LONDON'S GLOBAL UNIVERSITY
Department of Computer Science

GP-MOBO: Multi-Objective Bayesian Optimization with Independent Tanimoto Kernel Gaussian Processes for Diverse Pareto Front Exploration

Anabel Yong (ID: 23205123)
Supervisors: Professor Brooks Paige, UCL
& Dr. Layla-Hosseini Gerami @ IgnotaLabs.AI
Faculty of Engineering
Department of Computer Science

University College London
A Project Report Presented in Partial Fulfillment of the Degree
MSc Computational Statistics and Machine Learning

September 2024

Abstract

We present GP-MOBO, a novel multi-objective Bayesian Optimization algorithm that advances the state-of-the-art in molecular optimization. Our approach integrates a fast minimal package for Exact Gaussian Processes (GPs) capable of efficiently handling the full dimensionality of sparse molecular fingerprints without the need for extensive computational resources. GP-MOBO consistently outperforms traditional methods like GP-BO by fully leveraging fingerprint dimensionality, leading to the identification of higher-quality and valid SMILES. Moreover, our model achieves a broader exploration of the chemical search space, as demonstrated by its superior proximity to the Pareto front in all tested scenarios. Empirical results from the DockSTRING dataset reveal that GP-MOBO yields higher geometric mean values across 20 Bayesian optimization iterations, underscoring its effectiveness and efficiency in addressing complex multi-objective optimization challenges with minimal computational overhead.

Keywords— Gaussian Processes - Multi-objective Bayesian Optimization - Molecular Cheminformatics - Expected Hypervolume Improvement - Hypervolume Indicator - Multi-output Gaussian Process Regression - Virtual Screening

Acknowledgements

I want to thank my supervisors, Layla-Hosseini Gerami and Brooks Paige for their academic support and pushing me towards academic excellence, and the highest quality of guidance. Thank you Brooks for giving me the ideas for our novel GP-MOBO model. I want to thank Layla for the great emotional, and chemistry support, the Ignota team, and finally, for introducing me to Austin Tripp. My appreciation for Austin Tripp, my external supervisor, spans from his extreme patience regarding this project to his immense knowledge in the realm of drug discovery, Gaussian Processes, and advanced topics in kernel methods. Thank you for making me code everything from scratch. This would not have been a feasible project without Austin. Thank you to the 3 of you for pushing me to my academic limits and being the best supervisors I could ask for during my time at UCL.

Finally, I dedicate this to Jonathan, who showed me I was capable of doing difficult projects, and for relentlessly believing and supporting me throughout this whole journey this year - even when I felt like giving up. Thank you for being there for me when I am not capable of taking care of myself and making me ramen all the time. Additionally, I want to thank my parents for giving me this opportunity to pursue my degree and providing unconditional support and love despite being so far away.

Declaration

I, Anabel Yong, I declare that the thesis has been composed by myself and that the work has not be submitted for any other degree or professional qualification. I confirm that the work submitted is my own, except where work which has formed part of jointly-authored publications has been included and referenced. The report may be freely copied and distributed provided the source is explicitly acknowledged.

Anabel Yong	09/09/24
Signature	*Date*

Table of Contents

List of Figures

List of Tables

List of Algorithms

List of Abbreviations

ECFP: Extended-Connectivity Fingerprints

EHVI: Expected Hypervolume Improvement

EI-PT: Expected Improvement - PyTorch

EI: Expected Improvement

GP-MOBO: Gaussian Process Multi-Objective Bayesian Optimization

GP: Gaussian Processes

GPR: Gaussian Process Regression

HV: Hypervolume

KERN-GP-EHVI: Custom Kernel-Gaussian Process-Expected Hypervolume Improvement

KERN-GP-EI: Custom Kernel-Gaussian Process-Expected Improvement

MCS: Monte Carlo Sampling

MO-TKGP: Multi-output Tanimoto Kernel Gaussian Processes

MOBO: Multi-Objective Bayesian Optimization

MOGP: Multi-output Gaussian Processes

MPO: Multi-Property Objectives

NLML: Negative Log Marginal Likelihood

RBF: Radial Basis Function

RKHS: Reproducing Kernel Hilbert Space

RS: Random Sampling

SOBO: Single-Objective Bayesian Optimization

SVMs: Support Vector Machines

TDC: Therapeutics Data Common

UCB-PT: Upper Confidence Bound - PyTorch

1 | Introduction

1.1 Machine Learning Models for Molecular Optimization

In recent years, machine learning models have gained significant traction in molecular optimization, in the realm of drug discovery. These models offer potential solutions for navigating vast chemical spaces to identify molecules with desirable properties such as high efficacy and low toxicity. Molecular optimization is typically framed as an optimization task over a molecular space $\mathcal{M}$, with the goal of balancing multiple competing objectives - such as pharmacokinetic and pharmacodynamic properties - in real-world applications. The complexity of these problems grows substantially when multiple objectives must be optimized simultaneously[1][2].

Most existing optimization frameworks, particularly those that employ Bayesian Optimization (BO) have been designed for single-objective tasks. This traditional approach involves scalarizing multiple objectives into a single scalar function, which simplifies the optimization process but forces an implicit trade-off between objectives, even when the trade-offs may not be well understood or defined in advance. Real-world problems, however, require multi-objective optimization (MOO), which allows for simultaneous optimization across several criteria without pre-defining their trade-offs. In this thesis, we investigate the effectiveness of a simple multi-objective Bayesian Optimization (MOBO) approach using independent Gaussian Processes (GPs) and an Expected Hypervolume Improvement (EHVI) acquisition function. Despite the simplicity of this setup, it has not been thoroughly explored in the context of molecular optimization, and we hypothesize that it could yield competitive performance in comparison to more complex models.

1.1.1 Motivation for Multi-Objective Bayesian Optimization

Single-objective optimization frameworks dominate the literature, where multiple objectives are scalarized into a single composite score. This scalarization, however, introduces limitations, as it implicitly specifies a fixed trade-off between objectives. For example, optimizing for $a = 1, b = 2$ assumes this trade-off is preferable to $a = 2, b = 1$, yet in many practical cases, the preferred trade-off is not known a priori. In contrast, multi-objective Bayesian optimization (MOBO) seeks to approximate the Pareto frontier, where multiple trade-offs between objectives are discovered rather than being predefined.

Scalarized Bayesian Optimization(BO) models for molecular optimization have performed

adequately in some settings but they are ill-suited to complex, real-world tasks, particularly in drug discovery, where trade-offs between properties like efficacy and toxicity are often unknown. MOBO offers a more flexible and comprehensive solution by modeling each independently, allowing researchers to explore the trade-off space more freely.

Our motivation stems from this gap in the literature: many works focus on single-objective BO, while real-world problems are inherently multi-objective. We aim to demonstrate that a simpler MOBO approach, usign independent Gaussian Processes (GP) for each molecular objective, can be highly effective for molecular optimization. This research explores whether the MOBO acquisition function, called EHVI, which is well-established for finding Pareto-optimal solutions, combined with independent GPs, can outperform or at least match the performance of finding optimal chemical compounds shown in this benchmark paper by Gao et al(2022) [1].

1.2 Related Work

1.2.1 Molecular Optimization

Molecular optimization in drug discovery has seen significant progress through the integration of machine learning. Generative models, such as Variational Autoencoders (VAEs) [3][4], and Generative Adversarial Networks (GANs)[5], have demonstrated promise in proposing novel molecules. These VAEs [6][7][8][9][4][10] typically map molecular structures to a latent space and apply single-objective BO techniques, to find optimal solutions. Despite their potential, these models often require large datasets and suffer from high computational costs. Fitting a GP in the high-dimensional latent space of a VAE is challenging, as the latent space is often complex and not well-suited for direct GP application. The lack of smoothness and continuity in the VAE's latent space can further hinder the GP's ability to make accurate predictions, complicating the balance between exploration and exploitation in Bayesian Optimization (BO) tasks [11]. Moreover, modeling such latent spaces often requires a large number of samples, making the optimization process inefficient, especially for high-dimensional tasks [12].

Reinforcement learning frameworks like REINVENT[13] or GFlowNets[14], as well as genetic algorithms, have also been proposed to explore the molecular space. However, these methods face challenges in efficiently balancing exploration and exploitation, particularly when generating invalid SMILES strings or exploring sub-optimal regions of the chemical space. For example, genetic algorithms[15][16][17] rely on random mutations of known molecules, which can lead to inefficient search processes [2].

Gaussian Process Bayesian Optimization (GP-BO)[18] has emerged as a popular framework for molecular optimization. GP-BO excels at modeling uncertainty, making it suitable for applications with sparse or expensive-to-acquire data [18][19]. However, many of these models, including GP-BO, focus on single-objective optimization, using scalarization techniques to combine objectives into a single metric. While scalarization simplifies the optimization process, it pre-defines trade-offs between objectives, which may not be well-understood in real-world drug discovery scenarios. This limitation is particularly problematic in multi-objective optimization tasks where the goal is to find the Pareto frontier.

While many previous work have focused on large complex models, we demonstrate that a straightforward method, free from scalarization, can effectively handle multi-objective tasks, achieving competitive performance with current state-of-the-art GP-BO.

1.2.2 Multi-Objective Bayesian Optimization

Multi-objective Bayesian Optimization (MOBO) extends the principles of Bayesian optimization problems involving multiple conflicting objectives, aiming to efficiently approximate the Pareto front- the set of non-dominated solutions representing the best trade-offs among objectives [20]. Unlike single-objective optimization, MOBO does not require scalarization of objectives, avoiding the need to predefined trade-offs, which is particularly advantageous when these trade-offs are unknown or difficult to specify in advance.

Early approaches to MOBO often relied on scalarization techniques, such as weighted sums or utility functions, to combine multiple objectives into a single objective function[21]. However, these methods inherently require the specification of weights or parameters, which can bias the search towards certain regions of the Pareto front and may not capture true diversity of optimal solutions.

To overcome these limitations, researchers have developed acquisition functions specifically designed for multi-objective settings. One prominent example is the Expected Hypervolume Improvement (EHVI) acquisition function [22][23], which quantifies the expected increase in the hypervolume bounded by the current Pareto front and a reference point. EHVI guides the optimization process toward solutions that contribute most to improving the Pareto front, effectively balancing exploration and exploitation

Another notable acquisition function is the Pareto Expected Improvement (PEI), which extends the concept of Expected Improvement from single-objective optimization to multi-objective contexts[24]. PEI evaluates the expected improvement over the current

Pareto front, promoting diversity in the discovered solutions. Methods like Predictive Entropy Search for Multi-objective Optimization (PESMO)[25] and Multi-objective Upper Confidence Bound (MOUCB)[26] have also been proposed to efficiently navigate the trade-offs between objectives.

In terms of modeling approaches, both independent Gaussian Processes (GPs) for each objective and multi-output GPs that capture correlations between objectives have been employed [27][28]. While multi-output GPs can model inter-objective dependencies, they often come with increased computational complexity, especially in high-dimensional settings. Independent GPs offer a simpler alternative, with each GP modeling an individual objective, making them scalable and easier to implement.

MOBO has been successfully applied in various domains, including engineering design optimization[29], hyperparameter tuning in machine learning models [30], and materials science for discovering new compounds with desired properties [31]. Despite these advances, the application of MOBO in molecular optimization remains relatively underexplored. Most molecular optimization studies have focused on single-objective problems or have used scalarization methods when dealing with multiple objectives [13][31]. This gap suggests a missed opportunity to fully exploit the capabilities of MOBO in discovering diverse and Pareto-optimal molecules.

Recent efforts have started to bridge this gap. For example, Hernandez-Lobato et al.(2016)[32] proposed a general framework for constrained Bayesian optimization using information-based search strategies, which can be adapted to multi-objective scenarios. However, comprehensive studies that systematically apply MOBO techniques - particularly those utilizing simple and scalable models like independent GPs with EHVI - to molecular optimization tasks are still lacking.

1.3 Novelty Aspects and Contributions of this Paper

Our work aims to address this deficiency by investigating the effectiveness of a straightforward MOBO approach in molecular optimization. By employing independent GPs for each objective and leveraging the EHVI acquisition function, we seek to efficiently approximate the Pareto front without the need for complex modeling techniques or large training datasets. This approach not only simplifies the implementation but also has the potential to uncover a more diverse set of optimal molecules, better reflecting the multi-faceted objectives inherent in drug discovery and other chemical optimization problems.

Although Mehta et al(2022)[33] has acknowledged this issue and proposed a multi-

objective Bayesian optimization setup by multiplying single-objective acquisition function, the focus of their approach differs from ours. They coined their method the "multi-objective Bayesian optimization acquisition function", yet our approach integrates a more robust and interpretable framework to capture trade-offs. Additionally, the only other notable implementation of surrogate model-based MOBO for molecular optimization is from MIT Coley's Group with their MolPAL [34], which extends from a single-objective setup using message passing neural networks (MPNNs) as surrogate models. While their MOBO approach showed improvement over scalarization, it did not achieve competitive performance against single-objective MolPAL, ranking 13th in Gao's benchmark (see Table 5 of [1]). This highlights the need for more effective solutions like ours, which is benchmarked against Tripp et al.'s GP-BO (2021)[19], one of the best-performing model in molecular optimization.

In addition, we introduce `KERN-GP`, a kernel-only Gaussian Process package that enables the use of exact Tanimoto coefficients, retaining full molecular fingerprint dimensionality without the need for projection to lower dimensions. This allows us to better capture the intricacies of molecular structures, contributing to more accurate predictions and robust optimization results. We also incorporate MinMax Kernels to handle count fingerprints, providing enhanced flexibility and computational efficiency [2][35].

Our GP-MOBO framework is benchmarked against Tripp et al's GP-BO (2021)[19], the 4th best-performing model in molecular optimization[1]. By expanding GP-BO into a multi-objective framework, we demonstrate that even with a relatively simple setup, our model outperforms state-of-the-art methods in terms of Pareto front diversity and solution quality. This represents a significant improvement in the field of multi-objective molecular optimization, offering a scalable and interpretable alternative to more complex models.

The subsequent chapter will provide some cheminformatics preliminaries, theoretical background on Multi-Output Gaussian Processes, Tanimoto and MinMax Kernels, and multi-objective Bayesian optimization. This will be followed by a detailed methodology and experimental design to validate the performance of our GP-MOBO framework. Our results showcase the superiority of GP-MOBO over GP-BO, and the discussion will outline key takeaways and potential future directions to further enhance our approach.

2 | Background

2.1 GUACAMOL's Molecular Property Objectives (MPOs)

Multi-Property Objectives (MPOs) are a set of goals used in drug discovery to find compounds that satisfy several properties simultaneously. In this particular well-known benchmark dataset, GUACAMOL [15], MPOs are designed to replicate the complex criteria that real-world drug candidates must meet. These objectives often encompass a variety of properties such as chemical similarity, pharmacokinetic properties and structural constraints. In Table 2.1, some of these MPOs are as detailed below.

Table 2.1: Examples of Goal-Directed Benchmarks in GUACAMOL Dataset

Benchmark Name	**Scoring**	**Mean**	**Scoring Function(s)**	**Modifier**
Osimertinib MPO	top-1	geom	sim(osimertinib, FCFP4)	Thresholded(0.8)
	top-10		sim(osimertinib, ECFP6)	MinGaussian(0.85, 2)
	top-100		TPSA	MaxGaussian(100, 2)
			logP	MinGaussian(4, 2)
Fexofenadine MPO	top-1	geom	sim(fexofenadine, AP)	Thresholded(0.8)
	top-10		logP	MinGaussian(4, 2)
	top-100		TPSA	MaxGaussian(90, 2)
Ranolazine MPO	top-1	geom	sim(ranolazine, AP)	Thresholded(0.7)
	top-10		logP	MinGaussian(4, 2)
	top-100		TPSA	MaxGaussian(95, 20)
			number of fluorine atoms	Gaussian(1, 1)

The Osimertinib MPO is focused on finding molecules that are similar to Osimertinib (an anti-cancer drug) while meeting additional constraints such as logP and Topological Polar Surface Area (TPSA). The similarity to the target molecule is combined with other property scores using the geometric mean. These score modifiers shown in Table 2.1, details how each modifier influences the scoring function in multi-objective optimization problems. How these score modifiers evaluate molecules are mostly detailed in Brown's GUACAMOL paper [15].

All generative machine learning models, to date, utilize this scalarization by geometric mean optimization setup to make the single-objective optimization problem tractable [1]. However, this can mask trade-offs between objectives and makes it harder to achieve a truly balanced solution. Here, in this research, will be the first investigation into separating Guacamol MPO objectives and working with these scoring functions for our

algorithm. How did we do it without scalarizing the geometric mean? This is detailed in our Methodology and Experimental Design (Section 3.2) below.

2.2 SMILES

The Simplified Molecular-Input Line-Entry System (SMILES) offers a textual string format for encoding molecular structures, as introduced by Anderson et al.(1987)[36] and further developed by Weininger(1988)[37]. Examples of SMILES are illustrated in Figure 2.1.

Figure 2.1: Molecular Structure for SMILES string "CCCCCCC".

SMILES represents molecules as a sequence of atoms and bonds using short ASCII strings, where atoms are denoted by their chemical symbols. The advantage of using SMILES is its interpretability, allowing for direct integration with machine learning models. However, as acknowledged by Taleongpong and Paige (2024)[38], SMILES lack structural invariance, where the same molecule can be represented by different strings due to variations in atom ordering and molecular conformation.

2.3 Therapeutics Data Common Oracles

Oracles, provided by Therapeutics Data Commons (TDC)[39], are functions or models that evaluate specific molecular properties represented by SMILES strings. These predictive models are commonly used in drug discovery tasks to generate or optimize molecules for desirable properties such as high docking scores, low toxicity, or bioavailability.

```
from tdc import Oracle
CELECOXIB_ORACLE = Oracle("celecoxib-rediscovery")
```

```
SMILES = ['C1=C(C2=C(C=C10)OC(C(C2=O)=O)C3=CC=C(C(=C3)O)O)O']
ORACLE_SCORE = CELECOXIB_ORACLE(SMILES)
print(f"Oracle Score for C1=C(C2=C(C=C10)OC(C(C2=O)=O)C3=CC=C(C(=C3)O)O)O : {ORACLE_SCORE}")
```

```
Oracle Score for C1=C(C2=C(C=C10)OC(C(C2=O)=O)C3=CC=C(C(=C3)O)O)O : [0.1391304347826087]
```

Figure 2.2: **Use of an Oracle from Therapeutics Data Common(TDC) to evaluate SMILES corresponding to a molecular structure for its similarity to the celecoxib molecule:** The Oracle function is tailored for "celecoxib-rediscovery" task, which returns a numerical score that quantifies simlarity of molecule to desired properties of celecoxib.

The TDC package includes several oracles that evaluate physiochemical properties (e.g., logP, logD, QED, molecular weight) and toxicity (e.g., hERG inhibition). Many of these oracle functions, such as those used in Guacamol MPO tasks(Table 2.1), are readily available for use in molecular optimization efforts.

2.4 Molecular Fingerprints

Additionally, SMILES are also an integral part of generating molecular fingerprints. Molecular fingerprints represent chemical compounds as fixed-length vectors suitable for machine learning models[40]. These fingerprints capture the presence or absence of specific molecular substructures or properties, facilitating applications such as virtual screening, and similarity searching. Various types of molecular fingerprints exist, including structural key-based[41], path-based [42], circular [40] and pharmacophore [43] fingerprints. However, in this research, we will solely focus on Morgan fingerprinting [40], a circular fingerprinting technique. Among these, Morgan fingerprints have gained prominence due to their robustness and flexibility in capturing molecular features relevant to various cheminformatics tasks.

2.4.1 Morgan/ECFP Fingerprints

Extended Connectivity Fingerprints (ECFP)[40], represent a sophisticated method for encoding molecular structures. This approach assigns unique numeric identifiers to each atom in a molecule, which are iteratively updated based on the identifiers fingerprint's diameter, a parameter that defines the extent of atomic neighborhoods considered during

the fingerprinting process. For instance, ECFP6 fingerprints, which are widely used, involve three iterations of updating atom identifiers, corresponding to a diameter of six bonds in the molecular graph [40][44].

2.4.2 Generation of Molecular Fingerprints

An overview of the Morgan fingerprinting process is illustrated in Figure 2.3. Each atom in the molecule is initially assigned a numeric identifier based on its atomic properties, such as the number of neighboring atoms, atomic number, and the number of attached hydrogens. These initial identifiers are collected into a set representing the atom-centered environment. Subsequently, in the next step, each atom's identifier is updated by incorporating information from the identifiers of its immediate neighbors. This process continues iteratively, with each iteration expanding the scope of the atomic neighborhood considered. The result is a series of increasingly complex identifiers that encapsulate larger sub-structural features of the molecule. Once all iterations are complete, the algorithm identifies and removes duplicate identifiers, ensuring that each substructure is represented only once in the final fingerprint vector. This step prevents redundancy and ensures the compactness of the fingerprint. The final step involves hashing the unique identifiers to generate a fixed-length binary string (or bit vector). Each bit in this vector indicates the presence or absence of a specific substructure within the molecule. This binary representation is what constitutes the Morgan fingerprint, which can be used in various cheminformatics applications [45][46].

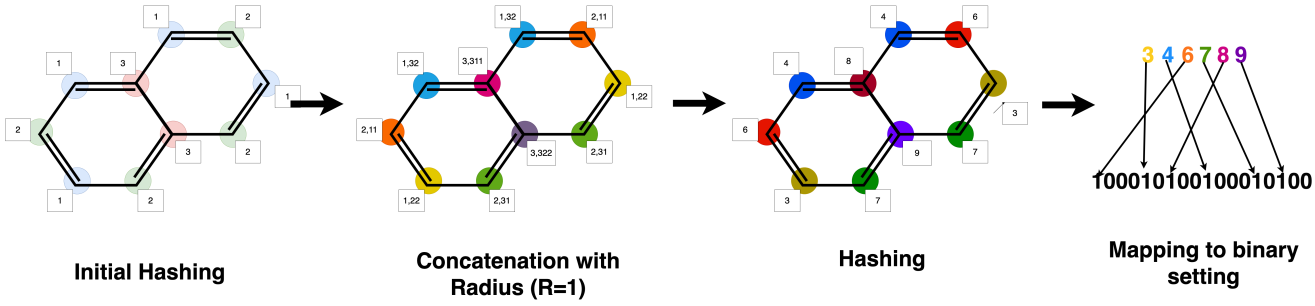

Figure 2.3: **Morgan fingerprinting process for a naphthalene molecule:** The process begins with initial hashing where each atom is assigned an integer based on its features (Step 1). These integers are then concatenated with the integers of their neighboring atoms (Step 2), followed by hashing the concatenated values to produce new integers (Step 3). Finally, the hashed integers are mapped to a binary string to generate the molecular fingerprint (Step 4). (***Adapted from Hernandez-Lobato, Jose M. "Machine Learning for Molecules.", 2018***)

A crucial advantage of Morgan fingerprints is their ability to encode stereo-chemical information, which is particularly important where molecular chirality plays a critical role. The resulting identifiers from the iterative process are hashed into a fixed-length binary or integer vector creating a compact and information-rich representation of the molecule. Unlike structural key-based fingerprints, which rely on predefined patterns, Morgan fingerprints can represent an essentially infinite number of substructures, including those not explicitly coded in any database.

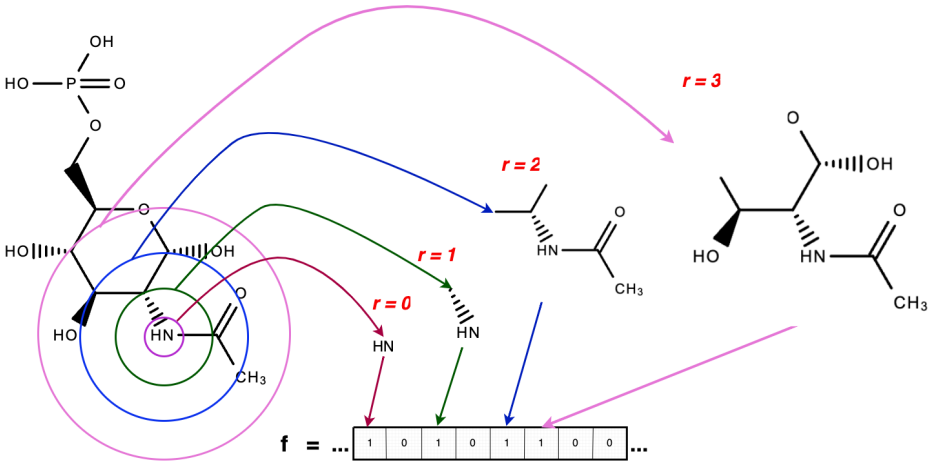

Figure 2.4: **Example of Morgan fingerprinting process applied to a molecule with a manually set radius parameter** $R = 3$**.**The central atom (NH) is considered at radius $r = 0$. Atoms directly bonded to NH are at radius $r = 1$, atoms bonded to those atoms are at radius $r = 2$, and so on. The features of these atoms are concatenated and hashed at each step to generate the molecular fingerprints.(***Adapted from Hernandez-Lobato, Jose M. "Machine Learning for Molecules." MLSS, 2018***)

This allows them to capture novel or unexpected molecular features. This is useful for tasks such as similarity searching and clustering. Furthermore, their ability to generate unique, non-redundant representations ensures efficient and accurate comparisons between molecules, a critical requirement in modern cheminformatics[45]. Acknowledged by Bradshaw et al(2020)[9], as these fingerprints are fixed, models cannot learn which characteristics of a molecule are important for our tasks.

