} = \mathcal{X}_1 \times \mathcal{X}_2 \times \mathcal{X}_3$ and $\mathbf{x} = (\mathrm{x}_1, \mathrm{x}_2, \mathrm{x}_3)$, then $\mathcal{X}_{-2} = \mathcal{X}_1 \times \mathcal{X}_3$, and $\mathbf{x}_{-2} = (\mathrm{x}_1, \mathrm{x}_3)$.



(a) MSE loss (FGM) (b) MSE loss (no FGM)

(c) f(x) value (FGM) (d) f(x) value (no FGM)

Figure 2: *Left column. FGM known.* The model without the FGM information (red) fits the data poorly (Fig. (a)) and leads to poor designs (Fig. (b)). The model with FGM information (blue) achieves a good fit and leads to designs largely superior to the data. *Right column. FGM unknown.* The model with *an* FGM decomposition (blue) achieves a slightly better fit than an oblivious model (red) but leads to significantly better designs.

the entire design space. Instead, it suffices for the dataset to span individual cliques of the space, which is a substantially less stringent condition—particularly when the cliques are small. To illustrate this result numerically, we conduct an experiment, shown in the left column of Figure 2. Specifically, we generate high-dimensional Gaussian data $\mathbf{x}$ and evaluate $f(\mathbf{x})$, which is modeled as a mixture of random radial basis functions (RBFs) constrained by an FGM of triangles. We then train two models, $f_\theta(\mathbf{x})$, each with the same parameter count: one adhering to the FGM decomposition and the other not. These models are used both to approximate the target function and to optimize $\mathbf{x}$. The results demonstrate that the FGM-equipped model significantly outperforms its counterpart, both in terms of the quality of fit and, more critically, in the quality of the optimized designs.

In the next section, we show how these results can be combined with a transformer, mitigate the distribution shift problem, and enable efficient MBO.

3 Cliqueformer

This section introduces a neural network model to solve MBO problems through standard end-to-end training on offline datasets. We present a new theoretical result, outline the key desiderata for such a model, and propose an architecture—*Cliqueformer*—that addresses these requirements.

Structure Discovery

The regret bound from Theorem 1 applies to methods that use the target function's FGM in their function approximation. It implies that such methods can solve even very high-dimensional problems if their underlying FGMs have low-dimensional cliques or, simply speaking, are *sparse*. Since, in general, no assumptions about the input can be made, this motivates learning a representation of the input for which one can make distributional assumptions and infer the FGM with statistical tests. Following this reasoning, Grudzien et al. (2024) offer a heuristic technique for discovering an FGM over learned, latent, normally-distributed variables. However, as we formalize with the following theorem, which we prove in Appendix B, even such attempts are futile in dealing with black-box functions.

Theorem 2. *For any $d \geq 2$ and random variable $\mathbf{x} \in \mathbb{R}^d$ with positive probability density, there exists a function $f(\mathbf{x})$ and two standard-normal reparameterizations $\mathbf{z} = z(\mathbf{x})$, $\mathbf{v} = v(\mathbf{x})$ such that the FGM of f is complete with respect to $\mathbf{z}$ but empty with respect to $\mathbf{v}$.*

The theorem (see Figure 1 for intuition behind the proof) implies that FGM is not a fixed attribute of a function that can be estimated from the data, but instead should be viewed as a property of the input's reparameterization. Furthermore, different reparameterizations feature different FGMs with varied levels of decomposability, some of which may not significantly simplify the target function. This motivates a reverse approach that begins from defining a desired FGM and learning representations of the input that align with the graph. To examine if FGM decomposition implemented this way can still bring benefits in MBO, we conduct another experiment in the right column of Figure 2. This time, we conceal the FGM information from both models, while implementing one of them with another decomposition, in a latent space that is computed with an attention layer (for fairness, we gave the FGM-oblivious model an attention layer too). Despite not bringing major improvements to the quality of fit, the decomposable model does lead to better designs, providing empirical support for the considered approach. Consequently, in the next subsection, we introduce Cliqueformer, where the FGM is specified as a hyperparameter of the model and a representation of the data that follows its structure is learned.

Architecture

The goal of this subsection is to derive an MBO model that can simultaneously learn the target function as well as its structure, and thus be readily applied to MBO. First, we would like the model to decompose its prediction into a sum of models defined over small subsets of the input variables in the manner of Equation (3). As discussed in the previous subsection, in general settings, while such a structure is not known a priori, a sufficiently expressive model can learn to follow a pre-defined structure. Thus, instead, we propose that the FGM be defined first, and a representation of the data be learned to align with the chosen structure.