2.4.3 Bit Collisions Phenomenon

Despite these advantages, a significant limitation arises in the context of dimensionality reduction. Most, if not all machine learning models (MIT's SynNET[16], SMILES GA[47], Stanford's MolDQN[48], GP BO [19] are some notable models) in cheminformatics, typically reduce the dimensionality of molecular fingerprints to a fixed size between 1024 to 4096 bits. This dimensionality reduction, while computationally efficient, introduces problem of bit collisions, where different molecular substructures might be hashed into the same bit position. This leads to loss of unique information, and a decrease in discriminative power of the fingerprint. Riniker and Landrum (2013)[49] highlight the impact of bit collisions in their study on dopamine receptor ligands, where reducing the

bit size from 2048 to 1024 bits resulted in overlap of different chemical environments. This problem is addressed in our methodology in Section 3.1.

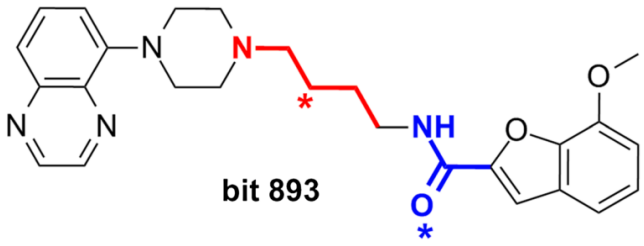

Figure 2.5: **Example of Bit Collision in Morgan2/CountMorgan2 Fingerprints:** The molecular environments highlighted in red and blue are both hashed to the same bit position (bit 893) in the fingerprint vector. The central atom of each environment is marked with a star, indicating the points where the collision occurs, leading to a loss of unique molecular information in the fingerprint(Riniker and Landrum (2013)).

2.4.4 Binary vs Count Fingerprints

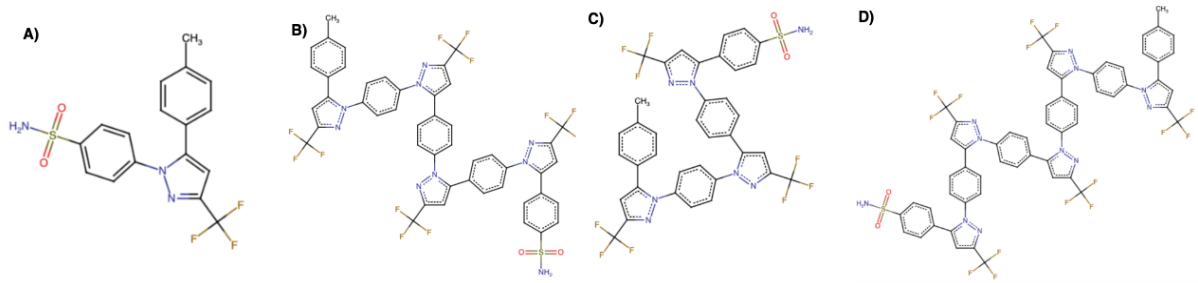

Figure 2.6: **Celecoxib (A) and its larger analogues (B,C,D) are represented by the same Binary Morgan Fingerprints:** This issue arises due to the hash-based nature of Morgan fingerprints, which can lead to identical fingerprint representations when different molecules share certain substructures.

While binary fingerprints indicate the presence or absence of a substructure, count fingerprints provide additional granularity by representing the frequency of those substructures within a molecule[2]. Figure 2.6 highlights a limitation of binary fingerprints, where molecules with different structural sizes, are reduced to identical binary representations. This leads to a potential loss of molecular detail. In contrast, count fingerprints not only capture whether a molecular feature is present but how frequently it occurs. As noted in literature, count-based methods generally improve model performance in tasks requiring

more detailed comparisons. Therefore, count fingerprints are often preferred over binary fingerprints where sub-structural frequency plays a role in molecular optimization.

Contrary to popular belief, the computational efficiency of binary and count fingerprints is essentially the same. Both utilize efficient data structures such as hash tables and hash maps for message passing operations, with similar computational complexity. Therefore, the preference for count-based fingerprints stems from their ability to offer richer molecular representations, particularly in tasks that require more detailed sub-structural comparisons. As a result, count fingerprints are often favored in molecular optimization tasks where the frequency of molecular features plays a crucial role.

Having established the limitations of binary fingerprints and the advantages of count fingerprints, we now turn to the underlying mathematical models that guide our approach. Gaussian Processes (GPs) form the foundation of our model, with the Tanimoto kernel specifically tailored to handle binary fingerprints, capturing molecular similarity through presence-absence patterns. To extend this capability, we incorporate the MinMax kernel, a derivative of the Tanimoto kernel, to better represent count fingerprints, ensuring that the frequency of molecular features is accurately modeled. This theoretical framework, along with Multi-Objective Bayesian Optimization (MOBO), enables efficient exploration of the Pareto front, optimizing for multiple conflicting objectives simultaneously.

2.5 Gaussian Processes

Gaussian Processes (GPs)[50] are a powerful and versatile class of models used in machine learning for regression and classification tasks. Unlike traditional machine learning algorithms that focus on finding a single best-fit model, GPs offer a probabilistic framework, representing a distribution over possible functions that fit the data. This probabilistic nature allows GPs to naturally quantify uncertainty in predictions, which is valuable in tasks where understanding the confidence of predictions is crucial.

At the core of GPs, they are defined as a collection of random variables, any finite subset of which has a joint Gaussian distribution. This means that instead of predicting a single output for the given input, GPs predict a distribution over possible outputs, characterized by mean function $m(x)$ and covariance function $k(x, x')$[50]. The mean function represents the expected output, while the covariance function, which can be a kernel, determined the similarity between different inputs and governs the smoothness and generalization ability of predictions. Formally, a GP is defined as:

$$f(x) \sim \mathcal{GP}(m(x), k(x, x'))$$

where $m(x) = \mathbb{E}[f(x)]$, and $k(x, x') = \mathbb{E}[(f(x) - m(x))(f(x') - m(x'))]$ is the covariance function. The kernel function $k(x, x')$ plays a crucial role in GPs as it encodes the assumptions about the function we wish to learn, such as smoothness or periodicity. The choice of kernel, in which we will justify our case for using these Tanimoto kernels in Section , directly influences the GP's predictions and its ability to model the underlying data effectively.

The flexibility of GPs comes from their non-parametric nature. Unlike parametric models, which assume a specific functional form for the data, GPs can model a wide range of functions by adjusting the kernel. This makes GPs a versatile and highly adaptable tool for various types of data, but also introduces the challenge of selecting an appropriate kernel, and optimizing its hyperparameters, which can be computationally demanding[51].

2.5.1 Predictive Inference with GPs

Gaussian Process Regression (GPR)[50] is the application of Gaussian Processes to regression problems. In GPR, the goal is to infer a distribution over possible functions that fit the observed data. Given a set of training data $\mathcal{D} = \{X, Y\}$, where X represents the input features and Y the corresponding outputs, GPR uses Bayes' theorem to update the prior distribution (the GP) with the observed data, resulting in a posterior distribution

over functions. This joint distribution of observed outputs Y and the function values f_* at the test points X_* is given by:

$$\begin{pmatrix} Y \\ f_* \end{pmatrix} \sim \mathcal{N}\left(\begin{pmatrix} m(X) \\ m(X_*) \end{pmatrix}, \begin{pmatrix} K(X,X) + \sigma_n^2 I & K(X,X_*) \\ K(X_*,X) & K(X_*,X_*) \end{pmatrix}\right)$$

Here $m(X)$ and $m(X_*)$ are the mean functions, $K(X,X)$ is the covariance matrix between training points $K(X,X_*)$ is the covariance between the training points and test points, and σ_n^2 is the variance of the Gaussian noise added to observations.

The posterior distribution, which is gives the predictive mean μ_* and variance $\sum_*$ for the test points, is derived as:

$$\mu_* = m(X_*) + K(X_*,X)\left[K(X,X) + \sigma_n^2 I\right]^{-1}(Y - m(X)) \tag{2.1}$$

$$\Sigma_* = K(X_*,X_*) - K(X_*,X)\left[K(X,X) + \sigma_n^2 I\right]^{-1} K(X,X_*) \tag{2.2}$$

This result demonstrates the power of GPR: it not only predicts the mean values of the ouputs at new point but also provides a measure of the uncertainty of those predictions. This ability to model uncertainty is one of the key strengths of GPR, especially in scenarios where data is sparse or noisy[50]. The predictive mean μ_* is a weighted sum of the observed outputs, where the weights are determined by the covariance between the test points and the training points, normalized by the covariance of the training points. The predictive variance $\sum_*$ on the other hand, decreases as more data points are observed reflecting the increasing certainty of the predictions. An example of this is seen in Figure 3.1 which shows the decrease in variance when increasing the number of training samples.

2.6 Multi-Output Gaussian Processes

While GPRs are a powerful approach in regression tasks, where the goal is to predict a continuous output given an input, many real-world applications, especially in fields of cheminformatics and drug discovery, require the simultaneous optimization of multiple objectives. In such cases, a traditional single-output GP model might fall short. The concept of Multi-Output Gaussian Processes (MOGPs)[28][52][53], extend this single-output GP framework to model multiple outputs jointly. However, in some cases, as assumed in our model, tasks are treated as independent, meaning there is no correlation between the outputs. Each task is modeled by an independent GP, which can be beneficial for computational simplicity and ease of interpretation. An illustration of the MOGP framework assuming independence is provided in Figure 2.7. Now in these later sections,

we delve into the theoretical details of what an MOGP entails.

2.6.1 Covariance Matrix and Multi-Output GP

In the independent task scenario, given a set of D independent objectives, the prior/joint distribution of the functions $f_1, f_2, ..., f_D$ can be represented by the following multivariate normal distribution:

$$\begin{bmatrix} f_1(x) \\ f_2(x) \\ \vdots \\ f_D(x) \end{bmatrix} \sim \mathcal{N} \left(\begin{bmatrix} m_1(x) \\ m_2(x) \\ \vdots \\ m_D(x) \end{bmatrix}, \begin{bmatrix} K_1 & 0 & \dots & 0 \\ 0 & K_2 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & K_D \end{bmatrix} + \Sigma \right) \tag{2.3}$$

where:

1. K_i is the covariance matrix is the covariance matrix corresponding to the i-th output, determined by an arbitrary kernel.
2. $\sum = \text{diag}(\sigma_1^2 I, \sigma_2^2 I, ..., \sigma_D^2 I)$ represents the noise in each objective function.

Generally, without any prior knowledge about the trends of the data, the prior mean function $[m_1(x), m_2(x), ..., m_D(x)]^T$ are usually set to 0 (see Rasmussen et al(2005)[50]. Hence, we set the mean functions to - for the remainder of this research, unless stated otherwise.

2.6.2 Covariance/Kernel Structure for Multi-Output GP

In more detail from Equation 2.3 above, Since we assume independence between tasks, the covariance function $K(f_j, f_{j'})$ for a multi-output GP model is block diagonal:

$$\mathbf{K}_{f,f} = \begin{bmatrix} K_1(x, x') & 0 & \dots & 0 \\ 0 & K_2(x, x') & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & K_D(x, x') \end{bmatrix}$$

This block diagonal structure of the covariance matrix indicates that there is no direct correlation between the different output functions $f_j(x)$ and $f_{j'}(x)$ for $j \neq j'$. This is consistent with the assumption that the outputs are conditionally independent given the latent functions. Each $K_i(x, x')$ represents the covariance for the i-th objective and is computed using a kernel.

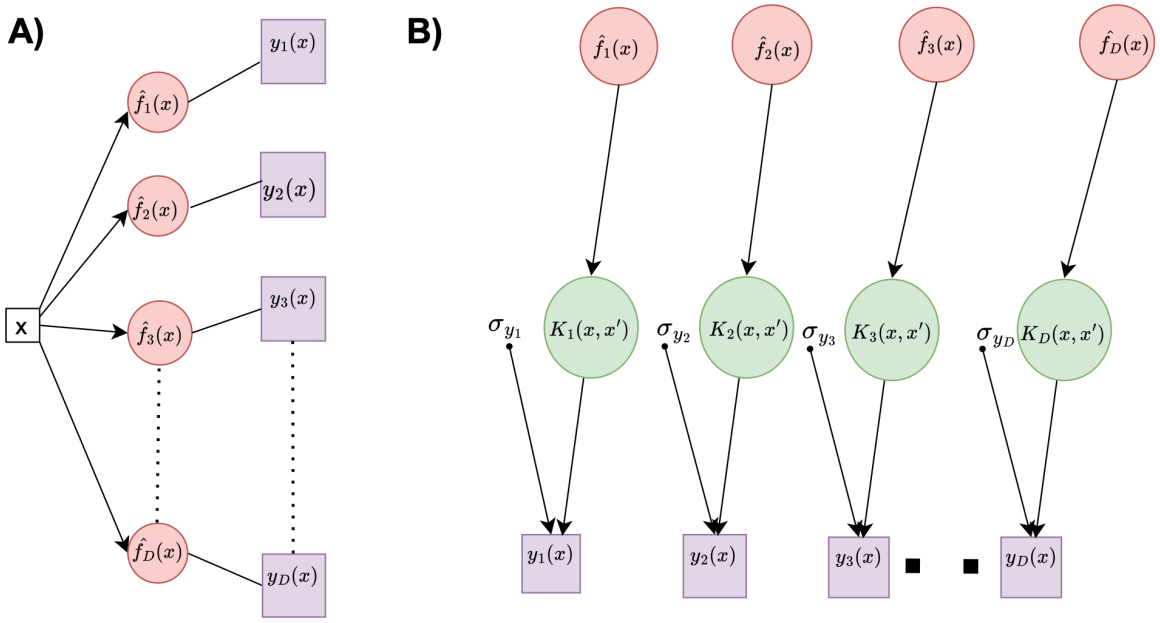

Figure 2.7: **Multi-Output Gaussian Processes (MOGP):** Panel (A) illustrates independent modeling of multiple outputs $y_1(x), y_2(x)..., y_D(x)$ using individual GPs $\hat{f}_1(x), \hat{f}_2(x), ..., \hat{f}_D(x)$ for each output. Input x is mapped to each latent function $\hat{f}_i(x)$ independently which predicts outputs. There is no cross-correlation between outputs, allowing each GP to operate independently. Panel (B) shows each latent function $\hat{f}_i(x)$ is associated with its kernel function $K_i(x, x')$, denoting covariance structure for the i-th output. Noise term σ_{y_i} added to each output to account for observation noise. This model depicted is suited for scenarios where outputs are conditionally independent given latent functions.

2.6.3 Reproducing Kernel for Vector-valued Functions

The kernel functions $k_i(x, x')$, denoted in Figure 2.7, used in a multi-output GP model are reproducing kernels in the Hilbert space of vector-valued functions [28]. Specifically, for each (molecular) objective $f_i(x)$, the kernel $k_i(x, x')$ is a matrix-valued function that maps from $\mathcal{X} \times \mathcal{X} \to \mathbb{R}^{D \times D}$, where D denotes the dimensionality of the output space, where for us, is dependent on the number of objectives we are trying to optimize for[28].

Given a vector-valued function $f(x) = [f_1(x), ..., f_D(x)]^T$ belonging to a Hilbert space $\mathcal{H}$, the reproducing property of the kernel $K(x, x')$ ensures that the inner product in $\mathcal{H}$

corresponds to the evaluation of the function $f(x)$. This is formally expressed as:

$$\langle f(x), f(x') \rangle_{\mathcal{H}} = f(x)^T K(x, x') f(x')$$

This kernel function $K(x, x')$ can be expressed as a matrix acting on a vector $c_j \in \mathbb{R}^D$, allowing us to represent the function $f(x)$ as a sum of kernel evaluations over the training data points:

$$f(x) = \sum_{i=1}^{N} K(x_i, x) c_i$$

Here, the matrix $K(x, x')$ is positive semi-definite and encodes the covariance structure between the different outputs and c_i are the coefficients. We detail this further in our kernels implemented in Section 2.7.3 and 2.7.4.

2.6.4 Gaussian Processes for Vector Valued Functions

For vector-valued functions in a multi-output GP model, the GP is defined as:

$$f(X) \sim \mathcal{GP}(m(X), K(X, X))$$

where $m(X)$ is the vector that concatenates the mean vectors associated with the outputs and the covariance matrix $K(X, X)$ is block-diagonal, with each block corresponding to the covariance matrix $K_i(X, X)$ for the i-th output. The prior distribution is over the outputs is given by:

$$f(X) \sim \mathcal{N}(m(X), K(X, X))$$

As these outputs $f_i(x)$ are independent, and given a set of input points X and corresponding outputs Y, the posterior distribution can be written as:

$$p(f|Y, X, \sum) = \mathcal{N}(f(X), K(X, X))$$

where $\sum$ accounts for noise in each task. For our independent objectives, the predictions for a new test point x_* have the same $\mu_*(x_*)$ and $\sigma_*(x_*)$ calculations to as seen in Equation 2.2 of the Single-output GPs. The predictive mean $\mu_*(x_*)$ and variance $\sigma_*(x_*)$ for each objective are returned as a tuple of vectors:

$$\mu_*(\mathbf{x}_*) = [\mu_1(\mathbf{x}_*), \mu_2(\mathbf{x}_*), \ldots, \mu_D(\mathbf{x}_*)],$$

$$\sigma_*^2(\mathbf{x}_*) = [\sigma_1^2(\mathbf{x}_*), \sigma_2^2(\mathbf{x}_*), \ldots, \sigma_D^2(\mathbf{x}_*)].$$

2.6.5 Gaussian Process Training

Now that we have introduced Gaussian Process Regression (GPR), Multi-Output Gaussian Processes (MOGPs) and their predictive mean and variance, it is crucial to discuss the training of Gaussian Processes, particularly focusing on how the hyperparameters of the model are determined. These hyperparameters play a critical role in the performance and flexibility of the GP model. In this section, we explore the concept of GP training by optimizing the **Negative Log Marginal Likelihood (NLML)**[50], which balances the model's fit to the data and its complexity, thereby avoiding overfitting.

In a GP model, the choice of kernel (or covariance function) is important. The kernel defines the relationship between points in the input space and governs the smoothness, periodicity and other properties of the functions drawn from the GP. Each kernel is parameterized by a set of hyperparameters, denoted by θ. For example, in the commonly used Radial Basis Function (RBF) kernel, we have:

$$k(x, x') = \sigma_f^2 \exp\left(-\frac{(x - x')^2}{2l^2}\right)$$

Here, σ_f controls variance of the function, and l(the length scale) determines how quickly the correlation between points decreases as they move apart in the input space. The noise term σ_n^2 is treated as a hyperparameter, controlling the variance of the noise assumed in the observations. We will discuss the Tanimoto and MinMax kernel functions in more detail in Section 2.7.3 and 2.7.4.

Negative Log Marginal Likelihood (NLML)

The hyperparameters θ are optimized by maximizing the marginal likelihood of the observed data y, given the inputs X and the hyperparameters. The marginal likelihood is obtained by integrating out the functions values from the joint probability distribution, resulting in a Gaussian distribution for the data y:

$$\log p(y|X, \theta) = -\frac{1}{2} y^\top (K_\theta(X, X) + \sigma_y^2 I)^{-1} y - \frac{1}{2} \log |K_\theta(X, X) + \sigma_y^2 I| - \frac{n}{2} \log(2\pi)$$

This expression combines the $-\frac{1}{2} y^\top (K_\theta(X, X) + \sigma_y^2 I)^{-1} y$, the data fit term which encourages the model to fit the observed data closely and $-\frac{1}{2} \log |K_\theta(X, X) + \sigma_y^2 I|$, the complexity penalty term, which penalizes overly complex models to prevent over-fitting. The final term is a normalization constant that does not depend on model parameters. Together, these terms represent the NLML which is minimized to find optimal hyperpa-

rameters usually.

The NLML embodies the principle of Occam's Razor[54], which favours simpler models that sufficiently explain the data without unnecessary complexity. Rasmussen et al(2006)[50], states that models that are too simple underfit the data, while those that are too complex may overfit, capturing noise rather than the underlying function. The marginal likelihood naturally penalizes models that are too complex by incorporating the determinant of the covariance matrix, which grows with model complexity.

The hyperparameters θ are typically optimized using gradient-based methods, given that the marginal likelihood is differentiable with respect to these parameters. This optimization process is a form of Bayesian model selection, where the model automatically balances fit and complexity to avoid over-fitting. For the zero-mean GP with the kernel $\alpha \cdot k(x, x') + s \cdot I$, the optimization of α and s involves computing the gradient of the NLML with respect to these hyperparameters and iteratively updating them to minimize the NLML. However, in this research, we are not optimizing by minimizing the NLML but an integral part of composing our Exact GP framework (see Section 3.1) and as a reminder for future work. In our setup as described in our Experimental Design section, we manually set these GP hyperparameters to the recommended values such as seen in Tripp et al (2021)[19][2].

2.7 Fingerprint-Based Kernels: Tanimoto & MinMax Kernels

Now, that we have discussed multi-output GPs, their theoretical underpinnings and hyperparameters, readers will recognise now that kernels are an integral part within the GP framework. In this section here, we discuss the type of kernel that is implemented within our algorithm and discuss why this is the recommended kernel in cheminformatics. Kernels are a powerful and flexible framework for measuring the similarity between data representations, particularly in fields of cheminformatics and bioinformatics [35]. While traditional graph kernels are designed to work directly with graph structures, in this work, we focus on a specific subset of kernels that operate on molecular fingerprints - **a vectorized representation derived from molecular graphs** as described in Section 2.4.

The Tanimoto and MinMax kernels (later introduced in Section 2.7.3 and 2.7.4), have been traditionally employed as graph kernels in various computational chemistry and cheminformatics studies[43][55]. These kernels were originally designed to measure the similarity between graphs by comparing specific substructures or features within the graphs, such as paths, walks, or other graph sub-components[56][57]. The work by Ralaivola et al(2005)[35] introduced the Tanimoto kernel as a normalized variant that evaluates the similarity between two graphs based on the overlap of common features, and the MinMax kernel [35][58] modifies this approach to better handle variations in the feature distributions between different graphs. Both of these kernels have traditionally been used in the context of graph-based representations [43][57][59][60].

However, one of the fundamental challenges in working with graph data, is the problem of graph isomorphism, where two graphs $\mathcal{G}_1$ and $\mathcal{G}_2$ are considered identical if there exists a bijective mapping f from the vertices of $\mathcal{G}_1$ to the vertices of $\mathcal{G}_2$ such that the edges are preserved[61][62]. Determining whether two graphs are isomorphic is a computationally challenging problem, and no polynomial-time algorithm is known for general graph isomorphism, making it an NP-complete problem. In other words, the process of checking if two graphs are the same (graph isomorphism), is tricky as there is no fast straightforward way to do it as it requires a lot of computation[61][62].

To address these challenges, molecular fingerprints offer a practical alternative by providing a vectorized summary of the graph's structure. This allows the application of fingerprint-based kernels, which compute a similarity measure between molecules based on these fingerprints rather than requiring direct comparison of the entire graph struc-

tures. This has recently been attempted by Tripp et al(2021)[19], Tripp and Lobato(2024)[2] and Griffiths et al(2022)[51].

Fingerprint-based kernels, such as Tanimoto and MinMax kernels, map molecular fingerprints into a high-dimensional feature space where a similarity measure, generally the inner product, can be computed efficiently. The key idea is to define a kernel function $k(x, x')$ that captures the similarity between two molecules represented by their fingerprints x and x'. These kernels are particularly effective because they can leverage structural information encoded in the fingerprints while avoiding the computational complexity associated with direct graph comparisons.

To further understand how these fingerprint-based kernels function in practice, we now delve into the mathematical foundation of kernel methods using the concept of Reproducing Kernel Hilbert Spaces (RKHS)[63][64]. The RKHS framework provides an understanding how kernel functions operate by mapping input data into high-dimensional spaces, enabling computation of similarity measures without explicitly performing the mapping. This approach is known as the *kernel trick*[63].

2.7.1 Defining a Reproducing Kernel Hilbert Space by implicit mapping

Reproducing Kernel Hilbert Spaces (RKHS) provide a powerful theoretical framework for kernel methods, which are central to many machine learning algorithms[63][64], including those in cheminformatics. In this section, we delve into the fundamentals of RKHS and illustrate how they are applied in fingerprint-based kernels like Tanimoto and MinMax kernels.

A Hilbert space $\mathcal{H}$ is a complete inner product space, meaning that it is a vector space over a field of scalars (typically real or complex) equipped with an inner product $\langle \cdot, \cdot \rangle_{\mathcal{H}}$. The inner product induces a norm $|| \cdot ||_{\mathcal{H}} = \sqrt{\langle \cdot, \cdot \rangle_{\mathcal{H}}}$, which can be interpreted as a measure of length or distance between vectors. Completeness in this context means that every Cauchy sequence (a sequence where the distance between successive terms can be made arbitrarily small) converges to a point within the space. This ensures that the space has no ***"gaps"*** and can support application of limit processes, which are critical in the analysis of functions and operators in infinite-dimensional spaces.