Desideratum 1. *The model should follow a pre-defined FGM decomposition in a learned space.*

Accomplishing this desideratum is simple. A model that we train should, after some transformations, split the input's

representations, denoted as $\mathbf{z}$, into partially-overlapping cliques, $(\mathbf{z}_{C_1}, \dots, \mathbf{z}_{C_{N_{\text{clique}}}})$, and process them independently, followed by a summation. To implement this, we specify how many cliques we want to decompose the function into, N_{clique}, as well as their dimensionality, d_{clique}, and the size of their *knots*—dimensions at which two consecutive cliques overlap, $d_{\text{knot}} = |\mathbf{z}_{C_i} \cap \mathbf{z}_{C_{i+1}}|$. Then, we pass each clique through an MLP network that is also equipped with a trigonometric clique embedding, similar to Vaswani et al. (2017) to express a different function for each clique,

$$f_\theta(\mathbf{z}) = \tfrac{1}{N_{\text{clique}}} \textstyle\sum_{i=1}^{N_{\text{clique}}} f_\theta(\mathbf{z}_{C_i}, \boldsymbol{c}_i),$$
$$\text{where } \boldsymbol{c}_{i,2j}, \boldsymbol{c}_{i,2j+1} = \sin(i \cdot \omega_j), \cos(i \cdot \omega_j)$$

and $\omega_j = 10^{-8j/d_{\text{model}}}$. Here, we use the arithmetic mean over cliques rather than summation because, while being functionally equivalent, it provides more stability when $N_{\text{clique}} \to \infty$. Pre-defining the cliques over the representations allows us to avoid the problem of discovering arbitrarily dense graphs, as explained in Subsection 3. This architectural choice implies that the regret from Theorem 1 will depend on the coverage of such representations' cliques, $\max_{i \in [N_{\text{clique}}]} 1/e_\theta(\mathbf{z}_{C_i})$, where $e_\theta(\mathbf{z}) = \mathbb{E}_{\mathbf{x} \sim \mathcal{D}}[e_\theta(\mathbf{z}|\mathbf{x})]$ is the marginal distribution of the representations learned by an encoder e_θ. This term can become dangerously large if individual distributions $e_\theta(\mathbf{z}_C)$ put disproportionally more density to some regions of the latent space than to others. Thus, to prevent that, we propose to train the latent space so that the distribution of cliques attain broad coverage[2].

Desideratum 2. *Individual cliques of the learned representations should attain broad coverage.*

To meet this requirement, we leverage tools from representation learning, but in a novel way. Namely, we put a variational bottleneck (Kingma and Welling 2013; Higgins et al. 2016; Alemi et al. 2016, VIB) on **individual** cliques of our representations that brings their distribution closer to a prior with wide coverage, which we choose to be the standard-normal prior. To implement it, when computing the loss for a single example, we sample a single clique to compute the VIB for at random, as opposed to computing it for the joint latent variable like in the classical VIB,

$$\text{VIB}(\mathbf{x}, \theta) = \mathbb{E}_{i \sim U[N_{\text{clique}}]}\big[\text{KL}\big(e_\theta(\mathbf{z}_{C_i}|\mathbf{x}), p_{C_i}(\mathbf{z}_{C_i})\big)\big],$$

where $e_\theta(\mathbf{z}_C|\mathbf{x})$ is the density of clique C produced by our learnable encoder $e_\theta(\mathbf{z}|\mathbf{x})$, and $p_C(\mathbf{z}_C)$ is the density of $\mathbf{z}_C$ under the standard-normal distrituion. Note that it is not equivalent to the classical, down-weighted VIB in expectation either since our cliques overlap, meaning that knots contribute to the VIB more often than regular dimensions. We ablate for the impact of this term in Appendix C.

Lastly, since in practical problems, such as biological and chemical design, the data is often discrete (Uehara et al. 2024a; Yang et al. 2024), the designs cannot always be readily optimized with gradient-based methods. Thus, to be generally applicable, our model should be equipped with a data type-agnostic, approximately invertible map from the data space to a continuous space, where gradient-based optimization is feasible. This brings us to our last desideratum.

[2]For a probability distribution of z to attain a *broad coverage* means to distribute its probability mass on a large set of values of z, as opposed to be tightly concentrated around its modes.

Algorithm 1: MBO with Cliquefrormer

1: Initialize the encoder, decoder, and predictive model $(e_\theta, d_\theta, f_\theta)$.
2: **for** $t = 1, \dots, T_{\text{model}}$ **do**
3: Take a gradient step on the parameter θ with respect to $L_{\text{clique}}(\theta)$ from Equation (4).
4: **end for**
5: Sample B examples $\mathbf{x}^{(i_b)} \sim \mathcal{D}$ from the dataset.
6: Encode the examples $\mathbf{z}^{(i_b)} \sim e_\theta(\mathbf{z}|\mathbf{x}^{(i_b)})$,.
7: **for** $t = 1, \dots, T_{\text{design}}$ **do**
8: Decay the representations, $\mathbf{z}^{(i_b)} \leftarrow (1 - \lambda)\mathbf{z}^{(i_b)}$.
9: Take a gradient ascent step on the parameter $\mathbf{z}$ with respect to L_{mbo} from Equation (5).
10: **end for**
11: Propose solution candidates by decoding the representations, $\mathbf{x}^\star \sim d_\theta(\mathbf{x}|\mathbf{z}^\star)$.

Desideratum 3. *The model should have a decoder that maps representations back to the original data space.*

Thus, together with the model $f_\theta(\mathbf{z})$ and the encoder $e_\theta(\mathbf{z}|\mathbf{x})$, we train a decoder $d_\theta(\mathbf{x}|\mathbf{z})$ that reconstructs the designs from the latent variables. Putting it all together, the training objective of our model is a VIB-style likelihood objective with a regression term,

$$L_{\text{clique}}(\theta) = \mathbb{E}_{(\mathbf{x},\text{y}) \sim \mathcal{D}, \mathbf{z} \sim e_\theta(\cdot|\mathbf{x})}\Big[\text{VIB}(\mathbf{x}, \theta) - \log d_\theta(\mathbf{x}|\mathbf{z}) + \tau \cdot \big(\text{y} - f_\theta(\mathbf{z})\big)^2\Big], \quad (4)$$

where τ is a positive coefficient that we set to 10.

We use transformer networks (Vaswani et al. 2017) for both encoder e_θ and decoder d_θ to achieve the expressivity needed for FGM decomposition in latent space. The encoder transforms input $\mathbf{x} \in \mathbb{R}^d$ into d vectors of size d_{model}, processes them as token embeddings, and outputs a normal distribution $e_\theta(\mathbf{z}|\mathbf{x})$. Sampled representations $\mathbf{z}$ are arranged into N_{clique} cliques (dimension d_{clique}, knot size d_{knot}) before being passed to both the predictive model f_θ and transformer decoder d_θ. For illustration of the information flow in Cliqueformer's training consult Figure 3.

Optimizing Designs With Cliqueformer

Once Cliqueformer is trained, we use it to optimize new designs. Typically, MBO methods initialize this step at a sample of designs $(\mathbf{x}^{i_b})_{b=1}^B$ drawn from the dataset (Trabucco et al. 2021). Since in our algorithm the optimization takes place in the latent space $\mathcal{Z}$, we perform this step by encoding the sample of designs with Cliqueformer's encoder, $\mathbf{z}^{i_b} \sim e_\theta(\mathbf{z}|\mathbf{x}^{i_b})$. We then optimize the representation $\mathbf{z}^{i_b}$ of design $\mathbf{x}^{i_b}$ to maximize our model's value,

$$L_{\text{mbo}}\big((\mathbf{z}^{i_b})_{b=1}^B\big) = \frac{1}{B} \sum_{b=1}^{B} f_\theta(\mathbf{z}^{i_b}), \quad (5)$$