Reproducing Kernel and Feature Maps

A concept in RKHS is the reproducing kernel[63], annotated generally as $k(x, y)$, which defines the inner product between elements in the space.

Definition 1 ***Hilbert spaces:*** *Given a set $\mathcal{X}$ and a Hilbert space $\mathcal{H}$ of real-valued functions on $\mathcal{X}$, the function $k : \mathcal{H} \cdot \mathcal{H} \to \mathbb{R}$ is a reproducing kernel if it satisfies the following properties for all $x, y \in \mathcal{X}$:*

- *$k(x, \cdot)$ belongs to $\mathcal{H}$ for every $x \in \mathcal{H}$*
- *For every function $f \in \mathcal{H}$ and every $x \in \mathcal{X}$, the reproducing property holds:*

$$f(x) = \langle f, k(x, \cdot) \rangle_{\mathcal{H}}$$

This reproducing property allows the evaluation of any function in $\mathcal{H}$ through an inner product with the kernel function. It implies that kernel function $k(x, y) = \langle k(x, \cdot), k(y, \cdot) \rangle$ not only acts as a measure of similarity between points x and y in the input space, but also represents the action of evaluating $f(x)$ any any x through the inner product in the Hilbert space $\mathcal{H}$. The RKHS framework is important in kernel methods as it allows the implicit mapping of input data into a high-dimensional feature space without explicitly computing the mapping. This is achieved through the kernel trick, which we will discuss in the next section. An example of this is molecular graphs are mapped into the Hilbert space here using their fingerprint vectors in the Figure 2.8 below.

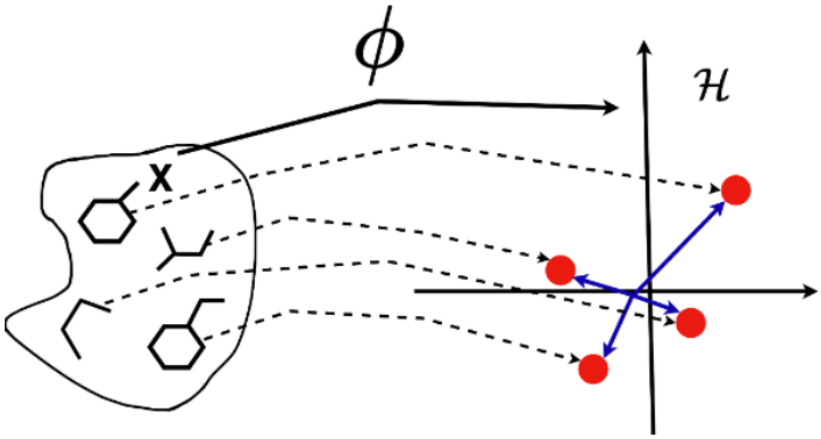

Figure 2.8: **Molecular structures mapped into the Hilbert Space $\mathcal{H}$ using fingerprint vectors (e.g. a binary digit string):**Molecular structures are represented as points in a high-dimensional feature space, where fingerprint-based kernels, such as Tanimoto and MinMax, compute the similarity between these molecular fingerprints. These kernels operate on the vector representation of molecules, capturing structural similarities implicitly in the Hilbert space $\mathcal{H}$, without requiring direct comparison of the original molecular graph structures

2.7.2 Kernel Trick in Fingerprint-based Kernels

The kernel trick is an essential concept in machine learning, particularly in the context of algorithms that involve similarity measures, such as Support Vector Machines (SVMs)[65]. The kernel trick revolves around the idea that many linear algorithms for regression or pattern recognition can be expressed solely in terms of inner products between feature vectors. Specifically let $\Phi(x)$ be a mapping that represents a molecular fingerprint x in a high-dimensional feature space $\mathcal{H}$. The kernel function $k(x, x')$ can be defined as the inner product of the mapped vectors in this space:

$$k(x, x') = \langle \Phi(x), \Phi(x') \rangle_{\mathcal{H}}$$

The inner product computes the similarity between two molecular fingerprints x and x' by comparing them in high-dimensional representations. However, directly computing the mapping $\Phi(x)$ for molecular fingerprints can be computationally inefficient, particularly when the feature space $\mathcal{H}$ is of very high or infinite dimensionality. From the Kernel Trick definition below, this allows us to bypass explicit computation of the mapping Φ by computing the kernel $k(x, x')$ directly, which is often significantly more efficient.

Definition 2 ***Kernel Trick:*** *Consider the computation of distances in the feature space $\mathcal{H}$ here, the distance between the feature representations of two fingerprints x and x' in $\mathcal{H}$ can be expressed as:*

$$d_{\mathcal{K}}(x, x')^2 = ||\Phi(x) - \Phi(x')||^2_{\mathcal{H}}$$

We have:

$$d_{\mathcal{K}}(x, x')^2 = \langle \Phi(x), \Phi(x) \rangle_{\mathcal{H}} + \langle \Phi(x'), \Phi(x') \rangle_{\mathcal{H}} - 2\langle \Phi(x), \Phi(x' \rangle_{\mathcal{H}}$$

With kernel trick, this distance is computed as:

$$d_{\mathcal{K}}(x, x')^2 = k(x, x) + k(x', x') - 2k(x, x')$$

From this formulation, it is advantageous as it avoids to explicitly compute or store the high-dimensional feature vectors, allowing kernel methods to scale to large datasets and complex graph structures. For the kernel trick to be applicable, it is crucial that the kernel function $k(x, x')$ be a positive definite kernel (see Appendix 8.2.3 for details on Positive Definite Kernels).

Having laid the theoretical foundation of the kernel trick and reproducing kernels, we now turn our focus to specific kernels used in molecular similarity analysis: the Tanimoto and MinMax kernels. These fingerprint-based kernels are relevant to cheminformatics, where measuring the similarity between molecular fingerprints is essential for tasks in drug discovery and molecular optimization. Both the Tanimoto and MinMax kernels implement the kernel trick to compute similarity efficiently without explicitly mapping molecular structures into high dimensional spaces. As proven by Ralaivola et al (2005)[35], the Tanimoto kernel is a Mercer kernel, which means it satisfies Mercer's theorem and can be used within different frameworks of kernel-based learning methods like SVMs or Gaussian Processes (GPs).

2.7.3 Tanimoto Kernels: Binary Morgan Fingerprints

The Tanimoto Kernel is particularly well-suited for binary Morgan fingerprints (explained in Section 2.4), which are designed to represent the presence or absence of specific molecular features within a molecule. These binary fingerprints are vectors of 0s and 1s, where each element signifies whether a particular molecular substructure or feature is present (1) or absent (0). The key advantage of using binary fingerprints lies in their ability to

encode molecular structures compactly, allowing for efficient similarity calculations using kernel methods.

In cheminformatics, the Tanimoto kernel is a fundamental tool for quantifying the similarity between two molecular fingerprints by comparing shared features. This kernel focuses on the structural overlap between molecules, making it ideal for applications where shared substructures are of primary importance.

Defining the Tanimoto Coefficient

At its core, the Tanimoto coefficient (sometimes referred to as the Jaccard index)[35][66][67] provides a measure of similarity between two sets, in this case, the sets of features present in two molecules. Given two binary fingerprints f_1 and f_2, the Tanimoto coefficient is defined as:

$$T(f_1, f_2) = \frac{|f_1 \cap f_2|}{|f_1 \cup f_2|} \tag{2.4}$$

This formulation here expresses the ratio of the number of shared features (the intersection) to the total number of features present in either molecule. The Tanimoto coefficient takes values between 0 and 1:

- $T(f_1, f_2) = 1$ when f_1 and f_2 are identical (i.e. all features are shared).
- $T(f_1, f_2) = 0$ when f_1 and f_2 have no shared features.

This measure is widely used in cheminformatics, particularly for comparing molecular structures encoded as binary fingerprints. When extended to kernels, the Tanimoto coefficient naturally forms the bases for the Tanimoto kernel, formally defined by Ralaivola et al(2005)[35], as:

$$k_T(f_1, f_2) = \frac{k_{\varphi_d}(f_1, f_2)}{k_{\varphi_d}(f_1, f_1) + k_{\varphi_d}(f_2, f_2) - k_{\varphi_d}(f_1, f_2)} \tag{2.5}$$

where $k_{\varphi_d}(f_1, f_2)$ represents a dot product kernel between the feature maps of the two fingerprints, as explained below.

Dot Product Kernels on Molecular Fingerprints

Once the molecular fingerprints are transformed into vectors via the binary features map, the next step is to define a **dot product kernel** that quantifies the similarity between two fingerprints based on their binary representations. For two fingerprints f_1 and f_2,

the dot product kernel is:

$$k_{\varphi_d^{\text{bin}}}(f_1, f_2) = \sum_{p \in \mathcal{P}(d)} \mathbb{I}\{p \subseteq f_1\} \cdot \mathbb{I}\{p \subseteq f_2\}$$

This dot product kernel measures the number of shared features between two fingerprints, where the binary feature map φ_d^{bin} is used to represent each features presence. Building on the dot product kernel, the Tanimoto kernel introduces normalization to account for the size of molecular fingerprints as defined in Equation 2.5. The kernel's ability to normalize for size of fingerprints makes it a robust similarity measure in cheminformatics. It ensures that the fingerprints with a large number of features do not unduly bias the similarity score, providing a balanced comparison between different-sized molecules. Additionally, it has been proven by Ralaivola et al(2005)[35], that this kernel is positive semi-definite and satisfies the Mercer's Theorem (definition provided in Appendix 8.2.2).

2.7.4 MinMax Kernels: Count-based Fingerprints

The MinMax kernel [58][35] builds upon the Tanimoto kernel, focusing on count-based Morgan fingerprints rather than binary representations. Whereas the Tanimoto kernel compares molecular structures based on the presence or absence of substructures, the MinMax kernel quantifies the similarity between fingerprints by considering both the presence and multiplicity of substructures. This makes it particularly well-suited for cases where features may repeat across different substructures within a molecule. MinMax kernels are often employed when comparing more complex molecules that have varying degrees of feature repetition, providing a more nuanced similarity measure compared to the Tanimoto kernel.

Count-based Morgan Fingerprints

Count-based Morgan fingerprints (as generally defined in Section 2.4 extend the binary representations discussed in the Tanimoto kernel by counting the number of occurrences of a given substructure or feature in the molecule. Each fingerprint is transformed into a vector where each element represents the number of times a particular substructure (or feature) occurs. For a molecule fingerprint f, the count-based feature map is defined as:

$$\varphi_d^{\text{count}}(f) = (\#\{p \subseteq f\})_{p \in \mathcal{P}(d)}$$

where $\#\{p \subseteq f\}$ represents the number of occurrences of a particular substructure p within the fingerprint f, and $\mathcal{P}(d)$ represents the set of possible features of length d.

This vector captures not only the presence of features but also their multiplicity, which is essential for molecules where substructures may occur more than once.

MinMax Kernel Definition

The MinMax kernel $k_{\text{MinMax}}(f_1, f_2)$ is designed to compare two molecular fingerprints f_1 and f_2 by computing the ratio of the sum of the minimum values to the sum of the maximum values of the feature counts across all paths $p \in \mathcal{P}(d)$. This formulation takes into account the multiplicity of features in each fingerprint, providing a detailed similarity measure.

For two fingerprints f_1 and f_2, the MinMax kernel is defined as:

$$k_{\text{MinMax}}(f_1, f_2) = \frac{\sum_{p \in \mathcal{P}(d)} \min(\#\{p \subseteq f_1\}, \#\{p \subseteq f_2\})}{\sum_{p \in \mathcal{P}(d)} \max(\#\{p \subseteq f_1\}, \#\{p \subseteq f_2\}} \tag{2.6}$$

where:

- $\min(\#\{p \subseteq f_1\}, \#\{p \subseteq f_2\})$ represents the minimum number of occurrences of a substructure p across two fingerprints.
- $\max(\#\{p \subseteq f_1\}, \#\{p \subseteq f_2\})$ represents the maximum number of occurrences of the same substructure.

This ratio ensures that the kernel reflects the proportional similarity between the two fingerprints, accounting for both shared substructures and their multiplicities. In scenarios where features are repeated within the molecules, such as larger analogues of repeated substructures, as shown in Figure 2.6, the MinMax kernel offers a more accurate measure of molecular similarity than the Tanimoto kernel, which only considers binary presence or absence.

Connection between MinMax and Tanimoto Kernel

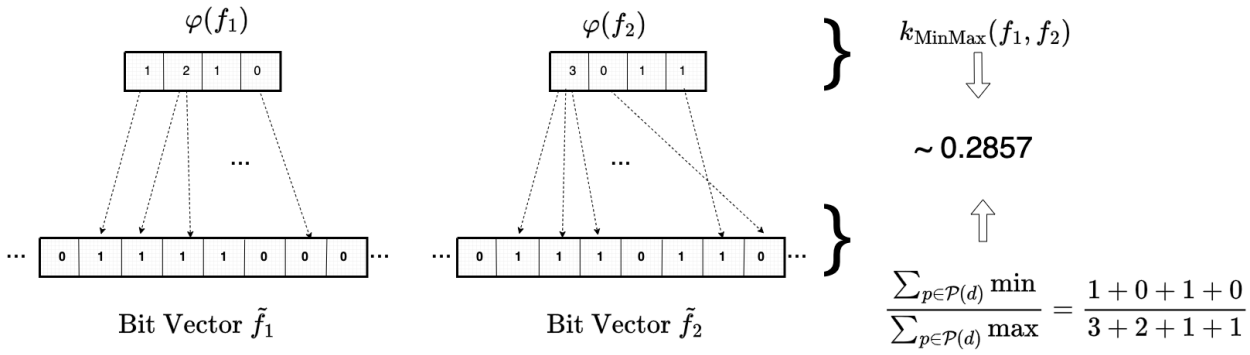

Figure 2.9: **The connection between MinMax and Tanimoto Kernels:** The feature vectors φf_1 and φf_2, representing two molecules, are transformed into their corresponding bit vectors $\tilde{f}_1$ and $\tilde{f}_2$, derived from the Morgan fingerprinting process (as depicted in Figure 2.3 and 2.4. The MinMax kernel is calculated by taking ratio fo sum of minmum values to sum of maximum values across all paths $p \in \mathcal{P}(d)$. This is an extension of the Tanimoto kernel, where feature counts are considered.

Formally, from Figure 2.9, the feature vectors φf_1 and φf_2 represent count-based Morgan fingerprints, where the values indicate how many times a substructure appears in a molecule. The corresponding bit vectors $\tilde{f}_1$ and $\tilde{f}_2$ represent the binary Morgan fingerprinting interpretation (depicted in Figure 2.3 for the two molecules. The MinMax kernel uses the feature counts (top row) whereas the Tanimoto kernel operates on the binary information in the bottom row.

The MinMax kernel can thus be considered an extension of the Tanimoto kernel, where binary presence or absence is replaced with feature counts, allowing for a finer-grained similarity measurement. In fact, the MinMax kernel reduces to the Tanimoto kernel when the fingerprints are strictly binary, where each features appears either once or not at all. The connection between the two lies in their shared structure - both compute a ratio of intersection to union, but differ in how they handle feature representation.

2.8 Multi-objective Bayesian Optimization

In multi-objective optimization, we often aim to find a set of solutions rather than a single optimal solution due to the inherent conflict between objectives. These solutions form the Pareto front, representing the best trade-offs across the objectives. The core goal of Multi-Objective Bayesian Optimization (MOBO) is to efficiently approximate this Pareto front by leveraging Gaussian Process (GP) surrogate models that estimate the underlying objective functions.

Building on the foundation laid in the previous sections on Tanimoto kernels and GPs, this section delves into MOBO by utilizing the predictive uncertainty provided by GPs to guide the search for Pareto-optimal solutions.

However, unlike Single-Objective Bayesian Optimization (SOBO), where only one objective is optimized, MOBO tackles the complexity of real-world problems involving conflicting objectives. The search for a Pareto front necessitates more sophisticated acquisition functions that account for the trade-offs between objectives. This leads us to the Hypervolume Indicator (HVI)[68](Section 2.8.2), which is critical for assessing the quality of Pareto approximations.

While Expected Improvement (EI) and other acquisition functions work well in SOBO, in the multi-objective setting, we require acquisition functions like Expected Hypervolume Improvement (EHVI)[69], which can identify new points that improve the Pareto front. EHVI operates by measuring the increase in the hypervolume dominated by the Pareto front. Hence, understanding hypervolume computations becomes essential in multi-objective settings.

In the following sections, we will dive deep into the Hypervolume Indicator (HVI) and its role in guiding MOBO towards better Pareto approximations. This detailed discussion is necessary because hypervolume-based methods, such as EHVI, are computationally intensive, especially as the number of objectives increases. Therefore, efficient algorithms for hypervolume calculation, like the Hypervolume by Slicing Objectives (HSO)[70](Section 2.8.4) and Improved Dimension-Sweep (IDSA)[71](Section 2.8.5) methods, are integral to ensuring the scalability of MOBO.

2.8.1 Acquisition Functions

In Bayesian Global Optimization (BGO), acquisition functions play a pivotal role in balancing exploration and exploitation by utilizing the uncertainty quantification provided by the Gaussian Process (GP) model. Common acquisition functions include

Expected Improvement (EI), Probability of Improvement (PI), and Upper Confidence Bound (UCB), which guide the selection of the next points to evaluate by optimizing a criterion that considers both the predicted mean and uncertainty.

For single-objective optimization, Expected Improvement (EI) is widely used. It is computed as

$$EI(x_*) = \mathbb{E}[\max(0, f(x_*) - f_{\text{best}})] \tag{2.7}$$

where f_{best} is the best function value observed so far. The EI acquisition function encourages sampling in regions where the GP predicts high mean values and/or high uncertainty, thereby efficiently guiding the search towards the global optimum.

However, in the context of multi-objective optimization, where the goal is to find a Pareto-optimal front rather than a single optimal point, the EI approach becomes insufficient. In multi-objective settings, we require an acquisition function that can simultaneously handle multiple objectives and conflicting trade-offs. This is where the concept of Expected Hypervolume Improvement (EHVI) comes into play.

Proposed by Emmerich et al(2006)[72], EHVI extends the idea of EI by incorporating the Hypervolume Indicator (HV), which measures the region in the objective space dominated by the Pareto front. EHVI leverages both a Pareto-front approximation and the predictive uncertainty provided by the GP model to identify points that improve the Pareto front. While EI works well for single objectives, EHVI is designed to handle the computational complexities of multi-objective optimization, providing an effective balance between convergence and diversity across objectives.

In the following sections, we will discuss hypervolume computations in more detail, as these are critical for multi-objective acquisition functions like EHVI, which require efficient handling of high-dimensional Pareto fronts.

2.8.2 Hypervolume Indicator

In Multi-Objective Bayesian Optimization (MOBO), the Hypervolume Indicator (HV) is a vital metric for evaluating the performance of Pareto-front approximations. The HV indicator measures the size of the region in the objective space that is dominated by the Pareto front and bounded by a predefined reference point r[68]. It plays a key role in acquisition functions like Expected Hypervolume Improvement (EHVI), which we will introduce in due time.

Definition 3 ***Hypervolume Indicator:*** *Given a set of points* $P = \{y^{(1)}, y^{(2)}..., y^{(n)}\} \subset \mathbb{R}^d$*, the Hypervolume Indicator HV(P) is formally defined as the d-dimensional Lebesgue*

measure of the region dominated by P and bounded above by a predefined reference point r:

$$HV(P) = \lambda_d \left(\bigcup_{y \in P} [y, r] \right) \tag{2.8}$$

where λ_d denotes the Lebesgue measure on $\mathbb{R}^d$, and $[y, r] = \{z \in \mathbb{R}^d | y \leq z \leq r\}$ represents the axis aligned hyperrectangle with diagonal corners at y and r.

Computing the HV indicator, specifically in higher dimensions, is computationally expensive. For dimensions $d \geq 2$, it can computed in $\mathcal{O}(n \log n)$ time, where n is the number of points in the set P. However, as the number of dimensions increases, complexity times grows exponentially. Despite the computational complexity, the HV indicator is often used in evolutionary multi-objective optimization algorithms (EMOAs)[68][73] as it is one of the few indicators that capture both convergence and diversity of solutions effectively. Additionally, the properties and relevance of the Lebesgue Measure [74] for the HV indicator are shown in Appendix 8.2.4.

2.8.3 Pareto Points and Pareto Optimality

In MOBO, Pareto optimality [75] is crucial in navigating the trade-offs between conflicting objectives. No single solution can typically be considered "best" across all objectives, so we instead focus on finding solutions that improve some objectives without deteriorating others. These solutions form the Pareto-optimal front, representing the optimal trade-offs in the objective space.

Definition 4 ***Non-dominated Solution Set:*** *This is the set of all solutions that are not dominated by any other solution in the decision space. Formally a solution x^* is considered non-dominated (or Pareto-optimal) if there is no other solution x such that:*

$$f_i(x) \leq f_i(x^*) \text{ for all } i \in \{1, 2, ..., m\}$$

and

$$\exists_j \in \{1, 2, .., m\} \text{ such that } f_j(x) < f_j(x^*)$$

Here, $f_1, f_2, ..., f_m$ are the objective functions, and x^* represents a decision vector in the feasible decision space. The pareto-optimal front consists of all such non-dominated solutions, providing a set of optimal trade-offs in the decision space.

In the context of MOBO, finding the Pareto-optimal set is essential for constructing

effective acquisition functions that balance multiple objectives. As we will see, acquisition functions like EHVI rely on this concept here to guide exploration and exploitation across conflicting objectives.

2.8.4 Hypervolume by Slicing (HSO) Algorithm

Hypervolume is one of the most critical metrics for evaluating Pareto-front solutions in MOBO, as it captures both the diversity and convergence of solutions. However, computing hypervolumes, particularly in higher-dimensional objective spaces, presents significant computational challenges. This section introduces the Hypervolume by Slicing Objectives (HSO) algorithm by While et al(2006)[70], a powerful method that addresses these challenges by efficiently calculating hypervolumes through lower-dimensional slices. Understanding this algorithm is crucial as it forms the backbone of hypervolume-based acquisition functions like Expected Hypervolume Improvement (EHVI), which are integral to MOBO strategies. The HSO algorithm here improves upon previous methods by focusing on processing objectives rather than individual points, which allows for significant reductions in computational complexity, especially in optimization problems with 3 or more objectives.

HSO operates by slicing the objective space into hypervolumes of lower dimensionality, processing these slices individually, and then summing their contributions to compute the total hypervolume. This approach significantly reduces redundant calculations, particularly in higher-dimensional spaces, making HSO faster than other HV indicator methods, such as the LebMeasure algorithm.

Definition 5 ***Hypervolume by Slicing (HSO):*** *Let $S = \{x_1, x_2, ..., x_m\}$ be a set of m mutually non-dominating points in n objectives, where each x_i is a vector from $(x_{i1}, x_{i2}, ..., x_{in})$. The hypervolume $HV(S)$ is the measure of the union of the hyperrectangles defined by these points and a reference point $r = \{r_1, r_2, ..., r_n\}$. The hypervolume is expressed as:*

$$HV(S) = \int_{\mathbb{R}^n} \mathbb{1}_{\bigcup_{x \in S} R(x)}(z) \delta z$$

where $R(x)$ is the hyperrectangle dominated by x and bounded by r.

HSO simplifies this by slicing the space along each objective, reducing the problem to a series of lower-dimensional hypervolume calculations. Specifically, after sorting the points by the first objective, HSO slices the hypervolume into sections, each corresponding to a distinct value of the first objective. Each section is then a hypervolume calculation in

$n-1$ objectives, and the process is repeated recursively. For convenience, as the original authors have not provided a high-level pseudoalgorithm for HSO, this is presented here below in Algorithm 1. A visual interpretation of the HSO algorithm is additionally provided below in Figure 2.10.

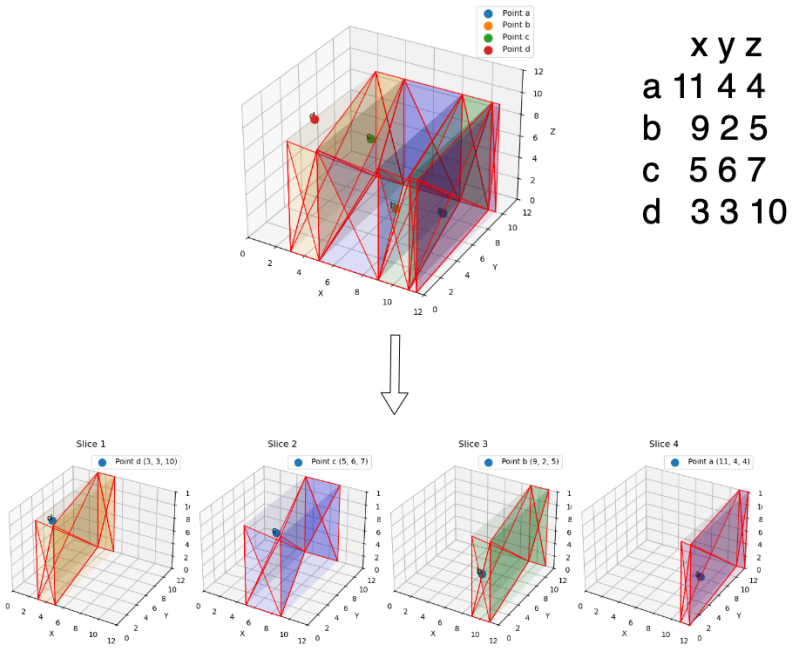

Figure 2.10: **Hypervolume by Slicing Objectives (HSO) applied to 4 three-objective points:** The 3D-space is sliced along the X-axis, generating two-objective shapes in the Y-Z plane for each slice. Points are labeled with their respective coordinates. The red lines represent the boundaries of each 3D slice, showing how the objective space is partitioned at each step in the HSO process.