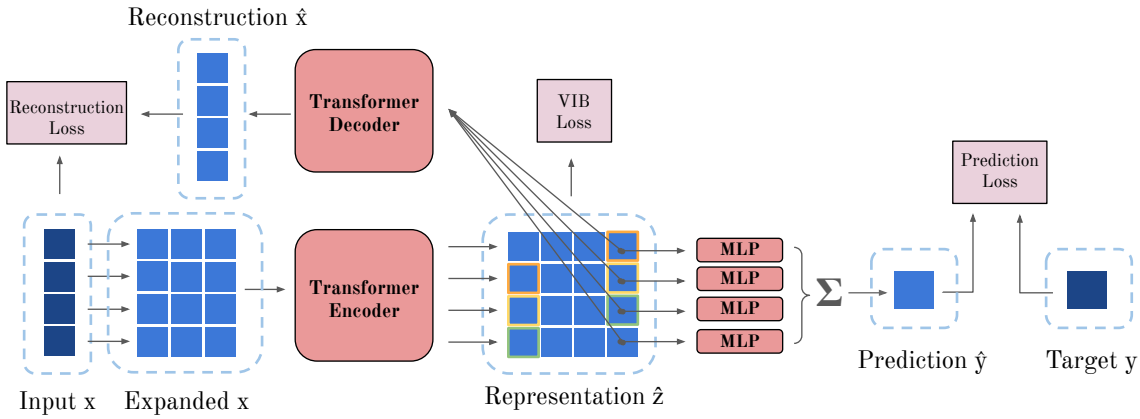


Figure 3: Illustration of information flow in Cliqueformer's training. Data are shown in navy, learnable variables in blue, neural modules in red, and loss functions in pink. The variables in input $\mathbf{x}$ are expanded into high-dimensional vectors (Expanded $\mathbf{x}$) and, treated as a sequence of embeddings, passed to a transformer encoder to compute representation $\mathbf{z}$ which is decomposed into cliques. The representation goes to the parallel MLPs whose outputs, added together, predict target $\mathbf{y}$. The representation $\mathbf{z}$ is also fed to a transformer decoder that tries to recover the original input $\mathbf{x}$. Additionally, the representation goes through an information bottleneck during training.

at the same time minding the denominator of the regret bound from Theorem 1. That is, we don't want the optimizer to explore regions under which the marginal densities $e_\theta(\mathbf{z}_C) = \mathbb{E}_{\mathbf{x}\sim\mathcal{D}}[e_\theta(\mathbf{z}_C|\mathbf{x})]$ are small. Fortunately, since the encoder was trained with standard-normal prior on the cliques, $p(\mathbf{z}_C) = N(0_{d_{\text{clique}}}, I_{d_{\text{clique}}})$, we know that values of $\mathbf{z}$ closer to the origin have unilaterally higher marginals. This simple property of standard-normal distribution allows us to confine the optimizer's exploration to designs with in-distribution cliques by exponentially decaying the design at every optimization step. Thus, we use AdamW as our optimizer (Loshchilov, Hutter et al. 2017) (see Appendix C for an ablation study). We provide the pseudocode of the whole procedure of designign with Cliqueformer in Algorithm 1.

4 Related Work

The idea of using machine learning models in optimization problems has existed for a long time, and has been mainly cultivated in the literature on Bayesian optimization (Williams and Rasmussen 2006; Snoek, Larochelle, and Adams 2012, BO). The BO paradigm relies on two core assumptions: availability of data of examples paired with their target function values, as well as access to an oracle that allows a learning algorithm to query values of proposed examples. Thus, similarly to reinforcement learning, the challenge of BO is to balance exploitation and exploration of the black-box function modeled by a Gaussian process.

Recently, to help BO tackle very high-dimensional problems, techniques of decomposing the target function have become more popular (Kandasamy, Schneider, and Póczos 2015; Rolland et al. 2018). Most commonly, these methods decompose the target function into functions defined on the input's partitions. While such models are likely to deviate far from the functions' ground-truth structure, one can derive theoretical guarantees for a range of such decompositions under the BO's query budget assumptions (Ziomek and Bou-Ammar 2023). However, these results do not apply to our setting of offline MBO, where no additional queries are available and the immediate reliability on the model is essential. Furthermore, instead of partitioning the input, we decompose our prediction over a latent variable that is learned by a transformer, enabling the model to acquire an expressive structure over which the decomposition is valid.

Offline model-based optimization (MBO) has gained traction in domains where BO assumptions cannot be met, offering solution generation from static datasets without additional queries. Early applications in molecule design (Gómez-Bombarelli et al. 2018) utilized variational autoencoders (Kingma and Welling 2013, VAE) to learn continuous molecular representations. While this demonstrated deep learning's potential, it didn't explore the target function's structural properties. The Conditioning by Adaptive Sampling (CBaS) algorithm (Brookes, Park, and Listgarten 2019) introduced a data type-agnostic approach for refining designs using non-differentiable oracle predictions. While this can be combined with trainable models through *autofocusing* (Fannjiang and Listgarten 2020), our setting differs by assuming a differentiable neural network model of the black-box function, enabling optimization through automatic differentiation (Paszke et al. 2019). Trabucco et al. (2021) introduced a neural network-based method, exactly for our setting, dubbed Conservative Objective Models (COMs), where a surrogate model is trained to both predict values of examples that can be found in the dataset, and penalize those that are not. COMs differs from our work fundamentally, since its contribution lies in the formulation of the conservative regularizer applied to arbitrary neural networks, while we focus on scalable model architectures that facilitate computational design. It is worth noting, however, that our Algorithm 1, similarly to COMs's conservative reg-

Task	Grad.Asc.	RWR	IOM	COMs	DDOM	MatchOpt	Transformer	Cliqueformer
Lat. RBF 11	0.00	0.08	0.00	**0.66**	0.00	0.56	0.47	0.65
Lat. RBF 31	0.00	0.31	0.00	0.50	0.00	**0.66**	0.00	0.64
Lat. RBF 41	0.00	0.35	0.00	0.45	0.00	0.32	0.20	**0.66**
Lat. RBF 61	0.00	0.29	0.00	0.25	0.00	**0.74**	0.16	0.66
Superconductor	1.13	1.03	1.03	0.97	1.22	0.84	0.96	**1.43**
TF-Bind-8	0.99	**1.58**	0.99	1.57	1.55	1.47	1.48	**1.58**
DNA HEPG2	**2.16**	1.91	1.97	1.20	1.82	1.72	2.13	2.10
DNA k562	2.11	1.91	2.62	1.80	2.61	2.25	2.60	**3.15**
Ave.score ↑	0.80	0.93	0.83	0.93	0.90	1.07	1.00	**1.36**
Ave.rank ↓	5.00	4.00	5.00	4.38	4.75	4.38	4.63	**1.63**

Table 1: Comparison of Cliqueformer and the baselines. Each score is the empirical estimate of the expected top 1% of scores, estimated by averaging the top-10 values out of 1000 generated designs, averaged over 5 runs. Scores are min-max normalized using the min and max from the dataset, ensuring that dataset designs fall in $[0, 1]$. Unlike (Trabucco et al. 2022), we do not use test data for normalization, improving interpretability (see Appendix D). We report both average score (higher is better) and average rank (lower is better).

ularizer, does constrain exploration of the design space, but it does so implicitly through weight decay. A bit more similarly to us, Invariant Objective Model (IOM) of Qi et al. (2022) also trains a model with a latent representation. Their focus, however, is to make the representation distribution-invariant via GANs (Goodfellow et al. 2014), while we focus on our network's decomposition abilities.

Another recent line of work proposes to tackle the design problem through means of generative modeling. BONET (Mashkaria, Krishnamoorthy, and Grover 2023), DDOM (Krishnamoorthy, Mashkaria, and Grover 2023) are examples of works that bring the most recent novelties of the field to address design tasks. BONET does so by training a transformer to generate sequences of designs that monotonically improve in their value, DDOM by training a value-conditioned diffusion model. That is, these methods attempt to generate high-value designs through novel conditional generation mechanisms. Instead, we model MBO as a maximization problem, and propose a scalable model that acquires the structure of the black-box function through standard gradient-based learning. To this end, we bring powerful techniques from generative modeling, like transformers (Vaswani et al. 2017) and variational-information bottlenecks (Kingma and Welling 2013; Alemi et al. 2016).

The work of Grudzien et al. (2024) introduced the theoretical foundations of functional graphical models (FGMs), including Theorem 1. However, as we have shown in Theorem 2, their graph discovery heuristic does not solve the problem of learning the black-box function's structure. On the other hand, we solved it by its integration of a pre-defined FGM into the architecture of our *Cliqueformer*. As such, the model learns the structure of the target function, as well as learns to predict its value, in synergy within an end-to-end training.

5 Experiments

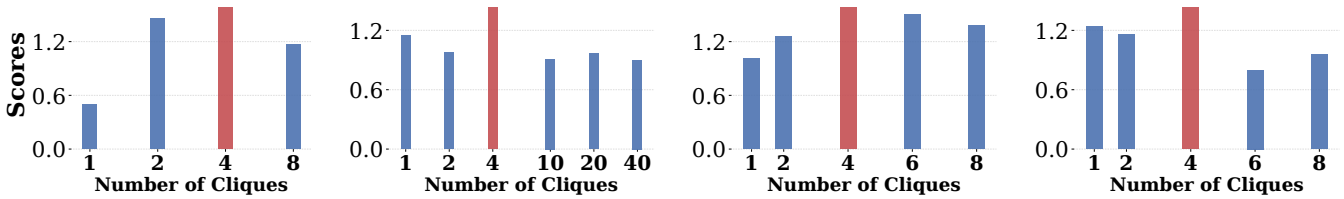
In this section, we provide the empirical evaluation of Cliqueformer. We begin by benchmarking Cliqueformer against prior methods on tasks from the MBO literature. We then evaluate the effect of the novel FGM decomposition layer in Cliqueformer through an ablation study.