They key operation here, is the slicing of the objective space. We will attempt to provide a definition here:

Definition 6 ***Objective Space Decomposition in HV calculation:*** *At each step k, the list of points is sliced based on their values in the k-th objective. Each slice represents a section of the objective space where all points have a fixed value in the k-th objective, reducing the dimensionality by one. For a given slice at dimension k, the hypervolume contribution V_k can be expressed as:*

$$V_k = (x_{(i+1)k} - x_{ik}) \times HV_{n-1}(S')$$

where $x_{(i+1)k}$ and x_{ik} are the boundaries of the slice in the k-th objective, and $HV_{n-1}(S')$ is the hypervolume of the slice calculated in the $(n-1)-$dimensional space.

This process is recursively applied, with each step reducing the problem by one dimension, until the final one-dimensional slices are summed to give the total hypervolume.

In Algorithm 1 below, it begins by sorting the set S according to the values of the first objective x_1 in descending order. It then initializes a list L with a single entry containing the entire set of S and a multiplier of 1. The algorithm proceeds by iteratively slicing S across each objective k from 1 to $n-1$, where each slice generates sublists for the next dimension. The contribution of each sublist is computed by multiplying the width of that slice in the current dimension by the sublist's corresponding multiplier, and the results are accumulated in a new list L'. This process is repeated until only the final dimension remains, at which point the hypervolume is computed by summing the products of the multipliers and the widths of the slices in this last dimension.

Algorithm 1 Hypervolume by Slicing Objectives (HSO) Algorithm (While et al(2006))

Input: Set of points S in n objectives, Reference point r
Output: Hypervolume $HV(S)$
procedure HSO(S, n)
  $S \leftarrow$ sort S by Objective 1 descending
  Initialize $L \leftarrow \{(1, S)\}$ ▷ Each entry: (multiplier, point list)
  for $k \leftarrow 1$ to $n-1$ **do**
    $L' \leftarrow \{\}$
    for (m, pl) in L **do**
      $L' \leftarrow L' \cup$ slice(pl, k, m)
    end for
    $L \leftarrow L'$
  end for
  $HV \leftarrow \sum_{(\text{m,pl}) \in L} \text{m} \times (\text{head(pl)}[n] - r[n])$
  return HV
end procedure
procedure SLICE(pl, k, m)
  Initialize $S \leftarrow \{\}$
  $p \leftarrow$ head(pl), $pl \leftarrow$ tail(pl)
  while $pl \neq \emptyset$ **do**
    $q \leftarrow$ head(pl)
    $S \leftarrow S \cup \{(m \times |p[k] - q[k]|, \text{current\_slice})\}$
    $p \leftarrow q$, $pl \leftarrow$ tail(pl)
  end while
  return $S \cup \{(m \times |p[k] - r[k]|, \text{current\_slice})\}$
end procedure

Complexity Analysis

While HSO offers significant computational efficiency improvements, particularly for problems with 3 or more objectives, it still faces scalability issues as the number of objectives grows. To address these, the Improved Dimension-Sweep Algorithm (IDSA)[71] further refines the process by introducing advanced pruning techniques and the reuse of previous calculations. This not only reduces redundant computations but also ensures faster convergence for higher-dimensional problems.

2.8.5 Improved Dimension-Sweep Algorithm

The Improved Dimension-Sweep algorithm (IDSA) by Fonseca et al(2006)[71], builds directly on the HSO framework. It incorporates key innovations in pruning and computational reuse that allow it to scale more effectively in higher-dimensional spaces. By reducing the number of redundant calculations and ensuring that previous hypervolume computations are reused where possible, IDSA offers a significant improvement in performance over traditional hypervolume calculation methods. The primary objective of this algorithm is to efficiently compute the HV indicator defined by Definition 5 here for a set of n-non-dominated points in d dimensions. These improvements are discussed in the Sections A, B, and C below.

A) Recursive Dimension-Sweeping HV Calculation

Based on Paquete et al (2006)[76], the computation of the HV indicator can be acknowledged as a specialized instance of Klee's Measure Problem (detailed in Appendix 8.2.5). In this context of the HV indicator, a special case of the Klee's Measure problem is considered, where all hyperrectangles share the same lower vertex, typically by a reference point r.

Definition 7 ***Recursive Dimension-Sweep Algorithm:*** *Let P be a set of points n points in $\mathbb{R}^d$ and $HV_d(P)$ denote the hypervolume of the region dominated by P with respect to a reference point r. The algorithm works by decomposing $HV_d(P)$ into lower-dimensional hypervolumes:*

$$HV_d(P) = \sum_{i=1}^{n} (p_{i,d} - p_{(i+1),d}) \cdot HV_{d-1}(P_i)$$

where $p_{i,d}$ is the d-th coordinate of the i-th point, and $HV_{d-1}(P_i)$ is the hypervolume of the (d-1)-dimensional region dominated by the points P_i in the slice corresponding to $p_{i,d}$.

The computational complexity of this algorithm is $\mathcal{O}(n^{d-2} \log n)$ for general d-dimensional cases, which is a significant improvement over $\mathcal{O}(n^{d-1})$ complexity of non-recursive approaches such as HSO. For $d > 3$, the algorithm first processes the highest dimension, decomposing the problem into a series of $(d-1)$-dimensional hypervolume calculations. Each of these calculations, in turn is further decomposed until the algorithm reaches the base case of 3 dimensions, where an optimized algorithm with $\mathcal{O}(n \log n)$ complexity is applied.

B) Pruning the Recursion Tree

The central idea behind pruning is to recognize when certain recursive branches can be safely ignored without affecting the final hypervolume calculation. Specifically, if a point p in a higher-dimensional space dominates another point q in all lower dimensions, the contribution of q to the hypervolume in those dimensions becomes redundant. As a result, the algorithm can skip the recursive call associated with q.

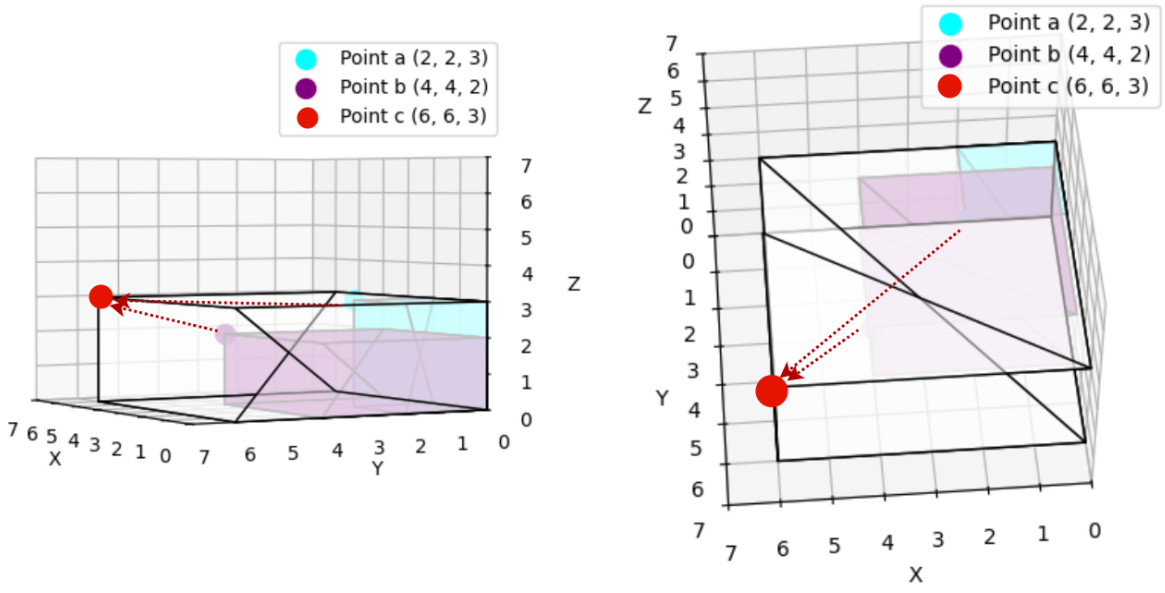

Figure 2.11: **Pruning the Recursion Tree in Recursive Dimension-Sweep Algorithm:** The hyperrectangles corresponding each respective point are shown, with the hypervolume in 3D space depicted by the rectangles' colors. The red arrow indicates the pruning process, where the hyperrectangles for Points A and B can be **ignored** in further calculations as it is dominated by Point C in the subsequent 2D slices.

Definition 8 ***Pruning Recursion Tree Condition:*** *Let $P = \{p_1, p_2, ..., p_n\}$ be a set of n non-dominated points in $\mathbb{R}^d$. The hypervolume contribution of a point p_i is denoted by $HV_d(p_i)$. Specifically, for each point p_i, if there exists another point p_j, such that p_j dominates p_i in the subspaces corresponding to the last $d-1$ dimensions, the contribution $HV_d(p_i)$ becomes redundant and can be omitted.*

For p_j to dominate p_i in the last d-1 dimensions, it must hold that:

$$p_{jk} \geq p_{ik} \forall k \in \{2, 3, ..., d\}$$

where p_{jk} and p_{ik} represent the coordinates of p_j and p_i in dimension k, respectively. If this condition is satisfied, hypervolume contribution of p_i within subspace defined by dimensions $\{2, ..., d\}$ is fully dominated by p_j, implies:

$$HV_d(p_i) \subseteq HV_d(p_j)$$

Recursive calculation for p_i can be pruned/skipped, as it does not contribute to any additional volume to the total HV.

C) Reusing Previous Calculations:

Another further improvement is the reuse of previous hypervolume calculations. When processing the hypervolume of a $(d-1)-$polytope defined by the remaining points below a certain level in T, the computed hypervolume can be stored and reused when possible. This reduces the number of calculations needed, as the algorithm does not need to recompute the hypervolume for every recursive call.

Let $V[p_i, j]$ be the stored hypervolume of the polytope defined by points below p_i in dimension j. Then, the hypervolume at each level can be updated efficiently as:

$$HV_d(p_i) = HV_d(p_{i+1}) + (p_{i,j} - p_{(i+1),j}) \times V[p_i, j]$$

This approach maintains a vector of bound values, b, which stores intermediate hypervolume calculations that can be quickly accessed and updated.

Now, with these improved properties for the HSO algorithm discussed above, these are summarised in this Pseudoalgorithm 2 below, which is an adaptation and concise version of the improved dimension-sweep algorithm shown below. Note that the pseudoalgorithm described below is Version 4 from Fonseca et al (2006)[71] which has not been directly described in detail in any of his existing papers.

Algorithm 2 Dimension-Sweep Algorithm (Version 4)(adapted from Version 3 of Fonseca et al(2006))

Input: d (dimensions), P (non-dominated points), r (reference point), L_i (sorted list by dimension i), len ($|L_{i-1}|$)
Output: $hvol$ (Hypervolume)
procedure H(i, L_i, r, len)
  if $i = 3$ **then**
    Call the specialized 3D hypervolume function (see Fonseca et al (2006)[71]); **return**
  end if
  Reset flags for all p in L_i
  $hvol \leftarrow 0$, $p \leftarrow nil(L_i)$
  while $prev_i(p) > b_i$ and $len > 1$ **do**
    $p \leftarrow prev_i(p)$, $b_j \leftarrow \min\{b_j, p_j\}$ for $j < i$
    Delete p from L_i, $len \leftarrow len - 1$, $q \leftarrow prev_i(p)$
  end while
  if $len > 1$ **then**
    $hvol \leftarrow V[prev_i(q), i] + H[prev_i(q), i] \cdot (q_i - prev_i(q))$
    $V[q, i] \leftarrow hvol$
  end if
  Call SkipDom(q, i, L_i, r, len)
  while $p \neq nil(L_i)$ **do**
    $hvol \leftarrow hvol + H[q, i] \cdot (p_i - q_i)$
    $b_i \leftarrow p_i$, $b_j \leftarrow \min\{b_j, p_j\}$ for $j < i$
    Reinsert p into L_i, $len \leftarrow len + 1$
    $q \leftarrow p$, $p \leftarrow next_i(p)$
    $V[q, i] \leftarrow hvol$
    Call SkipDom(q, i, L_i, r, len)
  end while
  $hvol \leftarrow hvol + H[q, i] \cdot (r_i - q_i)$
  return $hvol$
end procedure
procedure SkipDom(q, i, L_i, r, len)
  if flag$[q] \geq i$ **then**
    $H[q, i] \leftarrow H[prev_i(q), i]$
  else
    $H[q, i] \leftarrow H(i - 1, L_{i-1}, r, len)$
    if $H[q, i] \leq H[prev_i(q), i]$ **then**
      flag$[q] \leftarrow i$
    end if
  end if
end procedure

Complexity Analysis

In Version 3, given in the Fonseca's paper[71], the time complexity is $\mathcal{O}(n^{d-1})$ as this version recursively handles each dimension down to 3 dimensions. For each dimension i, it performs operations involving iterations over the set of points, resulting in an $\mathcal{O}(n)$ complexity for each level. The lack of the pruning and reuse of previous calculations means that algorithm often recalculates intermediate hypervolumes. Further, as it sorts and handles points for each dimension, this implies each recursive call is $\mathcal{O}(n)$ and since there are $d-1$ recursive levels, the total complexity is $\mathcal{O}(n^{d-1})$. Version 4 above, there is an explicit step where the algorithm checks if the hypervolume for a given subset has already been calculated and stored in $V[q,i]$. The algorithm additionally uses conditions **'flag[q]'** and **skipdom** (described in detail in Section 3C of Fonseca et al (2006)[71]) to decide which parts can be skipped or reused, leading to fewer recursive calls and decrease in time complexity with $\mathcal{O}(n^{d-2})$.

2.8.6 Expected Hypervolume Improvement (EHVI)

The algorithms discussed above HSO and ISA, offer efficient methods to compute the hypervolume. The significance of these algorithms extends beyond hypervolume calculation itself - they form the backbone of the Expected Hypervolume Improvement (EHVI)[72] acquisition function in MOBO.

EHVI uses the hypervolume as a key indicator of progress in the optimization process. Specifically, EHVI measures the potential improvement in the hypervolume when a new solution is proposed, guiding the optimization process toward regions in the objective space that provide the most significant Pareto-front expansion. The computational efficiencies offered by HSO and IDSA are critical here, as EHVI relies on frequent and accurate hypervolume calculations to inform decision-making.

By efficiently computing the hypervolume, these algorithms enable EHVI to function effectively in high-dimensional objective spaces, ensuring that the optimization process is both computationally feasible and accurate. The next section delves deeper into the concept of EHVI and how it leverages hypervolume to drive multi-objective optimization toward the best possible trade-offs.

Definition 9 ***Hypervolume Improvement (HVI):*** *Given a reference point $r \in \mathbb{R}^d$ and a set $P = \{y_1, y_2, \ldots, y_n\} \subset \mathbb{R}^d$, the hypervolume improvement when adding a point y to P is defined as:*

$$HVI(P, y) = HV(P \cup \{y\}) - HV(P) \tag{2.9}$$

where $HV(P)$ is the hypervolume of the region dominated by the set P with respect to the reference point r. The quantity $HVI(P, y)$ measures the contribution of point y to expanding the hypervolume dominated by the Pareto front.

When the reference point r needs to be emphasized, the hypervolume improvement can be denoted as $\Delta(P, y, r)$. This quantity is positive if y improves the Pareto front and contributes to the hypervolume. The improvement region is given by:

$$\Delta(y, P, r) = \lambda_d\{z \in \mathbb{R}^d \mid y \prec z, z \prec r \text{ and } \nexists q \in P : q \prec z\}$$

where λ_d denotes the Lebesgue measure in $\mathbb{R}^d$.

EHVI Definition

The EHVI extends the concept of HVI by accounting for uncertainty in the location of y. This extension is particularly relevant in the context of Bayesian optimization, where predictions are modeled using Gaussian random fields (GRFs) or Gaussian Processes (GPs)(1-dimensional GRF)[72].

Definition 10 ***EHVI:*** *Consider a vector of objective functions $y \in \mathbb{R}^d$ whose distribution is governed by a multivariate Gaussian process $\mathcal{N}(\mu(x), \Sigma(x, x'))$, where $\mu(x)$ is the mean vector and $\Sigma(x, x')$ is the covariance matrix determined by the covariance function $k(x, x')$. The EHVI for adding y to the Pareto front $\mathcal{P}$ is defined as:*

$$EHVI(\mu(x), \Sigma(x, x'), P, r) = \int_{\mathbb{R}^d} HVI(P, y) \cdot PDF_{\mu,\Sigma}(y)\, dy \tag{2.10}$$

where $PDF_{\mu,\Sigma}(y)$ is the probability density function of the multivariate Gaussian distribution defined by $\mu(x)$ and $\Sigma(x, x')$. This integral represents the expected increase in hypervolume over all possible realizations of y, weighted by their probability under the GRF model (described in Appendix 8.2.6).

Given the properties of a GRF (detailed in Appendix 8.2.6), each individual point $Y(x)$ follows a Gaussian distribution, and any subset of points also follows a multivariate Gaussian distribution. Specifically, the probability density function (PDF) for n points $Y(x_1), Y(x_2), \ldots, Y(x_n)$ is given by:

$$P(Y) = \frac{1}{(2\pi)^{n/2}|C|^{1/2}} \exp\left(-\frac{1}{2}(Y - \mu)^T C^{-1}(Y - \mu)\right) \tag{2.11}$$

where $|C|$ is the determinant of the covariance matrix and C^{-1} is its inverse.

In the context of EHVI, GRFs are relevant when considering uncertainty in predictions from surrogate models such as Gaussian Process Regression. The EHVI calculation integrates over the joint distribution of the objective functions, which under the assumption of Gaussian processes, is multivariate Gaussian. The integration in 2.10, given parameters $\mu(x)$ and $\Sigma(x, x')$ from Gaussian Process models, and Pareto-front approximation set $\mathcal{P}$, accounts for the expected improvement in hypervolume considering uncertainty in the predictions.

2.8.7 Numerical Integration of EHVI: Challenges and Complexity

The computation of EHVI requires integrating over the possible realizations of y, given the uncertainty in the predictions from Gaussian Process models. Given the dimensional complexity of the problem, the integration can be decomposed for clarity as follows:

$$EHVI(\mu, \sum, \mathcal{P}, r) = \int_{\mathbb{R}^d} \left(\int_{\mathbb{R}^{d-1}} \cdots \left(\int_{\mathbb{R}^1} HVI(\mathcal{P}, y) \cdot PDF_{\mu,\sum}(y) \delta y_d \right) \delta y_{d-1} \cdots \delta y_2 \right) \delta y_1 \tag{2.12}$$

The primary analytical challenges in computing the EHVI arises from the complexity of the integral itself, due to 3 reasons:

1. **Non-linearity of the HVI function:** The hypervolume $HV(\mathcal{P})$ improvement in Equation 2.8 depends non-linearly on y because of the union of hyperrectangles defined by $\mathcal{P} \bigcup \{y\}$ which cannot be easily expressed in closed form, especially in high dimensions.

2. **Multivariate Gaussian Distribution:** The distribution involves the quadratic form $(y-\mu)^T \sum^{-1} (y-\mu)$ which represents the Mahalanobis distance from y to the mean μ in the space defined by $\sum$. This is generally represented in Equation 2.11 This creates a highly nontrivial integral that is difficult to solve analytically.

3. **High Dimensionality:** The number of Pareto-dominating and dominated regions grows exponentially with d. This increases number of regions over which integration must be performed. The covariance matrix $\sum$ grows in size with d and the matrix inversion $\sum^{-1}$ becomes computationally expensive, scaling with $\mathcal{O}(d^3)$. The evaluation of the multivariate normal distribution involves calculating the determinant and inversion of $\sum$ which becomes increasingly difficult as d increases.

2.8.8 Monte Carlo Integration Method

To address these challenges, Monte Carlo integration was proposed by Emmerich et al(2006)[72] for a feasible approximation technique. Monte Carlo methods rely on generating a large number of random samples drawn from the Gaussian distribution and averaging the hypervolume improvement across these samples. This approach approximates the integral in 2.12. The Monte Carlo method breaks down the complexity by averaging the results over N samples y_i drawn from the multivariate Gaussian distribution $\mathcal{N}(\mu, \sum)$.

$$EHVI_{MC} = \frac{1}{N} \sum_{i=1}^{N} HVI(P, y_i)$$

where $y_i \sim \mathcal{N}(\mu, \sum)$ are independent samples from the Gaussian process models.

Monte Carlo methods offer a scalable solution for EHVI in high-dimensional spaces and are particularly suited for multi-objective optimization tasks where hypervolume computation becomes increasingly difficult. The key benefit of Monte Carlo integration is its ability to handle the non-linearity and high dimensionality in a flexible manner, although at the cost of requiring a potentially large number of samples for high precision.

In this chapter, we have covered the essential concepts of acquisition functions in Bayesian optimization, focusing on both single- and multi-objective optimization scenarios. We explored the transition from Expected Improvement (EI) to more sophisticated multi-objective acquisition functions, such as Expected Hypervolume Improvement (EHVI), which are necessary for balancing the exploration-exploitation trade-off across multiple objectives. We delved into computational methods such as the Hypervolume by Slicing Objectives (HSO) algorithm and its improvements, followed by the challenges posed by integrating EHVI analytically. To address these challenges, we discussed the Monte Carlo Integration method as a practical and scalable solution. This comprehensive review sets the stage for applying these advanced acquisition functions and optimization techniques in practical multi-objective optimization problems, ensuring efficient and robust search processes in high-dimensional spaces.

3 | Methodology

3.1 KERN-GP: Achieving Full Dimensionality of Molecular Fingerprints

Molecular fingerprints are often represented as high-dimensional and extremely sparse vectors, with only 1-2% non-zero elements. These vectors are commonly reduced in dimensionality to maintain computational efficiency. However, such reductions may lead to less accurate similarity measures. In contrast, `KERN_GP` enables the exact calculation of Tanimoto coefficients (see Equation 2.4 of Section 2.7.3)) across the full dimensionality of molecular fingerprints, ensuring greater precision in molecular similarity calculations.

Traditional implementations, particularly in frameworks like PyTorch, typically reduce the dimensionality to ranges between 1024-4096[19][7][8][48]], primarily because PyTorch expects dense matrices, which would result in excessive memory usage and computational overhead when handling sparse, high-dimensional vectors.

By leveraging the full dimensionality and using exact Tanimoto similarity calculations, `KERN_GP` allows us to overcome the inefficiencies associated with dense matrix operations in PyTorch. This is particularly beneficial when working with large-scale molecular datasets, where dimensionality reduction could compromise the accuracy of downstream predictions.

The memory savings from using sparse representations instead of dense matrices are significant. As demonstrated in Table 3.1, dense arrays can consume up to 2029 MB for Klekota-Roth fingerprints[77], while their sparse representation reduces this to 23 MB - a 88.2x memory saving (Adamczyk & Ludynia, 2024)[78]. These savings are particularly impactful during tasks such as virtual screening, where full-scale fingerprints are required.

Fingerprint name	**Dense array size (MB)**	**Sparse array size (MB)**	**Memory savings**
Klekota-Roth	2029	23	88.2x
FCFP	855	15	57x
Physiochemical Properties	855	17	50.3x
ECFP	855	19	45x
Topological Torsion	855	19	45x

Table 3.1: Memory usage of fingerprints in dense and sparse versions (Adamczyk & Ludynia (2024)

Given that `KERN_GP` allows for exact Tanimoto coefficient calculations without dimension

reduction, it ensures that all available information in the fingerprint is utilized. This provides a more faithful similarity measure compared to approaches that reduce dimensionality, potentially improving the accuracy of downstream predictions in molecular property estimation.

That said, it is important to note that while this implementation detail improves efficiency and accuracy, it does not fundamentally alter the results when compared to reduced-dimensional approaches. The added precision is incremental rather than transformative. Therefore, while `KERN_GP` is a more exact method, the overall outcomes remain in line with conventional approaches, with the main advantage being the ability to scale up without sacrificing fidelity.

In summary, the key advantage of `KERN_GP` lies in its ability to handle full-dimensional molecular fingerprints in an exact manner, without succumbing to the inefficiencies associated with dense matrix operations. For large-scale molecular datasets, this allows for more precise similarity calculations, which is particularly useful in tasks like virtual screening and similarity searching. However, it is important to recognize that this improvement primarily addresses the technical challenge of full-dimensionality rather than dramatically altering the predictive performance. What was developed is as demonstrated in Algorithm 3 is provided below here.