Benchmarking

We compare our model to five classes of algorithms, each represented by proven prior methods. As a *naïve* baseline, we employ gradient ascent on a learned model (*Grad. Asc.*). To compare to *exploratory* methods, we use Reward-Weighted Regression (Peters and Schaal 2007, *RWR*) which learns by regressing the policy against its most promising perturbations. To represent *conservative* algorithms, we compare to Conservative Objective Models (Trabucco et al. 2021; Kumar et al. 2021, *COMs*) and Invariant Objective Model (Qi et al. 2022, *IOM*). As a *conditional generative modeling* in MBO baseline, we use Denoising Diffusion Optimization Models (Krishnamoorthy, Mashkaria, and Grover 2023, DDOM), as well as recent MatchOpt (Hoang et al. 2025), which are inspired by generative diffusion models (Ho, Jain, and Abbeel 2020). Additionally, to evaluate the importance of the novel elements of our architecture and training, we evaluate gradient ascent with the transformer backbone (Vaswani et al. 2017, *Transformer*). We use standard hyperparameters for the baselines. While Cliqueformer can be tuned for each task, we keep most of the hyperparameters the same—see Appendix D for more details.

Following prior work (Trabucco et al. 2022), we normalize scores to $[0, 1]$ based on the dataset's range and report the empirical top 1%, estimated by averaging the top 10 designs from 1000 candidates across 5 seeds. Below, we describe benchmarks that we use for our experiments.

Latent Radial-Basis Functions (Lat. RBF). These synthetic tasks, introduced by Grudzien et al. (2024), stress-test MBO methods. Each datapoint is generated by sampling $\mathbf{z} \sim \mathcal{N}(0, I_{d_z})$ for $d_z \in 11, 31, 41, 61$, applying RBFs over d_C-dimensional cliques of a fixed FGM to get y, and mapping $\mathbf{z}$ through a nonlinear transformation $\mathbf{x} = T(\mathbf{z}) \in \mathcal{T} \subset \mathbb{R}^d$ with $d > d_z$. Only $\mathbf{x}$ and y are observed. Validity of



(a) TFBind-8 (d_z fixed) (b) Supercond. (d_z fixed) (c) TFBind-8 (d_{clique} fixed) (d) Supercond. (d_{clique} fixed)

Figure 4: Ablation experiments on the number of cliques N_{clique} used in the FGM decomposition of Cliqueformer (the y-axis shows the design score and the x-axis shows the number of cliques used). In Figures (a) & (b), for each task, we fixed the size d_z of the latent variable $\mathbf{z}$ and varied the number of cliques N_{clique}. In Figures (c) & (d), for each task, we fixed the clique size d_{clique}, and varied the number of cliques N_{clique}. In every figure, the red bar corresponds to the best (maximal) score.

a new design $\hat{\mathbf{x}}$ is judged by whether $\hat{\mathbf{x}}$ lies on the manifold $\mathcal{T}$, i.e., whether $T^{-1}(\hat{\mathbf{x}})$ exists. In our experiments, invalid designs score 0. These tasks also test a model's ability to leverage RBF structure which becomes increasingly important as the task's dimension grows. As Table 1 shows, Cliqueformer sustains strong performance as predicted by Theorem 1, whereas some baselines, like COMs, decline in higher dimensions.

Superconductor. This task involves designing an 81-dimensional material to maximize its critical temperature (Hamidieh 2018; Fannjiang and Listgarten 2020; Trabucco et al. 2022), thus evaluating MBO methods in real-valued, continuous domains. Here, greedy optimization (e.g., gradient ascent) is highly effective. Though Cliqueformer is designed to be robust to distribution shift, its ability to "stitch" in-distribution cliques enables substantial gains. It outperforms all baselines, particularly COMs——limited by its conservative updates——and gradient ascent (w/ and wo/ Transformer), that lacks Cliqueformer's decomposition strategy.

TFBind-8 & DNA Enhancers. These tasks optimize discrete DNA sequences of lengths 8 and 200, respectively (Trabucco et al. 2022; Uehara et al. 2024b). TFBind-8 measures binding affinity for a transcription factor, while DNA Enhancers target HEPG2 and k562 expression levels. TFBind-8, though simple, demonstrates Cliqueformer's ability to solve discrete tasks. For DNA Enhancers——large-scale tasks with $\sim 2\times10^5$ samples——Cliqueformer matches the strong performance of gradient ascent on HEPG2 and significantly outperforms all baselines on k562, showing its scalability. Averaged across all tasks, Cliqueformer achieves the highest overall performance.

Ablations

While some components of Cliqueformer, such as transformer blocks (Vaswani et al. 2017) and variational-information bottlenecks (Kingma and Welling 2013; Alemi et al. 2016), are well-known deep learning tools, the decomposing mechanism of our model is novel. In Figure (4), we verify the utility of the decomposing mechanism with an ablation study, in which we sweep over the number of cliques N_{clique} of Cliqueformer in the discrete TFBind-8 and continuous Superconductor tasks. In each of the tasks, we fix the size d_z of the latent variable $\mathbf{z}$ so that it approximately matches the dimensionality of the data, and sweep over the number of cliques into which it can be decomposed under the formula $d_z = d_{\text{knot}} + N_{\text{clique}}(d_{\text{clique}} - d_{\text{knot}})$. We cover the case $N_{\text{clique}} = 1$ to compare Cliqueformer to an FGM-oblivious VAE with transformer backbone.

Results in Figure (4) demonstrate the essential role of FGM decomposition in achieving superior performance across both tasks. The results reveal a clear trade-off in parameter selection: with fixed d_z, too few cliques (particularly $N_{\text{clique}} = 1$) prevent effective function decomposition, while too many cliques oversimplify the target function. We further investigated the impact of varying the number of cliques while maintaining fixed clique sizes ($d_{\text{clique}} = 3$ for TFBind-8 and $d_{\text{clique}} = 21$ for Superconductor). Figure (4) also confirms that the optimal N_{clique} values remain unchanged, likely because deviating from these values results in latent spaces that are either too constrained or too expansive relative to the data dimensionality.

In our experiments, we consistently obtained good performance by setting the clique size to $d_{\text{clique}} = 3$ and choosing the number of cliques so that the total latent dimension approximately matches that of the design. In DNA Enhancers, we doubled the clique size and halved the number of cliques to decrease the computational cost of attention layers. We offer more ablation studies in Appendix C, and we include a detailed report of hyperparameters in Appendix D. An in-depth analysis of the relation between hyperparameters and the performance is an exciting avenue of future work.

6 Conclusion

In this work, we introduced Cliqueformer, a scalable neural architecture for offline model-based optimization that leverages functional graphical models to learn the structure of black-box target functions. By incorporating recent theoretical advances in MBO, our model eliminates the need for explicit conservative regularization or iterative retraining when proposing designs. Cliqueformer achieves and maintains strong performance across all evaluation tasks, paving the way for research into scaling MBO with large neural networks and expanded design datasets.

References

Alemi, A. A.; Fischer, I.; Dillon, J. V.; and Murphy, K. 2016. Deep variational information bottleneck. *arXiv preprint arXiv:1612.00410*.

Brochu, E.; Cora, V. M.; and De Freitas, N. 2010. A tutorial on Bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning. *arXiv preprint arXiv:1012.2599*.

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

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

Geng, X. 2023. *Offline Data-Driven Optimization: Benchmarks, Algorithms and Applications*. University of California, Berkeley.

Gómez-Bombarelli, R.; Wei, J. N.; Duvenaud, D.; Hernández-Lobato, J. M.; Sánchez-Lengeling, B.; Sheberla, D.; Aguilera-Iparraguirre, J.; Hirzel, T. D.; Adams, R. P.; and Aspuru-Guzik, A. 2018. Automatic chemical design using a data-driven continuous representation of molecules. *ACS central science*, 4(2): 268–276.

Goodfellow, I.; Pouget-Abadie, J.; Mirza, M.; Xu, B.; Warde-Farley, D.; Ozair, S.; Courville, A.; and Bengio, Y. 2014. Generative adversarial nets. *Advances in neural information processing systems*, 27.

Grudzien, K.; Uehara, M.; Levine, S.; and Abbeel, P. 2024. Functional Graphical Models: Structure Enables Offline Data-Driven Optimization. In *International Conference on Artificial Intelligence and Statistics*, 2449–2457. PMLR.

Hamidieh, K. 2018. A data-driven statistical model for predicting the critical temperature of a superconductor. *Computational Materials Science*, 154: 346–354.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 770–778.

Higgins, I.; Matthey, L.; Pal, A.; Burgess, C.; Glorot, X.; Botvinick, M.; Mohamed, S.; and Lerchner, A. 2016. beta-vae: Learning basic visual concepts with a constrained variational framework.

Ho, J.; Jain, A.; and Abbeel, P. 2020. Denoising diffusion probabilistic models. *Advances in Neural Information Processing Systems*, 33: 6840–6851.