Minimal Kernel-only GP Package for Fingerprints - Full Dimensionality

Algorithm 3 Kernel-Only Gaussian Process Inference for Fingerprints Full Dimensionality

Input: Training data $\mathbf{x}_i$, $\mathbf{y}_i$, GP hyperparameters a, s, Query SMILES $\mathbf{x}_q$
Output: Predictive means and variances for query molecules

Compute fingerprints for $\mathbf{x}_i$ and $\mathbf{x}_q$
Compute kernel matrices K_{ii}, K_{iq}, K_{qq}
Perform **Cholesky decomposition** of $K_{ii} + \frac{s}{a}I$ using $\mathbf{L} = \text{Cholesky}(K_{ii} + \frac{s}{a}I)$ (see Appendix 8.2.1)
Compute Negative Log Marginal Likelihood (NLML) (see Section 2.6.5):
  a) Data fit term: Calculate $-\frac{1}{2a}\mathbf{y}_i^\top \mathbf{L}^{-1}\mathbf{y}_i$
  b) Complexity penalty term: Calculate $-\frac{1}{2}\log\det(\mathbf{L}) - \frac{\log a}{2}|\mathbf{L}|$
  c) Combine to get NLML: NLML = data fit + complexity penalty + constant term
Compute predictive mean: $\mu_*(\mathbf{x}_q) = K_{iq}^\top (K_{ii} + sI)^{-1}\mathbf{y}_i$
Compute predictive variance:
  a) Covariance adjustment: $\mathbf{V} = \text{Triangular\_Solve}(\mathbf{L}, K_{iq}^\top)$
  b) Variance: $\sigma^2(\mathbf{x}_q) = K_{qq} - \mathbf{V}^\top \cdot \mathbf{V}$
Return: Predictive means $\mu_*(\mathbf{x}_q)$ and variances $\sigma^2(\mathbf{x}_q)$

Additionally, it is good to note that the base package of this `KERN_GP` can be modified to any base kernel we want. The GP model can also be modified to other GP variations, such as sparse GPs, and in our work, we coined the multi-output GP under this framework of `KERN_GP`, as Multi-output Tanimoto Kernel GPs (MOTKGP).

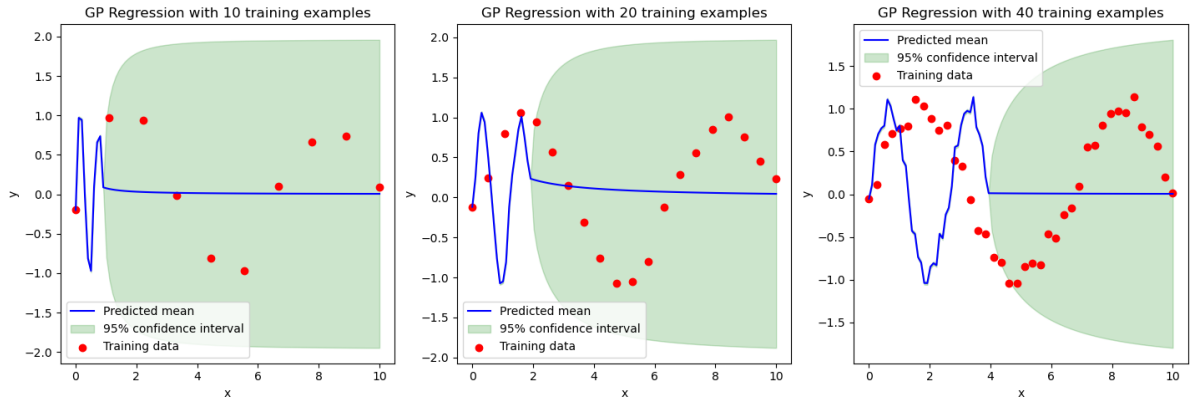

Figure 3.1: **Single-output Gaussian Process regression with a full-dimensional Tanimoto Kernel as used in `KERN_GP` for exact molecular similarity calculations:** The blue line represents the mean of posterior predictive distribution and the green shaded region represents the 95% confidence interval based on the model's variance estimates. As the number of training examples increases, the mean function begins to exhibit more complex behaviors to match the observed data, but the confidence region does not shrink in the same way as seen in RBF kernels.

3.2 State-of-the-art: GP-MOBO

In this work, we present a novel approach to multi-objective Bayesian optimization (MOBO), specifically tailored for molecular optimization using count fingerprints. The key innovation lies in leveraging Tanimoto Kernel Gaussian Processes (GPs), which were previously defined in Section 3.1 (`KERN-GP`). This GP-based MOBO framework models each molecular objective independently, unlike previous multi-objective works, which often propose complex models that assume correlations between outputs. Our approach provides a more scalable and computationally efficient solution, especially for large-scale molecular optimization tasks.

Unlike scalarization methods that combine multiple objectives into a single number, multi-objective problems more realistically capture the trade-offs between objectives without implicitly enforcing a specific preference. Scalarization methods, while commonly used, assume that trade-offs (such as preferring a=1, b=2 over a=2, b=1) are predetermined. However, real-world problems often involve uncertainties in these trade-offs, leading to a desire to explore the Pareto frontier directly, which captures all possible non-dominated solutions.

Despite the simplicity of this framework, it has not been investigated thoroughly in molecular optimization. In this thesis, we propose a straightforward algorithm that models each objective independently with Gaussian Processes (GPs), while utilizing a standard multi-objective acquisition function (EHVI) for efficient exploration and optimization. As a result, our **GP-MOBO** approach ensures scalability and computational efficiency, making it particularly well-suited for tasks that require optimizing multiple molecular properties simultaneously, such as virtual screening and molecular design. We will explain thoroughly and concisely what this algorithm entails in the subsequent sections below.

Our Novel Algorithm for Multi-Objective Bayesian Optimization (GP-MOBO)

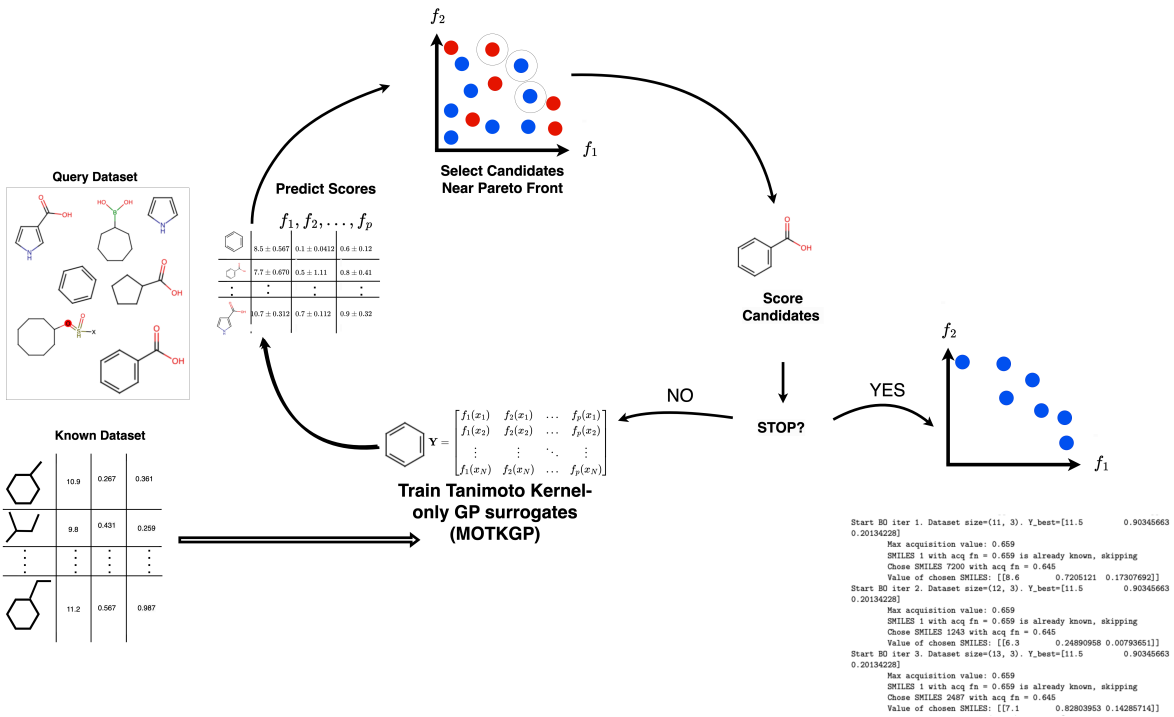

Figure 3.2: **Overview of our GP-MOBO algorithm: Combining independent Tanimoto Kernel GP surrogates with EHVI to guide molecular optimization, identifying optimal candidates near the Pareto frontier.** The process begins with an initial dataset of molecular structures (SMILES), where independent Gaussian Processes (GPs) for each objective are trained using the Tanimoto kernel. The figure illustrates the iterative optimization process. In each iteration, candidate molecules are selected based on their proximity to the Pareto front, scored for their objective values, and appended to the dataset. The GP is retrained after each iteration, refining the Pareto front until a stopping criterion is met. This iterative loop continues until the trade-offs between objectives are optimized, yielding non-dominated solutions.

The core of our methodology is illustrated in Figure 3.2, which provides an overview of the **GP-MOBO** process. This approach begins with an existing dataset of molecular SMILES/structures, and independent GP surrogate models are trained for each molecular objective. These GPs leverage the Tanimoto kernel, allowing us to capture molecular similarities across the full dimensionality of the fingerprint space, as implemented in the

`KERN_GP` framework. This is consequently extended to MOTKGP, which is the multi-output version defined below here in Definition 11.

Definition 11 (Multi-Output Tanimoto Kernel Gaussian Processes (MOTKGP)) *For each molecular property $f_1, f_2, \ldots, f_k$, we define independent Gaussian Process (GP) models. The joint predictive distribution is multivariate Gaussian with a diagonal covariance matrix:*

$$\begin{bmatrix} f_1(m) \\ f_2(m) \\ \vdots \\ f_k(m) \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} \mu_1(m) \\ \mu_2(m) \\ \vdots \\ \mu_k(m) \end{bmatrix}, \begin{bmatrix} \sigma_1^2(m) & 0 & \ldots & 0 \\ 0 & \sigma_2^2(m) & \ldots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \ldots & \sigma_k^2(m) \end{bmatrix} \right)$$

Each property f_j is modeled with an independent GP:

$$f_j \sim \mathcal{GP}(\mu_j, K_j(x_i, x_q))$$

where μ_j is the mean function and $K_j(x_i, x_q)$ is the Tanimoto kernel, representing molecular similarity between fingerprints x_i and x_q. The kernel is given by:

$$k(x, x') = a_j T(x, x')$$

with a_j as the kernel amplitude. Predictions for each property are returned as:

$$\vec{\mu}(m) = [\mu_1(m), \ldots, \mu_k(m)] \quad \textit{and} \quad \vec{\sigma}^2(m) = [\sigma_1^2(m), \ldots, \sigma_k^2(m)]$$

Following from the MOTKGP Definition 11, we now transition to the core methodology of the Bayesian Optimization method of our proposed **GP-MOBO** algorithm. This algorithm (defined in Algorithm 4) leverages Bayesian optimization (BO) within the context of multi-objective optimization, where each molecular property is modeled independently. In Bayesian Optimization, the idea is to balance exploration (finding diverse regions of the chemical space) with exploitation (focusing on regions that are known to yield high-quality results). In multi-objective optimization (MO), the objective is not to find a single optimal solution but to approximate the Pareto front, which consists of non-dominated solutions, representing optimal trade-offs between multiple objectives.

Problem Setup for GP-MOBO

Given a dataset $\mathcal{D}_0 = \{(x_i, y_i)\}_{i=1}^n$ where x_i are the SMILES strings and y_i are the corresponding objective function values for each property $f_1, f_2, ...f_D$, we aim to sequentially add new data points to $\mathcal{D}_0$ such that we refine our estimate of the Pareto front.

We define each objective function as shown in MOTKGP framework above, and the kernel function which is the Tanimoto kernel (otherwise known as MinMax for count-based fingerprints), evaluates molecular similarity between fingerprints x_i and x_q.

Algorithm 4 Our Proposed Novel Algorithm: **GP-MOBO**

Input: Dataset $\mathcal{D}_0 = \{(\mathbf{x}_1, \mathbf{y}_1), \ldots, (\mathbf{x}_n, \mathbf{y}_n)\}$, GP hyperparameters $\{\mu_j, a_j, s_j\}$, EHVI acquisition function α, max reference point R_{max}, scale λ, number of iterations n_{iter}
Output: Pareto-front approximation $\mathcal{P}$ and optimized SMILES
Initialize μ points $\{\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(\mu)}\}$ from $\mathcal{D}_0$
Train independent MOTKGP model $p(\hat{f})$ on $\mathcal{D}_0$
Evaluate the initial set of μ points $\mathbf{y}(\{\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(\mu)}\})$
Store evaluated points in $\mathcal{D}_0$
Compute initial Pareto front $\mathcal{P}_0$ from non-dominated subset of $\mathcal{D}_0$
for $t = 1$ to n_{iter} **do**
  Compute reference point $R \leftarrow$ infer_reference_point$(\mathcal{P}_{t-1}, R_{\text{max}}, \lambda)$
  Calculate hypervolume $HV_{\text{current}} \leftarrow$ compute_hypervolume$(\mathcal{P}_{t-1}, R)$
  for each $\mathbf{x_q} \in$ Query SMILES **do**
    Predict mean $\mu_{\mathbf{x_q}}$ and variance $\sigma^2_{\mathbf{x_q}}$ using MOTKGP models
    Compute EHVI $\alpha(\mathbf{x_q})$
  end for
  Select next candidate $\mathbf{x}_* \leftarrow \arg\max_{\mathbf{x_q}} EHVI_{\mathbf{x_q}}$
  Acquire new objective $\mathbf{y}_* \leftarrow \mathbf{y}(\mathbf{x}_*)$
  Update dataset $\mathcal{D}_t \leftarrow \mathcal{D}_{t-1} \cup \{(\mathbf{x}_*, \mathbf{y}_*)\}$
  Update Pareto front $\mathcal{P}_t$ as non-dominated subset of $\mathcal{D}_t$
  Recalculate hypervolume $HV_{\text{new}} \leftarrow$ compute_hypervolume$(\mathcal{P}_t, R)$
  if budget exhausted **then**
    return $\mathcal{P}_t$ ▷ Terminate if budget is exhausted
  end if
end for
return Pareto-front approximation $\mathcal{P}_t$ and optimized SMILES

Bayesian Optimization in GP-MOBO

The Bayesian Optimization loop starts by predicting objective values using the trained GP surrogates. The Expected Hypervolume Improvement (EHVI) acquisition function guides the selection of new query molecules, balancing exploration and exploitation.

The EHVI function here (discussed in Section 2.8.6) is designed to compute the expected gain in the hypervolume (HV) of the Pareto front. This improvement select candidates

that will best expand the Pareto front in subsequent iterations. Formally, for each query point (SMILES) x_q, EHVI is computed as:

$$EHVI(x_q) = \mathbb{E}[\max(HV_{x_q} - HV_{\text{current}}, 0)]$$

where HV_{x_q} is the hypervolume of the Pareto front where new query SMILES x_q is added, and HV_{current} is current hypervolume of Pareto front.

As illustrated in Algorithm 4, the **GP-MOBO** process starts by initializing with a dataset $\mathcal{D}_0$ with known SMILES and their corresponding objective values y_i. The non-dominated solutions are computed from $\mathcal{D}_0$ forming the initial approximation of the Pareto front $\mathcal{P}_0$. At each iteration t, the GP models predict the mean and variance $\mu_{x_q}, \sigma^2_{x_q}$ for each query molecule x_q. These predictions are fed into the EHVI acquisition function to score each query molecule for its potential to improve the Pareto front. The molecule with the highest EHVI score is selected and its true objectives are evaluated and the dataset $\mathcal{D}_t$ is updated. The Pareto front is recalculated and hypervolume is updated accordingly. The loop continues until the computational budget is exhausted or the desired Pareto front is sufficiently refined. The final Pareto front is returned, providing the optimal trade-offs across all molecular objectives.

The core mathematical operations - training of independent GPs, hypervolume computation, and acquisition function evaluation are all defined within the Background Chapter in Sections 2.8, 2.8.2, 2.8.3, 2.8.4, and 2.8.6, where we discuss how this entire optimization pipeline is computationally efficient. It is also good to acknowledge that our implementation of EHVI, has been tested with the readily available test cases that have been provided by BoTorch and have passed with the same numerical accuracy. The test case results are as shown in the Appendix 8.4.2.

Now that we have thoroughly explained the design and technicalities of the **GP-MOBO** algorithm, including how it efficiently handles multi-objective optimization using independent GP surrogates and EHVI as the acquisition function, we can proceed to experimental validation. In the following chapter, we will benchmark our **GP-MOBO** model against single-objective approaches, highlighting the benefits and trade-offs of modeling objectives independently. Through a series of experiments, we aim to demonstrate the scalability, computational efficiency, and accuracy of our method when applied to real-world molecular optimization tasks.

4 | Experimental Design

4.1 Datasets

For benchmarking GP-MOBO, we use two widely referenced datasets: DockSTRING and GUACAMOL. Both are frequently used in molecular optimization tasks and are considered standard benchmarks in the literature [1][19].

- DockSTRING[79]: This dataset offers a robust framework for docking and binding affinity prediction tasks. We use a subset of first 10000 SMILES, where for the toy MPO setup, we begin with a set of 10 initial `known_SMILES`, and GP-MOBO will sample `query_SMILES` from the remaining dataset over 20 Bayesian Optimization iterations.
- GUACAMOL[15]: This dataset focuses on de novo molecular design, targeting drug-likeness, novelty, and synthetic accessibility. For our setup, we also use a subset of 10000 SMILES from the `guacamol_v1_train.smiles` file. We trained all the benchmarking models and **GP-MOBO** with 10 initial `known_SMILES`, and the models will sample from the $\sim$ 9980 `query_SMILES` for the next 20 Bayesian Optimization iterations. Further, for assessing the GP's prediction, we use `guacamol_v1_valid.smiles`

4.2 Oracles

In our experimental design, oracles serve as evaluation functions that mimic real-world drug discovery tasks. Each oracle computes specific molecular properties or docking scores for a given SMILES string x and returns the corresponding objective values y. Our utility functions handle these evaluations, ensuring that only valid values are processed. These evaluations are implemented in our utility functions (see example in Appendix 8.4.1) , which handle multiple objectives for each dataset, filtering out any NaN values and ensuring consistency in the evaluation. For the toy MPO setup (see Section 4.3 below), we defined 3 distinct objectives to be optimized concurrently. For more complex, real-world drug discovery tasks, we specifically chose 3 GUACAMOL's MPO tasks, Fexofenadine MPO, Amlodipine MPO, and Perindopril MPO. These GUACAMOL MPO setup is clearly demonstrated after our Toy MPO Setup. These MPO definitions are provided in Appendix 8.3.

4.3 Toy Multi-Property Objective (MPO) Setup

To validate our GP-MOBO model, we devised a toy experiment using a set of toy multi-property objectives from the DockSTRING dataset. The chosen objectives were selected to challenge our model with diverse molecular properties, aiming to balance and optimize conflicting criteria simultaneously. The selected objectives are:

$$f_1(m) = -\text{DockingScore}(\text{PPARD}, m) \tag{4.1}$$

$$f_2(m) = \text{QED}(m) \tag{4.2}$$

$$f_3(m) = -\text{sim}(m, \text{celecoxib}) \tag{4.3}$$

The definitions of these objectives are as described in Appendix 8.3. These objectives target molecules that bind to the PPARD protein, are drug-like, and are structurally similar to the reference molecule, celecoxib. Though this setup does not represent a realistic drug discovery task, it serves as a demonstrative example for evaluating our methodology.

For each optimization step, we begin with a small set of molecules 10 where all objective values have been observed (with some Gaussian noise). These molecules (represented as SMILES) form our `known_SMILES` list. The corresponding objective values are stored in the array `known_Y`, which has the shape $(N, 3)$ where N represents the number of molecules and 3 corresponds to the objective values of f_1, f_2 and f_3. To compute these objective values, we have the `evaluate_objectives()` function which ensures the handling of `NaN` values and provides consistency in input-output mapping. As the GP model is trained independently for each objective (see Definition 11), we have the specific hyperparameters used for this setup here, for 3 independent objectives in Table 4.1. For

GP Hyperparameter	f_1	f_2	f_3
GP Mean (μ)	0.0	0.0	0.0
GP Noise (s)	$1e^{-4}$	$1e^{-4}$	$1e^{-4}$
GP Amplitude (α)	1.0	1.0	1.0

Table 4.1: **GP Hyperparameters for Seed Prototype Model of GP-MOBO implemented on DockSTRING dataset:** These hyperparameters were specifically chosen for comparison with the original GP-BO model which have these hyperparameters (Tripp & Hernandez-Lobato(2024))

the Bayesian Optimization (BO) process, we utilize Expected Hypervolume Improvement (EHVI) as the acquisition function to guide the selection of new SMILES strings that will

improve the current objectives $f_1(m)$,$f_2(m)$ and $f_3(m)$. As computing EHVI in closed form can be computationally intractable for multi-objective optimization, we approximate it using Monte Carlo (MC) integration. In our setup, we employ 1000 Monte Carlo samples to estimate the EHVI at each step of the optimization. These MC samples are drawn from the posterior distribution of the objectives, providing an efficient and scalable way to explore the objective space and identify promising candidates. By doing so, we ensure that the BO process can effectively balance the trade-offs between objectives while progressively improving the molecular properties of the generated SMILES strings over 20 iterations.

As a result, 20 chosen SMILES will then be evaluated using `evaluate_objectives()` function which will return their f_1, f_2 and f_3 values respectively.

4.4 GP-MOBO on GUACAMOL's MPO Setup

For the GUACAMOL MPO Setup, we extend our GP-MOBO evaluation to real-world drug discovery tasks by focusing on three distinct multi-property objectives (MPOs) from the GUACAMOL dataset (Table 4.2).

MPO Task	**Mean**	**Scoring Function(s)**	**Modifier**
Fexofenadine MPO	geom	sim(fexofenadine, AP)	Thresholded(0.8)
		TPSA	MaxGaussian(90, 2)
		logP	MinGaussian(4, 2)
Amlodipine MPO	geom	sim(amlodipine, ECFP4)	none
		number rings	Gaussian(3, 0.5)
Perindopril MPO	geom	sim(perindopril, ECFP4)	none
		number aromatic rings	Gaussian(2, 0.5)

Table 4.2: Selected Guacamol's MPO Tasks for Benchmarking GP-MOBO performance with GP BO

In this section here, we illustrate how GP-MOBO tackles this MPO task. As an example, we focus on Fexofenadine MPO (Table 4.2), the selected objectives are:

$$f_1(m) = -\text{sim}(m, \text{Fexofenadine}, AP) \tag{4.4}$$

$$f_2(m) = \text{TPSA}(m) \tag{4.5}$$

$$f_3(m) = \log P(m) \tag{4.6}$$

It's important to note that we modified the original Fexofenadine MPO oracle, which originally combined the objectives above here by scalarizing into a single multi-property score with the geometric mean. We split it into **3 separate objectives** for these experiments. This modification allows us to evaluate and optimize each molecular property individually, increasing interpretability of model's performance on specific attributes. For this task, the GP hyperparameters are as standardized from the Toy MPO setup (Table 4.1), and the number of Monte Carlo (MC) samples to approximate the EHVI at each step maintains the same at $N = 1000$. Now, with this setup, we expect, for this Fexofenadine task, for 20 BO iterations, 20 new SMILES will be selected that maximize these properties independently. These 20 chosen SMILES will then be evaluated using `evaluate_objectives()` function which will return their f_1, f_2 and f_3 values respectively.

This approach is similarly applied to the other two tasks, Amlodipine MPO and Perindopril MPO, each with their respective scoring functions and modifiers, as outlined in Table 4.2. For Amlodipine MPO and Perindopril MPO, both MPOs will return f_1 and f_2. This section sets the stage for the evaluation and comparison of GP-MOBO's performance across these tasks with 3 other derivatives of the single-objective GP-BO model by Tripp et al(2021).

4.5 Benchmarking GP-MOBO

The purpose of our experimental design is to systematically compare the performance of the GP-MOBO algorithm against existing single-objective Bayesian optimization models in both low and full fingerprint dimensionalities. This design evaluates how single- and multi-objective optimization techniques perform on a variety of molecular optimization tasks, using acquisition functions such as UCB, EI, and EHVI. The workflow is divided into three main stages: initial SMILES selection, fingerprint dimensionality preprocessing, and Bayesian optimization.

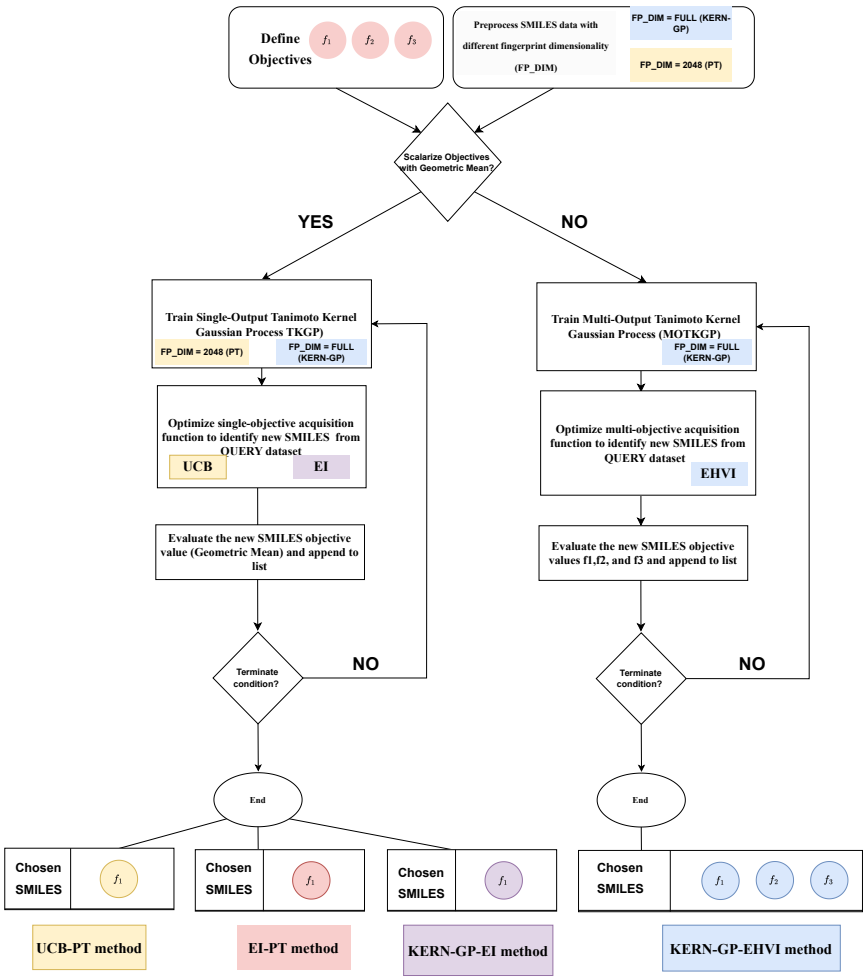

Figure 4.1: **Experimental Design Workflow:** This design compares SMILES values using both single- and multi-objective optimization. We define objectives f_1, f_2, f_3, followed by preprocessing SMILES data with either full fingerprint dimensionality (`FP_DIM = FULL`) or reduced default dimensionality(`FP_DIM = 2048`). Whether scalarization (geometric mean) is applied, either a single-output GP (TKGP) or multi-output GP (MOTKGP) is trained. Acquisition functions EI, UCB, & EHVI are then used to identify and evaluate new SMILES until termination. Final SMILES are selected via UCB-PT (GP-BO), EI-PT, KERN-GP-EI, or KERN-GP-EHVI (GP-MOBO).

Initial SMILES selection

For both the Toy MPO setup and the GUACAMOL MPO Tasks, we randomly selected 10 initial `known_SMILES`. These initial molecules serve as a starting point for the optimization process. For each optimization task, we evaluate the selected SMILES based on their scalarized MPOs (for single-objective cases) or independent objective functions (for multi-objective cases), represented by f_1, f_2 and f_3. This ensures that the optimization process begins from the same baseline across all experiments. The initial objective values are consistent in both setups, reducing any bias and allowing for fair comparisons.

Count Fingerprint Dimensionality Processing

The selected SMILES are converted into molecular fingerprints using two approaches:

- Full Dimensionality (`FP_DIM = FULL`): We employ our `KERN_GP` model to preserve the full dimensionality of count molecular fingerprints. This configuration allows the model to leverage the complete information encoded in the molecular structures.
- Reduced Dimensionality (`FP_DIM = 2048`): We apply the default reduced fingerprint size (2048 bits), a common configuration used in most PyTorch (PT)-based model implementations.

This step enables us to investigate how variations in fingerprint dimensionality affect the performance of Bayesian optimization across different molecular complexities.

Bayesian Optimization Setup

The optimization process is divided into two branches based on the task's objectives:

Single-Objective Optimization: In these cases, the objectives are scalarized using the geometric mean of f_1, f_2, and f_3, reducing the optimization task to a single scalar value. We employ two acquisition functions:

- **Upper Confidence Bound (UCB):** Optimizes for the upper bound of confidence intervals around expected values.
- **Expected Improvement (EI):** Focuses on maximizing the expected improvement over the current best-known SMILES.

These configurations are referred to as **UCB-PT** and **EI-PT** respectively when processed with `FP_DIM = 2048`, and as **KERN-GP-EI** when processed with full-dimensional fingerprints.

Multi-Objective Optimization (GP-MOBO): In contrast to scalarization, the GP-MOBO framework optimizes multiple objectives simultaneously, leveraging the EHVI acquisition function to balance trade-offs between objectives f_1, f_2, and f_3. This approach allows the model to efficiently explore the Pareto front of the optimization task, ensuring a diverse set of optimal molecules.

4.5.1 Model Training

The GP-MOBO model is trained using the Tanimoto kernel on full-dimensional **count** fingerprints (MinMax Kernel (Section 2.7.4), allowing the model to directly optimize for multiple objectives in parallel. After each Bayesian Optimization iteration, new SMILES strings are selected, and their objective values are updated in the `known_SMILES` list. The same process is applied to the single-objective models, with UCB, EI, and scalarized objectives. Training proceeds for **20 BO iterations** across all setups, ensuring a comprehensive and consistent evaluation. We additionally evaluated the GP's predictions with negative log predictive density (NLPD) (see Appendix 8.4.3).

4.5.2 Comparison

To fairly benchmark the performance of the GP-MOBO model, we compare it against the baseline GP-BO models (Tripp et al., 2024)[2], which employ UCB-based acquisition functions and reduced fingerprint dimensionality (FP_DIM = 2048). Additionally, we extend the original GP-BO to include the KERN-GP-EI method, where we maintain full fingerprint dimensionality for comparison. This setup allows us to isolate the effects of dimensionality and acquisition function on model performance. Furthermore, as the original GP BO that optimizes the GP acquisition function with Graph GA methods in an inner loop [19], this was modified to just sample from the query dataset (`query_SMILES`) for both setups to make a fair comparison.

4.5.3 Evaluation Procedure

These methods are evaluated in terms of how effectively they balance multiple objectives or optimize scalarized objectives in the BO process. This setup ultimately allows us to systematically benchmark the GP-MOBO model and compare its performance in terms of 20 chosen BEST SMILES from `query_SMILES`.

To conclude our experimental design, we outline the evaluation procedure across all methods. For the single-objective methods (UCB-PT, EI-PT, KERN-GP-EI), the initial

SMILES were scalarized using the geometric mean of the three objectives (f_1, f_2, f_3) to guide their selection process. To evaluate how these single-objective acquisition functions perform in terms of approaching the Pareto front, the 20 SMILES chosen by these methods were re-evaluated independently, as performed with the multi-objective EHVI method. This allowed us to investigate how well the single-objective approaches balance the conflicting objectives when compared to the EHVI method.

Additionally, we sought to determine if the GP-MOBO method could select better SMILES than the single-objective methods. For this, the 20 SMILES chosen by each method were scalarized again, calculating the geometric mean of their respective f_1, f_2, and f_3 values to provide a holistic measure of performance. The formula for calculating the geometric mean across the three objectives is given by:

$$\text{Geometric Mean} = (\Pi_{i=1}^{n} x_i)^{\frac{1}{n}} = (f_1(m) \times f_2(m) \times f_3(m))^{\frac{1}{3}}$$

In the results that follow, we present a detailed comparison of the performance of these methods. Specifically, we show how the geometric mean of f_1, f_2, and f_3 was used to assess the selected SMILES for the single-objective methods and how the independent evaluations of these objectives in the multi-objective EHVI method led to more diverse and balanced selections. This comparative analysis provides critical insights into the strengths and trade-offs between the different optimization strategies, facilitating a thorough evaluation of the GP-MOBO model.

5 | Results and Analysis

5.1 Performance of our GP-MOBO Over Current GP BO Methods in Toy MPO Setup

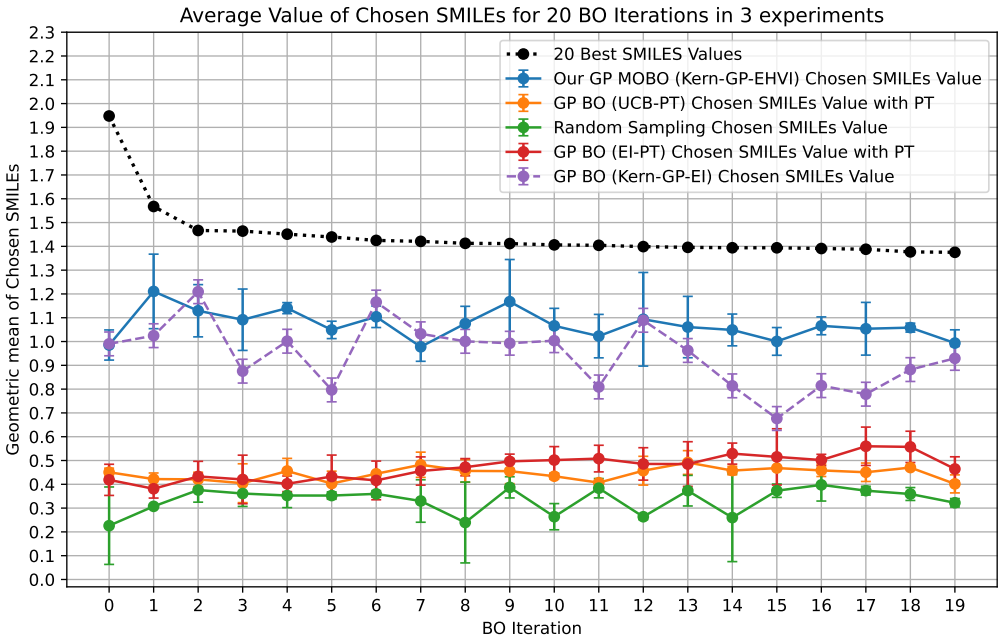

Figure 5.1: **Comparison of our GP-MOBO (KERN-GP-EHVI) and GP BO (UCB and EI with PyTorch(PT)) Models on Chosen SMILES Values across 20 BO Iterations on Toy-MPO DockSTRING dataset:**The error bars represent the standard deviation across three independent experimental runs for each model, indicating the variability in the performance of chosen SMILES values during the optimization process.

Our GP-MOBO model, implemented with the KERN-GP Package (detailed in Section 3.1), demonstrates a significant advantage over the GP BO models implemented by Tripp et al (2024)[2]. The KERN-GP package enables superior optimization performance, particularly when employing Expected Hypervolume Improvement (EHVI) acquisition function. This approach not only outperforms traditional PyTorch-based GP BO implementations but also consistently provides higher values of chosen SMILES across 20 BO iterations, showcasing robustness and efficacy of our methodology.

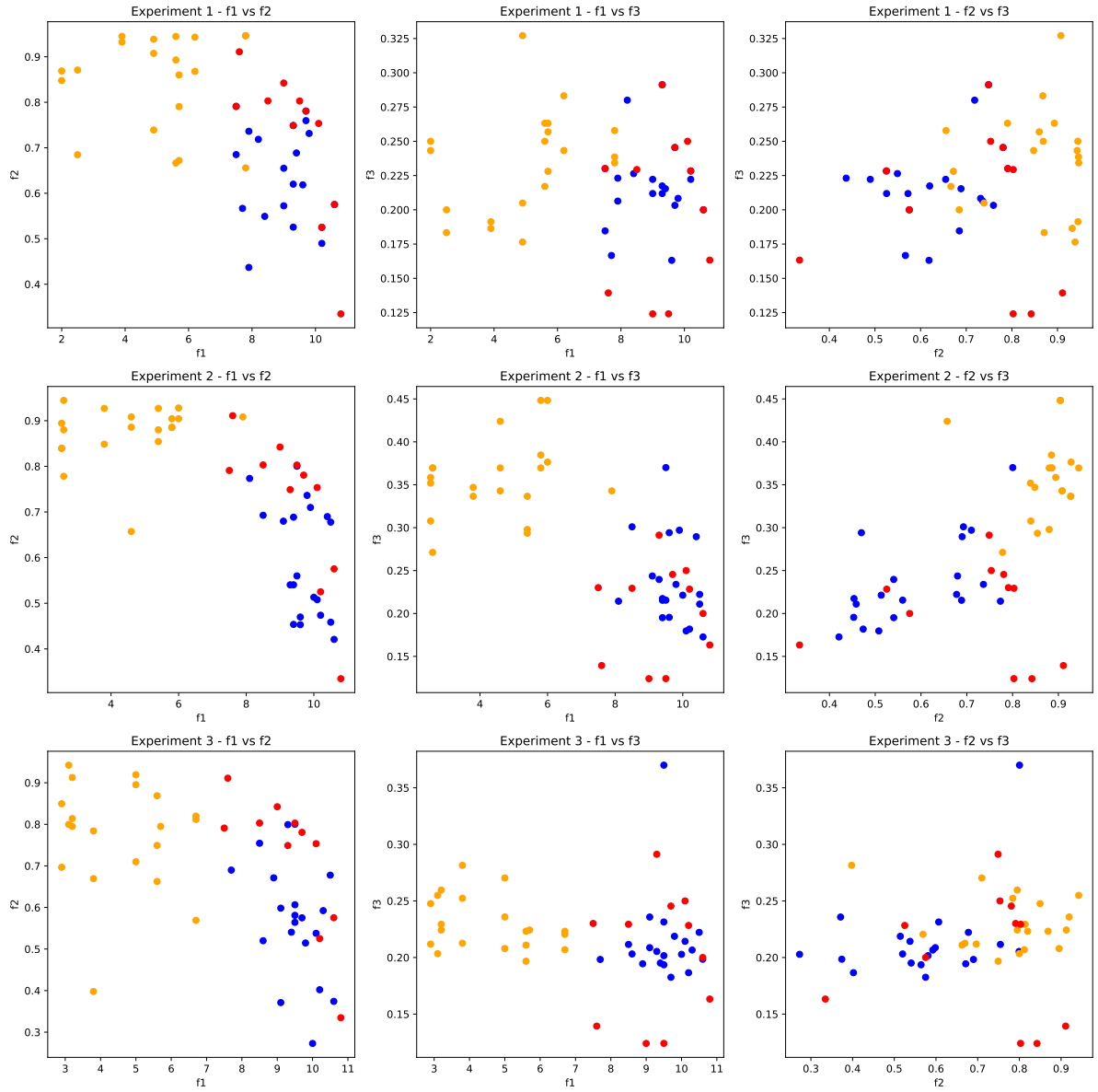

Figure 5.2: **Comparison of Pareto Front Clustering Between Our GP-MOBO (Kern-GP-EHVI) Model and GP BO (UCB-PT) Across Three Experiments:** The performance of our GP-MOBO model (KERN-GP-EHVI) is contrasted with the GP BO (UCB-PT)(Tripp et al (2024)) approach in terms of how closely the selected points cluster around the Pareto optimal points. Each subplot represents a pairwise comparison of the objectives f_1, f_2, f_3. The blue points represent the results from our GP-MOBO model, the orange points represent the GP BO (UCB-PT) results, and the red points denote the Pareto optimal points.

Across all three experiments, it is evident that the points selected by our GP-MOBO model consistently cluster closer to the Pareto front compared to the GP BO (UCB-PT) method. This clustering indicates that our model is more effective at identifying solutions that achieve a balanced trade-off between the multiple objectives. The enhanced proximity to the Pareto front illustrates the superior exploration-exploitation balance achieved by our model, driven by maximizing the EHVI. This leads to a more nuanced and accurate optimization process, especially in complex multi-objective landscapes.

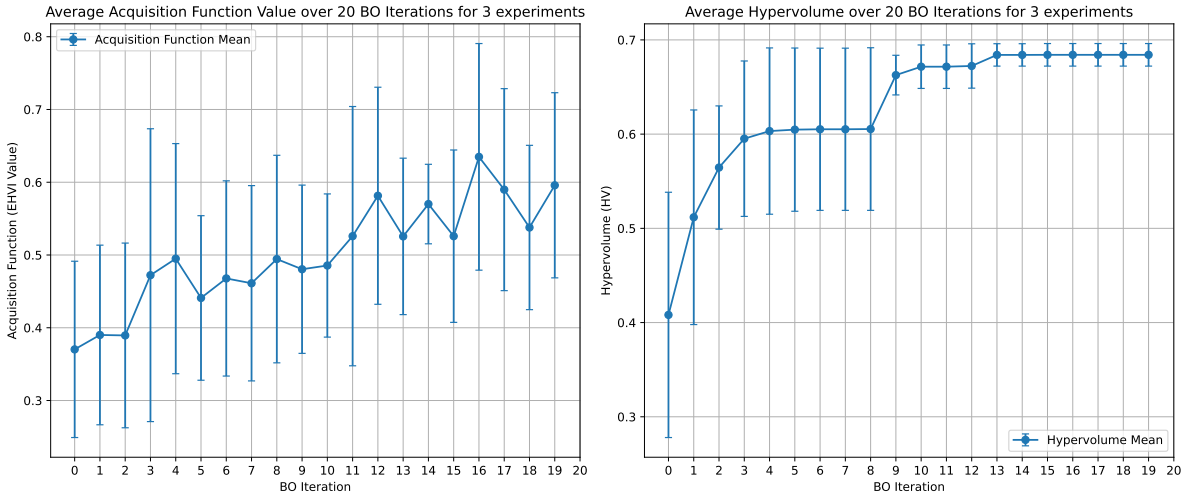

Figure 5.3: **Average EHVI Acquisition Function and Hypervolume Values for GP-MOBO across 20 BO iterations:** The left plot illustrates the average acquisition function value (EHVI) over 20 BO iterations, an indication that GP-MOBO's optimization process is working. The right plot shows average hypervolume values demonstrating that GP-MOBO is converging towards the Pareto front, as iterations increases. Error bars represent the standard deviation across three independent experimental runs for GP-MOBO with 10 random `known_SMILES`, indicating the variability in the performance of chosen SMILES values during the optimization process.

In 3 experiments, GP-MOBO's EHVI acquisition function, as illustrated in Figure 5.3, demonstrates a clear convergence pattern over 20 BO iterations. EHVI increases steadily indicating the optimization process is effectively identifying better candidate SMILES. This behaviour aligns with the behaviour observed in Figure 5.1, where the geometric mean values of the chosen SMILES of GP-MOBO levels off close to the dataset best compared to the other methods, which are the 20 best SMILES evaluated in the dataset(see Appendix 8.5.2). This suggests that GP-MOBO method, particularly with EHVI acquisition function, is refining its search near the dataset's best possible values early on in the process.

5.2 Guacamol MPO Tasks

5.2.1 Guacamol's Fexofenadine MPO Task (3 Objectives)

Figure 5.4: **Fexofenadine MPO Task: Comparison of the average value of chosen SMILES across 20 Bayesian Optimization (BO) iterations for different methods: KERN-GP-EHVI, KERN-GP-EI, EI-PT, UCB-PT, and Random Sampling.** The error bars represent the standard deviation across three independent experimental runs with 10 random initial `known_SMILES` for each model, indicating the variability in the performance of chosen SMILES values during the optimization process.

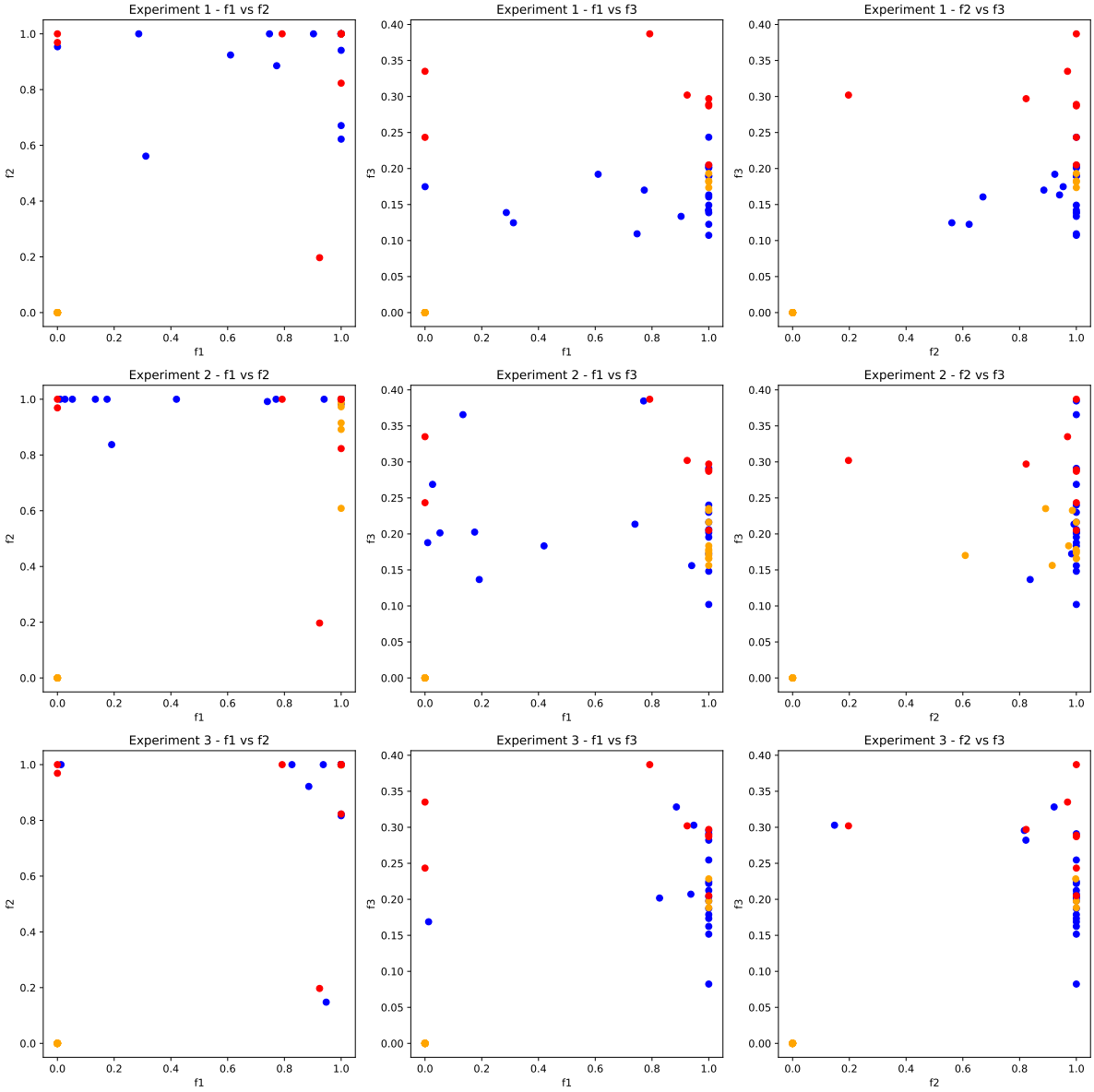

Figure 5.5: **Fexofenadine MPO Pareto Plots: Pareto plots for the Fexofenadine MPO optimization problem showing the objective values (f_1, f_2, f_3) for three different optimization experiments. The blue points represent the SMILES strings selected using the KERN-GP-EHVI approach, while the yellow points are those selected using the UCB-PT approach. The red points indicate the Pareto optimal solutions.** The distribution of blue points closer to the Pareto frontier across multiple plots indicates that KERN-GP-EHVI achieves a more diverse and effective exploration of the objective space, leading to better coverage of the Pareto front compared to UCB-PT.