Hoang, M.; Fadhel, A.; Deshwal, A.; Doppa, J. R.; and Hoang, T. N. 2025. Learning surrogates for offline black-box optimization via gradient matching. *arXiv preprint arXiv:2503.01883*.

Hochreiter, S.; and Schmidhuber, J. 1997. Long short-term memory. *Neural computation*, 9(8): 1735–1780.

Kandasamy, K.; Schneider, J.; and Póczos, B. 2015. High dimensional Bayesian optimisation and bandits via additive models. In *International conference on machine learning*, 295–304. PMLR.

Kim, S.; Thiessen, P. A.; Bolton, E. E.; Chen, J.; Fu, G.; Gindulyte, A.; Han, L.; He, J.; He, S.; Shoemaker, B. A.; et al. 2016. PubChem substance and compound databases. *Nucleic acids research*, 44(D1): D1202–D1213.

Kingma, D. P.; and Welling, M. 2013. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*.

Krishnamoorthy, S.; Mashkaria, S. M.; and Grover, A. 2023. Diffusion Models for Black-Box Optimization. In Krause, A.; Brunskill, E.; Cho, K.; Engelhardt, B.; Sabato, S.; and Scarlett, J., eds., *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, 17842–17857. PMLR.

Krizhevsky, A.; Sutskever, I.; and Hinton, G. E. 2017. Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, 60(6): 84–90.

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

Kumar, A.; Yazdanbakhsh, A.; Hashemi, M.; Swersky, K.; and Levine, S. 2021. Data-driven offline optimization for architecting hardware accelerators. *arXiv preprint arXiv:2110.11346*.

Kumar, A.; Zhou, A.; Tucker, G.; and Levine, S. 2020. Conservative q-learning for offline reinforcement learning. *Advances in Neural Information Processing Systems*, 33: 1179–1191.

Loshchilov, I.; Hutter, F.; et al. 2017. Fixing weight decay regularization in adam. *arXiv preprint arXiv:1711.05101*, 5.

Mashkaria, S. M.; Krishnamoorthy, S.; and Grover, A. 2023. Generative pretraining for black-box optimization. In *International Conference on Machine Learning*, 24173–24197. PMLR.

Paszke, A.; Gross, S.; Massa, F.; Lerer, A.; Bradbury, J.; Chanan, G.; Killeen, T.; Lin, Z.; Gimelshein, N.; Antiga, L.; et al. 2019. PyTorch: An Imperative Style, High-Performance Deep Learning Library. *Advances in Neural Information Processing Systems*, 32: 8026–8037.

Peters, J.; and Schaal, S. 2007. Reinforcement learning by reward-weighted regression for operational space control. In *Proceedings of the 24th international conference on Machine learning*, 745–750.

Qi, H.; Su, Y.; Kumar, A.; and Levine, S. 2022. Data-Driven Offline Decision-Making via Invariant Representation Learning. *arXiv preprint arXiv:2211.11349*.

Rolland, P.; Scarlett, J.; Bogunovic, I.; and Cevher, V. 2018. High-dimensional Bayesian optimization via additive models with overlapping groups. In *International conference on artificial intelligence and statistics*, 298–307. PMLR.

Snoek, J.; Larochelle, H.; and Adams, R. P. 2012. Practical bayesian optimization of machine learning algorithms. *Advances in neural information processing systems*, 25.

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

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

Uehara, M.; Zhao, Y.; Biancalani, T.; and Levine, S. 2024a. Understanding Reinforcement Learning-Based Fine-Tuning of Diffusion Models: A Tutorial and Review. *arXiv preprint arXiv:2407.13734*.

Uehara, M.; Zhao, Y.; Hajiramezanali, E.; Scalia, G.; Eraslan, G.; Lal, A.; Levine, S.; and Biancalani, T. 2024b. Bridging Model-Based Optimization and Generative Modeling via Conservative Fine-Tuning of Diffusion Models. *arXiv preprint arXiv:2405.19673*.

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, Ł.; and Polosukhin, I. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.

Williams, C. K.; and Rasmussen, C. E. 2006. *Gaussian processes for machine learning*, volume 2. MIT press Cambridge, MA.

Yang, S.; Batzner, S.; Gao, R.; Aykol, M.; Gaunt, A. L.; McMorrow, B.; Rezende, D. J.; Schuurmans, D.; Mordatch, I.; and Cubuk, E. D. 2024. Generative Hierarchical Materials Search. *arXiv preprint arXiv:2409.06762*.

Ziomek, J.; and Bou-Ammar, H. 2023. Are Random Decompositions all we need in High Dimensional Bayesian Optimisation? *arXiv preprint arXiv:2301.12844*.