5.2.2 Guacamol's Amlodipine MPO Task (2 Objectives)

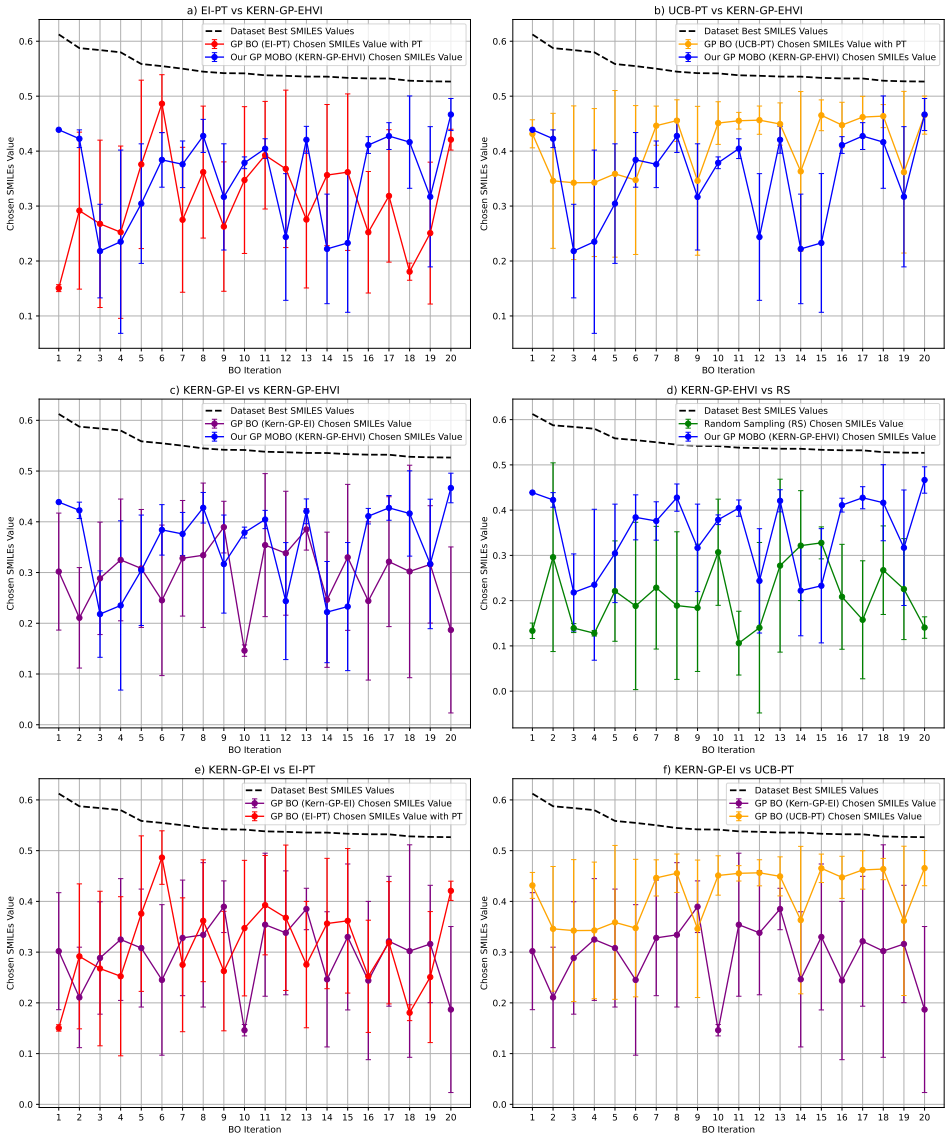

Figure 5.6: **Amlodipine MPO Task: Comparison of the average value of chosen SMILES across 20 Bayesian Optimization (BO) iterations for different methods: KERN-GP-EHVI, KERN-GP-EI, EI-PT, UCB-PT, and Random Sampling.** The error bars represent the standard deviation across three independent experimental runs with 10 random initial `known_SMILES` for each model, indicating the variability in the performance of chosen SMILES values during the optimization process.

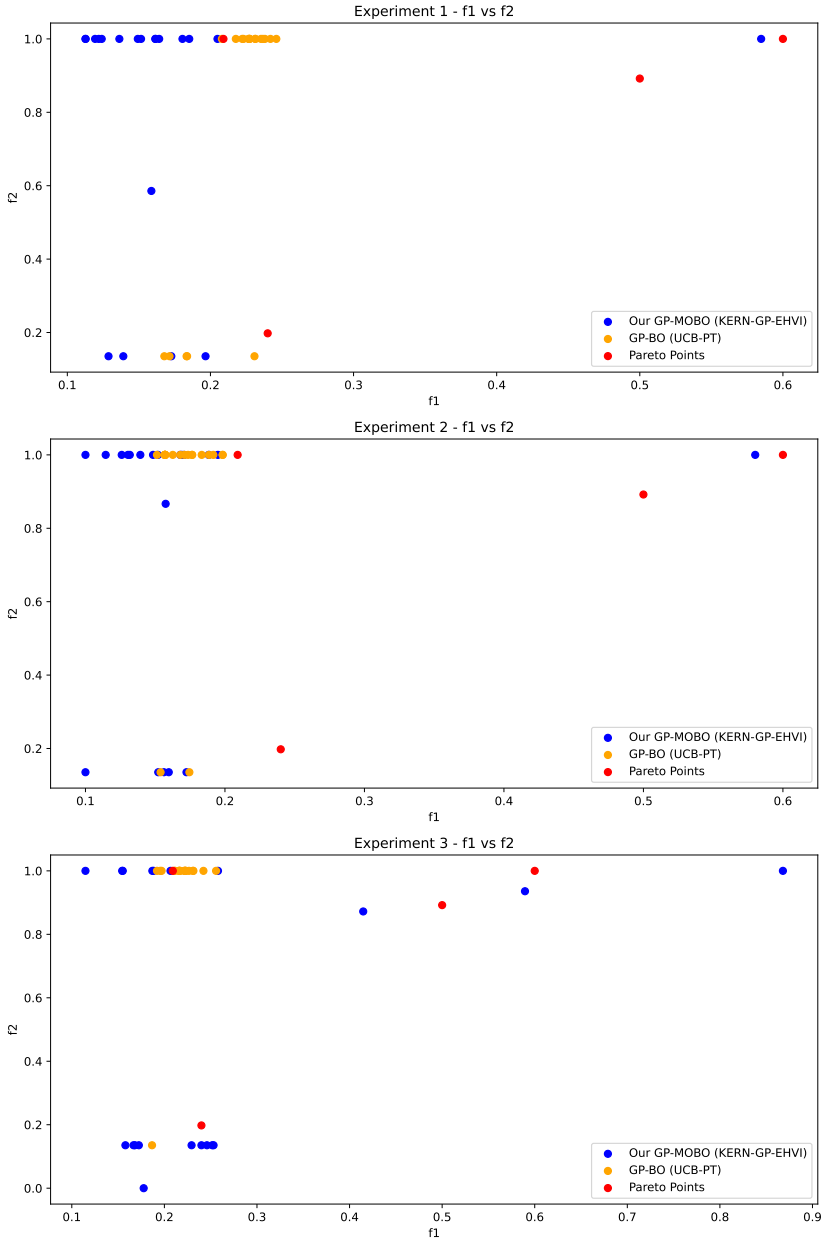

Figure 5.7: **Amlodipine MPO Pareto Plots: This figure presents pairwise plots of the objectives f_1 and f_2 across 3 experiments for Amlodipine MPO:** The blue points correspond to the SMILES strings selected using the KERN-GP-EHVI approach, while the yellow points are selected using the UCB-PT approach. The red points represent the Pareto-optimal solutions, which are non-dominated with respect to the other points.

5.2.3 Guacamol's Perindopril MPO Task (2 Objectives)

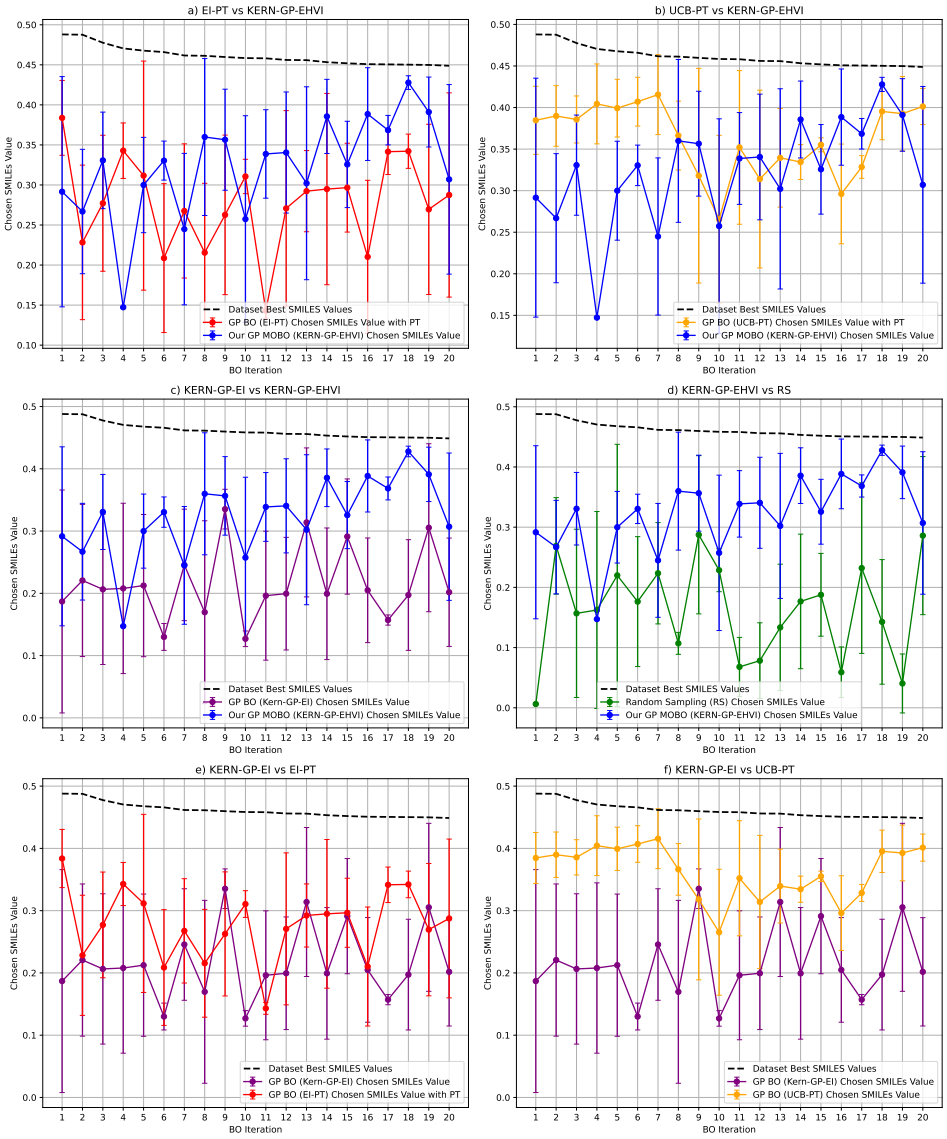

Figure 5.8: **Perindopril MPO Task: Comparison of the average value of chosen SMILES across 20 Bayesian Optimization (BO) iterations for different methods: KERN-GP-EHVI, KERN-GP-EI, EI-PT, UCB-PT, and Random Sampling.**The error bars represent the standard deviation across three independent experimental runs with 10 random initial `known_SMILES` for each model, indicating the variability in the performance of chosen SMILES values during the optimization process

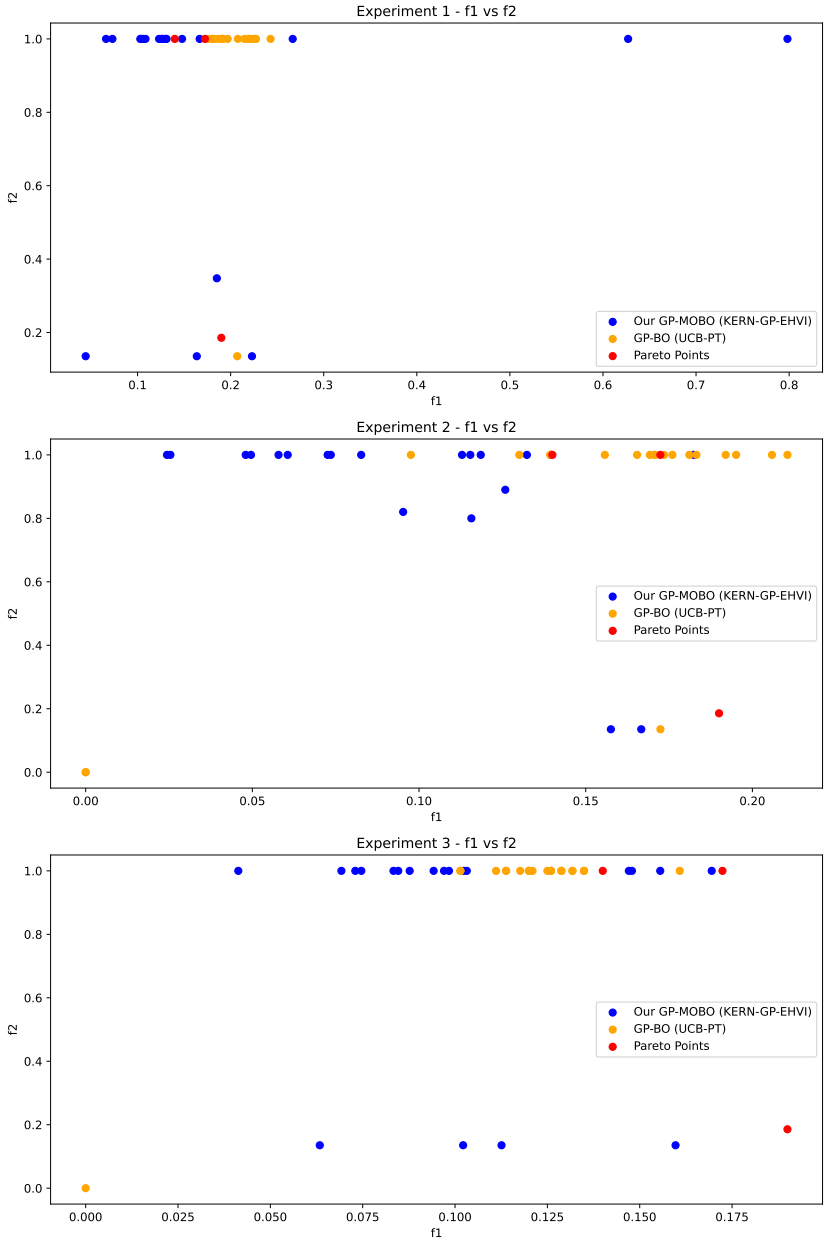

Figure 5.9: **Perindopril MPO Pareto Plots: This figure presents pairwise plots of the objectives f_1 and f_2 across 3 experiments for Perindopril MPO.** The blue points represent the SMILES strings selected using the KERN-GP-EHVI approach, while the yellow points are selected using the UCB-PT approach. The red points indicate the Pareto-optimal solutions.

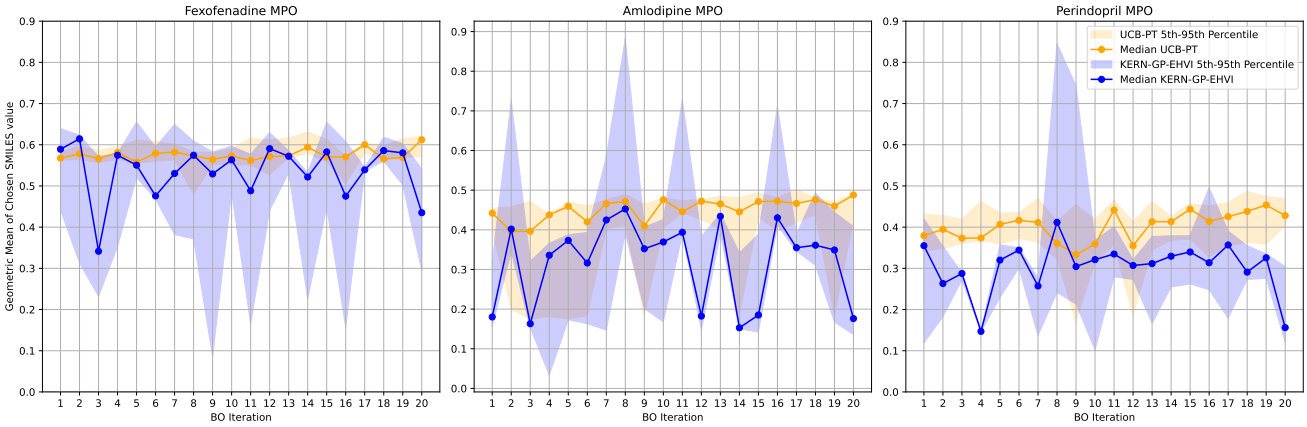

Figure 5.10: **Comparison of Geometric Mean of Chosen SMILES Values Across Three MPO Tasks for UCB-PT and KERN-GP-EHVI Methods Over 20 BO Iterations: The performance comparison between the UCB-PT (orange) and KERN-GP-EHVI (blue) methods across three multi-objective optimization (MPO) tasks: Fexofenadine, Amlodipine, and Perindopril.** The geometric mean of the chosen SMILES values is plotted across 20 Bayesian Optimization (BO) iterations. The shaded regions represent the 5th to 95th percentile range, providing insight into the variability of each method, while the solid lines denote the median values.

The KERN-GP-EHVI (GP-MOBO) method demonstrates broader exploration of the chemical space for all 3 MPO tasks, as evidenced by the wider shaded areas compared to UCB-PT (GP-BO). This indicates that KERN-GP-EHVI captures a more diverse range of SMILES, leading to better coverage of the Pareto front. However, the median values for KERN-GP-EHVI are generally lower than those for UCB-PT, suggesting that while KERN-GP-EHVI explores more, UCB-PT consistently identifies higher-value SMILES on average. This trade-off highlights KERN-GP-EHVI's strength in exploring the chemical space, which may be advantageous when discovering diverse molecules is crucial, while UCB-PT excels when maximizing SMILES value is prioritized.

To assess the performance of GP-MOBO in comparison to UCB-PT, we visualized the distribution of selected SMILES in Figures 5.5, 5.7 and 5.9. A key observation is that GP-MOBO tends to select SMILES closer to the Pareto front across all plots, especially for Fexofenadine MPO, indicating higher diversity and exploration. GP-MOBO shows better coverage of the Pareto front for Fexofenadine, results in 2D cases indicate similar trends for both methods. While UCB-PT excels in identifying higher objective values, GP-MOBO provides superior diversity.

6 | Discussion

6.1 Why our GP-MOBO over GP BO?

The performance across the 2-dimensional problems, such as the final two MPO tasks, showed limited differences between GP-MOBO and GP-BO. However, in the case of Fexofenadine MPO, Figures 5.5 and 5.10 indicate a clear advantage for GP-MOBO. While GP-MOBO did not consistently select better SMILES than GP-BO in every instance, it demonstrated a broader exploration of the chemical search space, identifying more diverse molecules. This characteristic allows GP-MOBO to uncover molecules with higher objective scores, which GP-BO often misses. The potential to explore a wider search space with higher-scoring molecules makes GP-MOBO a strong candidate for optimizing molecular properties where traditional models like GP-BO fall short.

In the following sections, we delve into the specific reasons why GP-MOBO outperforms GP-BO in certain setups, and why it should be considered as a preferred model in generative chemistry tasks.

6.1.1 Why Our GP-MOBO Outperforms GP BO in Toy MPO, but Not in Real-World Drug Discovery GUACAMOL MPO Tasks?

In the Toy MPO Setup in Figure 5.1, the performance of GP-MOBO consistently shows advantages over GP BO, even from the first Bayesian Optimization (BO) iteration. We investigated the first 10 initial `known_SMILES` to determine whether there was a difference in the training dataset, indicating a less fair outcome. Prompting further, in the Toy-MPO setup, all the values for `known_SMILES` for GP-MOBO and GP-BO were within a similar range for all repeat experiments. An example of `known_SMILES` and their respective f_1, f_2, and f_3 (`known_Y`) are provided for the GP-MOBO setup is provided as well as `known_SMILES` and their scalarized geometric mean for the single-objective cases (`known_Y`) are provided in Table 8.1 and 8.2 (Appendix 8.5.1) respectively. Therefore, there is a different explanation as to why GP-MOBO picks better SMILES than GP-BO from BO iteration 1.

From the GUACAMOL setup (Section 5.2), we notice that this trend does not extend to real-life drug discovery tasks. To understand this difference, we need to explore the role of the fingerprint dimensionality and its interaction with the specific objectives of the Toy MPO setup.

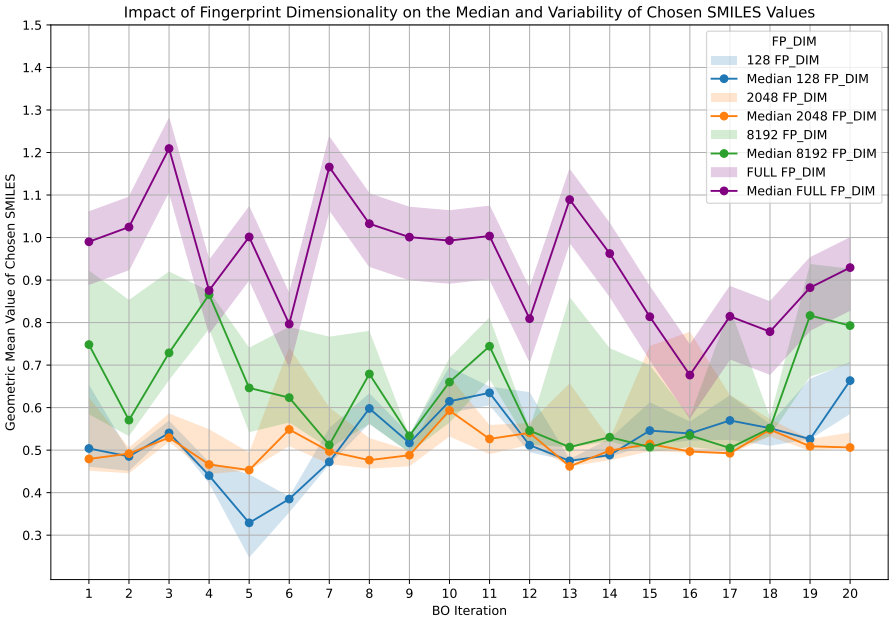

Figure 6.1: **Relationship of Fingerprint Dimensionality(`FP_DIM`) with the Median and Variability of Chosen SMILES:** This compares the performance of different fingerprint dimensionalities (128, 2048, 8192, and FULL) by showing the median value and variability (25th to 75th percentile) of the geometric mean of chosen SMILES across 20 Bayesian Optimization (BO) iterations for Toy-MPO Objectives (Section 4.3).

Figure 6.1 clearly demonstrates that, in the Toy MPO environment, higher fingerprint dimensionality (`FP_DIM = FULL`) correlates strongly with improved geometric mean values for chosen SMILES. In contrast, lower dimensionalities (such as 2048 or 8192) do not capture the variability as effectively, leading to lower performance. However, we know from the GUACAMOL setup that `FP_DIM = 2048` is sufficient for GP BO to sufficiently perform similarly to GP-MOBO.

This result suggests that certain substructures, represented by the higher-dimensional fingerprint features, play a significant role in determining the objective values in the Toy MPO setup.

Correlation Between Fingerprint Features and Performance of GP BO

A key hypothesis emerging from our results is that the Toy MPO objectives benefit from specific substructures that are better captured by higher-dimensional fingerprints.

These substructures, represented by the full fingerprint dimensionality directly correlate with the objectives of the Toy MPO. As a result, the models GP-MOBO and GP-BO (purple and blue lines in Figure 5.1), when using `KERN_GP`, with full dimensionality, is able to explore and exploit these substructures more effectively, achieving better results compared to GP BO.

Conversely, the standard dimensionality of 2048 (often used in real-world tasks for most models in molecular optimization) appears insufficient to fully capture the substructures, leading to the diminished performance in our toy MPO setup. The initial SMILES in Table 8.1 and 8.2 (Appendix 8.5.1) provide further evidence the only variable that has been changed in this setup is the fingerprint dimensionality and the single and multi-objective acquisition functions. This trend is consistent with the hypothesis that GP-MOBO benefits from the greater expressiveness provided by the full fingerprint representation.

In real-world tasks, where substructures are more complex and diverse, the full fingerprint dimensionality does not seem to confer to the same advantage. The variance in the complexity of drug-like molecules means that the relatively simplistic correlation between fingerprint dimensionality and objectives in the Toy MPO may not hold. In this regard, GP BO's approach, which operates effectively across multiple acquisition functions, shows competitive performance in the drug discovery task due to its robustness to different types of chemical representations and objective spaces. This would be left for further work to identify the correlation between the fingerprint dimensionality.

6.1.2 Diversity on the Pareto Front

The ability to explore diverse regions of the chemical latent space is critical in multi-objective optimization, particularly in drug discovery tasks where structural diversity can lead to novel compounds with improved or unique properties. In this context, GP-MOBO (KERN-GP-EHVI) demonstrates a clear advantage over GP BO (UCB-PT) by more effectively diversifying the chemical space it explores. We observe this habit in the Pareto plots in Figures 5.5, 5.9, 5.7, where yellow points (UCB-PT) are clustered around a similar region, not exploring the chemical space in contrast to the blue points (GP-MOBO) which explores the region and are fairly closer to the Pareto points (red).

As illustrated in Figure 6.2, the SMILES selected by GP-MOBO show higher diversity compared to those chosen by GP-BO. The top row displays the three selected SMILES from GP-MOBO, with similarity scores of 0.1583, 0.1892, and 0.1642 relative to Fexofenadine. These lower similarity scores indicate that GP-MOBO explores a broader range of chemical structures, capturing a wider variety of substructures while maintain-

ing marginally better structural alignment to the target compound (Fexofenadine).

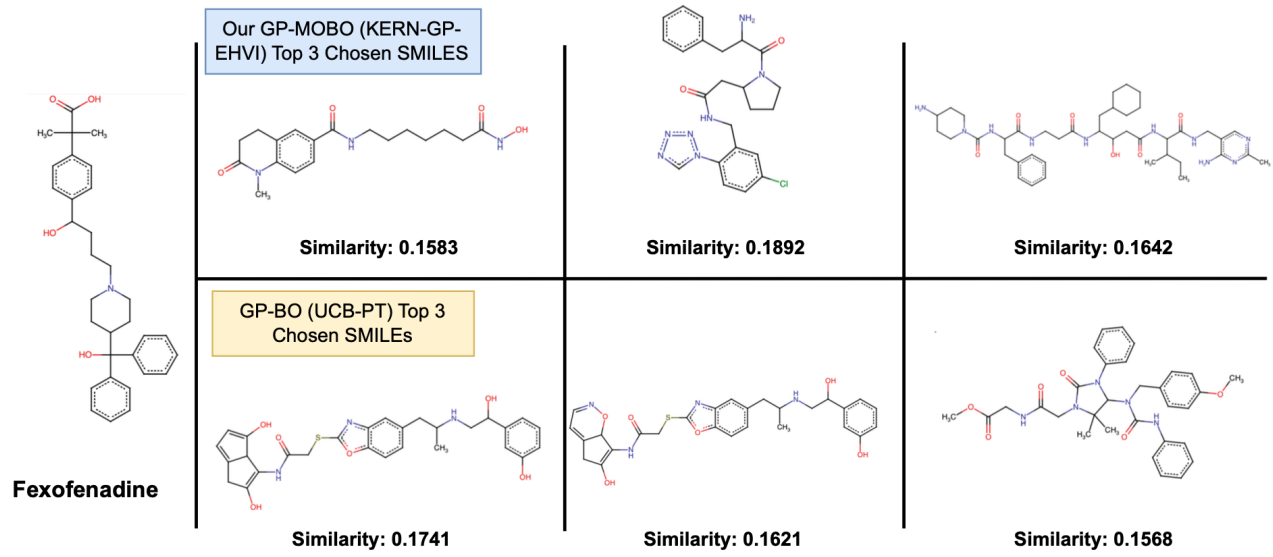

Figure 6.2: **Chemical Structure Comparison of Top 3 Chosen SMILES for Fexofenadine MPO by KERN-GP-EHVI and UCB-PT:** KERN-GP-EHVI (top row), selected SMILES strings with higher diversity and marginally better structural similarity (`TPSA_score`) to Fexofenadine (similarity scores: 0.1583, 0.1892, 0.1642) compared to UCB-PT, whose chosen SMILES (bottom row) exhibit lower diversity and slightly lower similarity scores (0.1741, 0.1621, 0.1568).

In contrast, GP BO (UCB-PT), shown in the bottom row of Figure 6.2, selects SMILES with higher similarity scores (0.1741, 0.1621, 0.1568). These SMILES are more closely clustered in the chemical space, indicating less diversity in the molecules GP-BO explores. This clustering behavior is consistent with the optimization strategy of UCB, which tends to focus on exploitation rather than exploration, resulting in a narrower chemical search space.

The greater diversity in the GP-MOBO results arises from its utilization of the Expected Hypervolume Improvement (EHVI) acquisition function, which balances both exploitation and exploration. This balance allows GP-MOBO to search unexplored regions of the chemical space more effectively, leading to the selection of structurally diverse molecules. The structural diversity of the selected SMILES is not only beneficial in terms of achieving better multi-objective performance but also increases the likelihood of discovering

novel compounds with optimized properties for multiple objectives, such as Fexofenadine's MPO.

The observed behavior in this study is consistent with previous experiments using Amlodipine and Perindopril MPOs, where GP-MOBO exhibited greater diversity in the selected SMILES compared to GP-BO. This highlights GP-MOBO's superior capability in exploring broader chemical landscapes, a key factor in drug discovery applications where novelty and diversity are highly sought after.

6.2 GP-MOBO's Prediction Evaluation

After assessing the diversity of molecules generated by GP-MOBO (KERN-GP-EHVI), it is crucial to evaluate the model's predictive capabilities. While diversity on the Pareto front is key to exploring novel chemical structures, the effectiveness of these predictions directly impacts the success of optimization. Specifically, the Gaussian Process (GP) model plays a critical role in guiding the selection of SMILES through its predicted means and uncertainty quantification.

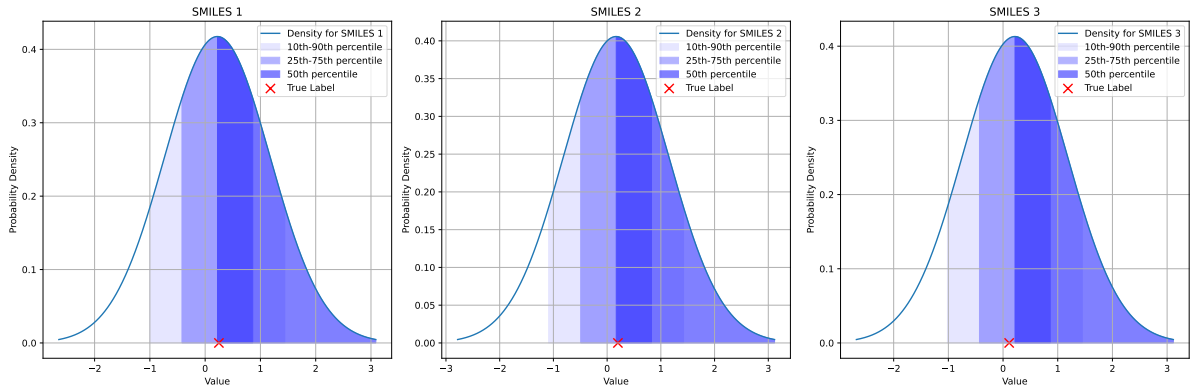

Figure 6.3: **Predictive Distributions of the Gaussian Process (GP) Model Across Fexofenadine MPO from Section 5.2.1 for Selected SMILES:** The shaded areas represent the 10^{th}, 25^{th}, and 50^{th} percentiles of the predicted distributions, while the red crosses mark the observed objective values (`KNOWN_Y`) of Fexofenadine MPO.

Figure 6.3 presents the predictive distributions for three selected SMILES from the Fexofenadine MPO task. These distributions help us understand how well the GP model predicts multi-objective outcomes and how closely the predicted means align with the observed values (`KNOWN_Y`). By analyzing the variance associated with each prediction,

we can gauge the model's confidence, which is essential for making informed decisions in multi-objective optimization.

Table 6.1 further reinforces this by comparing the actual objective values (`KNOWN_Y`) with the GP's predicted means and variances, as well as the Negative Log Predictive Density (NLPD)[80], a performance metric that penalizes overconfident or under-confident predictions. Lower NLPD values indicate that the GP model is well-calibrated, suggesting that its predictions are reliable and the associated uncertainties are appropriately sized (see NLPD Definition in Appendix 8.4.3).

Table 6.1: Table presents 3 SMILES from GUACAMOL's validation set (`guacamol_v1_valid.smiles`) corresponding to the molecules analyzed in Figure 6.3, along with their experimentally determined Fexofeandine MPO objective values (`KNOWN_Y`) and the GP model's predicted means and variances, and performance metric for GP's prediction (NLPD).

SMILES String	KNOWN_Y	GP Mean	GP Variance	NLPD
CCCC(=O)NNC(=O)Nc1ccccc1	0.2489	0.2205	9.1320e-01	0.877
CC(=O)NC1CCC2(C)C(CCC3(C)C2 C(=O)C=C2C4C(C)C(C) CCC4(C)CCC23C)C1(C)C(=O)O	0.2008	0.1669	9.6686e-01	0.903
CC(=O)NC(C)Cc1ccc(C#Cc2ccnc (N3CCCC(F)C3)n2)cc1	0.1118	0.2159	9.3302e-01	0.890

In this way, evaluating GP-MOBO's predictive accuracy is integral to understanding the model's overall performance in optimizing multi-objective tasks. The balance between exploration and exploitation not only depends on generating diverse SMILES but also on the GP's ability to accurately predict how those molecules will perform across objectives. This evaluation provides insight into the reliability of the GP predictions and opportunities for further model refinement.

6.3 Monte Carlo Integration Error

We concluded the GP-MOBO's prediction evaluation by demonstrating that the model provides good predictions with minimal uncertainty for chosen SMILES. A vital part of GP-MOBO's performance, especially when dealing with high-dimensional chemical spaces, is how accurately it estimates improvements across multiple objectives. This brings us to the importance of Monte Carlo (MC) integration methods in the Expected Hypervolume Improvement (EHVI) computation (discussed in Section 2.8.6).

The accuracy of EHVI is essential in the success of our GP-MOBO algorithm, espe-

cially when selecting molecules in high-dimensional chemical spaces. The MC integration method plays a pivotal role in this estimation, as it is used to approximate the EHVI, which directly influences the search for promising candidates. As Bayesian optimization relies on EHVI to guide the optimization process towards the Pareto front, the reliability of this estimate is crucial.

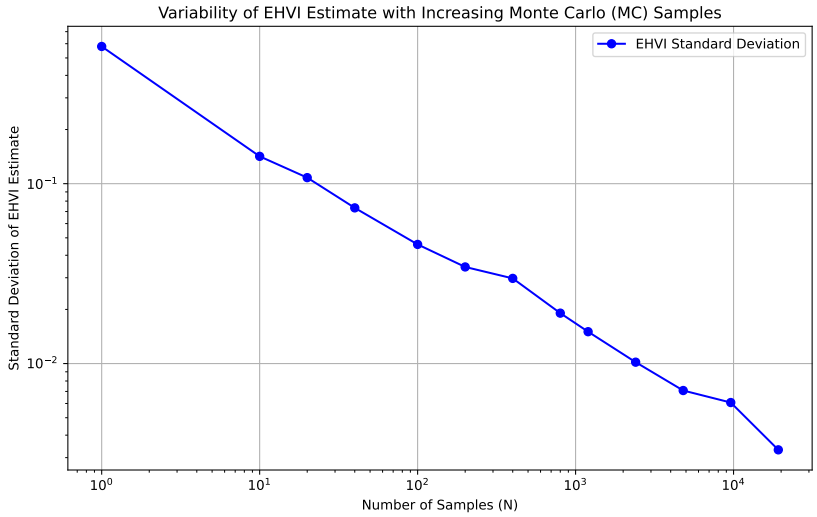

Figure 6.4: **Relationship between the number of Monte Carlo samples (N) and the variability of the Expected Hypervolume Improvement (EHVI) estimate in a Gaussian Process (GP) model:** This convergence is important in the context of Bayesian Optimization, where accurate estimation of EHVI is critical for selecting the most promising candidates in high-dimensional spaces.

As we increase the number of MC samples, the accuracy of EHVI improves, reducing noise and providing a clearer signal for selecting optimal candidates. However, this improvement comes at the cost of computational efficiency. To halve the error, we need to quadruple the number of MC samples, as demonstrated by the error scaling in Figure 6.4. Thus, a trade-off exists between computational effort and accuracy. In our experiments, we balanced this trade-off to ensure that the MC sample size provided a reasonable estimate of EHVI without excessively increasing computational time. The variability decreases as the number of MC samples increases, which is expected due to the central limit theorem as outlined by Lepage(1978)[81]. Specifically, the standard deviation of the EHVI estimate decreases proportionally to the inverse square root of the number of samples ($1/\sqrt{N}$). This convergence ensures that the optimization process remains reliable even when navigating high-dimensional chemical spaces.

By utilizing Monte Carlo integration effectively, we ensure that our GP-MOBO model reliably explores the Pareto front, especially in complex chemical spaces where exact analytical methods would be computationally prohibitive. This emphasizes the robustness of the GP-MOBO framework in handling real-world drug discovery tasks, making it an essential tool in the optimization of multi-objective problems. mWe leave for future work for testing different number of MC samples to further improve GP-MOBO's performance.

6.4 Limitations and Further Work

Bayesian Optimization (BO), especially when dealing with high-dimensional fingerprint vectors, poses significant computational challenges. The exact Gaussian Processes (GPs) used in this work scale poorly with the size of the dataset, exhibiting a computational complexity of $\mathcal{O}(N^3)$ and memory requirements as $\mathcal{O}(N^2)$, where N is the number of data points. Such limitations make GPs less feasible for large-scale problems, especially in high-dimensional drug discovery applications[82].

To mitigate these limitations, Sparse Gaussian Processes (SGPs)[83] offer a viable alternative by approximating the full GP using a smaller set of inducing points or variables, as outlined by Michalis Titsias (2009)[83]. SGPs can summarize the dataset efficiently while maintaining computational feasibility. Variants of SGPs, including Variational Fourier Features (VFF)[84] and other inducing point methods, are commonly used to reduce the computational burden without sacrificing significant predictive performance. This makes SGPs particularly suited for high-dimensional input spaces such as molecular fingerprints, which are often sparse in nature. By leveraging the sparse nature of fingerprints, SGPs help reduce memory usage and accelerate computations, allowing for larger datasets to be processed.

Despite these advances, challenges still remain. For instance, in our GP-MOBO, the Improved Dimension-Sweep Algorithm (IDSA)[71] was used to reduce the time complexity to $\mathcal{O}(n^{d-2} \log n)$. However, this algorithm for hypervolume computation still faces scalability issues when the number of non-dominated points n increases, or when dealing with higher dimensions. Furthermore, the recursive nature of the algorithm can lead to memory inefficiency in certain cases, particularly when processing large datasets. Additionally, the algorithm's performance is sensitive to the order in which objectives are processed, potentially leading to sub-optimal pruning, inflating computational costs.

Future work should explore the implementation of faster and more scalable EHVI algorithms. Additional methods like Multi-Objective Max-Value Entropy Search (MESMO)[85]

and Sequential Greedy Optimization [86] approaches have been shown to achieve comparable or superior performance while significantly reducing wall times in other optimization tasks. These methods provide promising avenues for further research, particularly for cases where higher-dimensional multi-objective problems arise, such as optimizing multiple properties of drug-like molecules.

Additionally, future improvements include extending the current work to higher-dimensional MPO tasks such as GUACAMOL's Osimertinib MPO (4-dimensional), with more Bayesian optimization iterations. We could also optimize the performance by tuning the GP hyperparameters by minimizing the negative log marginal likelihood (NLML) (see Section 2.6.5). Further, investigating the integration of constraints into EHVI formulations to handle practical, real-world constraints that arise in drug discovery tasks is also important. We also leave the investigation of the relationship between fingerprint features and the DockSTRING objectives in the Toy MPO setup for future work to investigate the difference in performance as discussed above.

7 | Conclusion

In this work, we present GP-MOBO, which is developed and rigorously tested for multi-objective molecular optimization. This work compares with both scalarized single-objective methods (Expected Improvement and Upper Confidence Bound acquisition functions) and a multi-objective approach using the EHVI acquisition function. Our primary contributions include a detailed comparison of GP-MOBO with the current state-of-the-art model GP-BO model by Tripp & Hernandez-Lobato(2024)[2], Gao et al (2022)[1], showing significant improvements in handling higher-dimensional fingerprint data and balancing conflicting objectives.

Key findings from our experiments show that GP-MOBO consistently outperforms GP-BO in finding diverse and optimal SMILES across multiple tasks, particularly when maximizing hypervolume. The ability of GP-MOBO to better explore chemical space, as seen in the improved diversity on the Pareto front, demonstrates the superiority in identifying molecules that meet multiple objectives more effectively. The results are particularly pronounced in tasks such as Fexofenadine MPO, where GP-MOBO demonstrated enhanced exploration of structural diversity, resulting in better objective performance when compared to the current GP-BO model (UCB acquisition function and `FP_DIM = 2048` were implemented) by Tripp & Hernandez-Lobato(2024)[2].

However, there are limitations to our approach. While GP-MOBO excels in synthetic tasks, such as the Toy MPO setup, its advantages diminish in more complex real-world tasks like the GUACAMOL MPO setup. This is primarily due to the different role fingerprint dimensionality plays in these scenarios. In synthetic tasks, using full fingerprint dimensionality correlates strongly with improved performance, as the higher-dimensional representations capture specific substructures that are essential for optimizing the objectives. However, in real-world tasks, where the chemical space is more diverse and complex, the standard dimensionality of 2048 appears to be sufficient for GP-BO models, as it captures enough variability to achieve similar results. Our findings suggest that while higher-dimensional fingerprints offer advantages in synthetic tasks, they do not confer the same benefit in more realistic, diverse drug discovery tasks.

In future work, further investigation into the scalability of GP-MOBO for larger datasets, beyond 10,000 SMILES used in this work, would also be beneficial. Additionally, investigating why GP-MOBO outperforms GP-BO in the toy MPO setup needs to be explored. Finally, examining more diverse chemical spaces and different multi-objective acquisition functions could yield additional insights into optimizing molecular properties more robustly.

References

[1] Wenhao Gao, Tianfan Fu, Jimeng Sun, and Connor Coley. Sample efficiency matters: A benchmark for practical molecular optimization, 10 2024.

[2] Austin Tripp and Jose Miguel Hernandez-Lobato. Diagnosing and fixing common problems in bayesian optimization for molecule design, 2024.

[3] Matt J Kusner, Brooks Paige, and Jose Miguel Hernandez-Lobato. Grammar variational autoencoder, 2017.

[4] Wengong Jin, Regina Barzilay, and Tommi Jaakkola. Junction tree variational autoencoder for molecular graph generation, 03 2019.

[5] Nicola De Cao and Thomas Kipf. Molgan: An implicit generative model for small molecular graphs. *arXiv:1805.11973 [cs, stat]*, 09 2022.

[6] Zaccary Quadrant, Artem Cherkasov, Jason Tyler, and Rolfe Quadrant. All smiles variational autoencoder, 06 2019.

[7] Andrea Karlova, Wim Dehaen, and Daniel Svozil. Molecular fingerprint vae, 2021.

[8] Ruslan Tazhigulov, Joshua Schiller, Jacob Oppenheim, and Max Winston. Molecular fingerprints for robust and efficient ml-driven molecular generation, 10 2022.

[9] John Bradshaw, Brooks Paige, Matt Kusner, Marwin Segler, and Jose Miguel HernÃ¡ndez-Lobato. Barking up the right tree: an approach to search over molecule synthesis dags, 12 2020.

[10] Natalie Maus, Haydn Jones, Juston Moore, Matt Kusner, John Bradshaw, and Jacob Gardner. Local latent space bayesian optimization over structured inputs, 2022.

[11] Antoine Grosnit, Rasul Tutunov, Alexandre Max Maraval, Ryan-Rhys Griffiths, Alexander I Cowen-Rivers, Lin Yang, Lin Zhu, Wenlong Lyu, Zhitang Chen, Jun Wang, Jan Peters, and Haitham Bou-Ammar. High-dimensional bayesian optimisation with variational autoencoders and deep metric learning, 2021.

[12] Judith Butepage, Lucas Maystre, and Mounia Lalmas. Gaussian process encoders: Vaes with reliable latent-space uncertainty. *Lecture notes in computer science*, 12976:84–99, 01 2021.

[13] Marcus Olivecrona, Thomas Blaschke, Ola Engkvist, and Hongming Chen. Molecular de-novo design through deep reinforcement learning. *Journal of Cheminformatics*, 9, 09 2017.

[14] Yoshua Bengio, Salem Lahlou, Tristan Deleu, Edward J Hu, Mo Tiwari, and Emmanuel Bengio. Gflownet foundations, 2021.

[15] Nathan Brown, Marco Fiscato, Marwin H.S. Segler, and Alain C. Vaucher. Guacamol: Benchmarking models for de novo molecular design. *Journal of Chemical Information and Modeling*, 59:1096–1108, 03 2019.

[16] Wenhao Gao, RocÃo Mercado, and Connor W. Coley. Amortized tree generation for bottom-up synthesis planning and synthesizable molecular design. *arXiv:2110.06389 [cs, q-bio]*, 03 2022.

[17] Jan H. Jensen. A graph-based genetic algorithm and generative model/monte carlo tree search for the exploration of chemical space. *Chemical Science*, 10:3567–3572, 2019.

[18] Jose Miguel HernÃ¡ndez-Lobato, James Requeima, Edward O Pyzer-Knapp, and AlÃ¡n Aspuru-Guzik. Parallel and distributed thompson sampling for large-scale accelerated exploration of chemical space. *PMLR*, 70:1470–1479, 07 2017.

[19] Austin Tripp, Gregor Simm, and Jose Miguel HernÃ¡ndez-Lobato. A fresh look at de novo molecular design benchmarks, 05 2023.

[20] J. Knowles. Parego: a hybrid algorithm with on-line landscape approximation for expensive multiobjective optimization problems. *IEEE Transactions on Evolutionary Computation*, 10:50–66, 02 2006.

[21] R. Timothy Marler and Jasbir S. Arora. The weighted sum method for multi-objective optimization: new insights. *Structural and Multidisciplinary Optimization*, 41:853–862, 12 2009.

[22] Michael Emmerich, Andre Deutz, Jan-Willem Klinkenberg, and Niels Bohrweg. The computation of the expected improvement in dominated hypervolume of pareto front approximations, 2008.

[23] Kaifeng Yang, Michael Emmerich, Andre Deutz, and Thomas Back. Efficient computation of expected hypervolume improvement using box decomposition algorithms. *Journal of Global Optimization*, 75:3–34, 07 2019.

[24] A. J. Keane. Statistical improvement criteria for use in multiobjective design optimization. *AIAA Journal*, 44:879–891, 04 2006.

[25] Daniel Hernandez-Lobato, Jose Hernandez-Lobato, Amar Shah, and Ryan Adams. Predictive entropy search for multi-objective bayesian optimization. *PMLR*, 48:1492–1501, 06 2016.

[26] Samuel Daulton, Maximilian Balandat, and Eytan Bakshy. Differentiable expected hypervolume improvement for parallel multi-objective bayesian optimization, 2020.

[27] Edwin V Bonilla, Kian Chai, and Christopher Williams. Multi-task gaussian process prediction, 2007.

[28] Mauricio A Alvarez, Lorenzo Rosasco, and Neil D Lawrence. Kernels for vector-valued functions: a review, 2024.

[29] Stefan Falkner, Aaron Klein, and Frank Hutter. Bohb: Robust and efficient hyperparameter optimization at scale, 07 2018.

[30] Peter I Frazier, Warren B Powell, and Savas Dayanik. A knowledge-gradient policy for sequential information collection. *SIAM Journal on Control and Optimization*, 47:2410–2439, 01 2008.

[31] Rafael Gomez-Bombarelli, Jennifer N. Wei, David Duvenaud, Jose Miguel Hernandez-Lobato, Benjamin Sanchez-Lengeling, Dennis Sheberia, Jorge Aguilera-Iparraguirre, Timothy D. Hirzel, Ryan P. Adams, and Ala Aspuru-Guzik. Automatic chemical design using a data-driven continuous representation of molecules. *ACS Central Science*, 4:268–276, 01 2018.

[32] Jose Miguel Hernandez-Lobato, Michael Gelbart, Ryan Adams, Matthew Hoffman, and Zoubin Ghahramani. A general framework for constrained bayesian optimization using information-based search. *Journal of Machine Learning Research*, 17:1–53, 2016.

[33] Sarvesh Mehta, Manan Goel, and U. Deva Priyakumar. Mo-memes: A method for accelerating virtual screening using multi-objective bayesian optimization. *Frontiers in Medicine*, 9, 09 2022.

[34] David E. Graff, Eugene I. Shakhnovich, and Connor W. Coley. Accelerating high-throughput virtual screening through molecular pool-based active learning. *Chemical Science*, 12:7866–7881, 2021.

[35] Liva Ralaivola, Sanjay J. Swamidass, Hiroto Saigo, and Pierre Baldi. Graph kernels for chemical informatics. *Neural Networks*, 18:1093–1110, 10 2005.

[36] E. Anderson, G. Veith, and D. Weininger. Smiles (simplified molecular identification and line entry system): A line notation and computerized interpreter for chemical structures | science inventory | us epa, 1987.

[37] David Weininger. Smiles, a chemical language and information system. 1. introduction to methodology and encoding rules. *Journal of Chemical Information and Modeling*, 28:31–36, 02 1988.

[38] Panukorn Taleongpong and Brooks Paige. Improving fragment-based deep molecular generative models, 07 2024.

[39] Kexin Huang, Tianfan Fu, Wenhao Gao, Yue Zhao, Yusuf Roohani, Jure Leskovec, Connor Coley, Cao Xiao, Jimeng Sun, and Marinka Zitnik. Therapeutics data commons machine learning datasets and tasks for drug discovery and development, 2021.

[40] David Rogers and Mathew Hahn. Extended-connectivity fingerprints. *Journal of Chemical Information and Modeling*, 50:742–754, 04 2010.

[41] Joseph L. Durant, Burton A. Leland, Douglas R. Henry, and James G. Nourse. Reoptimization of mdl keys for use in drug discovery. *Journal of Chemical Information and Computer Sciences*, 42:1273–1280, 11 2002.

[42] Georg Hinselmann, Lars Rosenbaum, Andreas Jahn, Nikolas Fechner, and Andreas Zell. jcompoundmapper: An open source java library and command-line tool for chemical fingerprints. *Journal of Cheminformatics*, 3, 01 2011.

[43] Pierre Mahe, Liva Ralaivola, Veronique Stoven, and Jean-Philippe Vert. The pharmacophore kernel for virtual screening with support vector machines. *Journal of Chemical Information and Modeling*, 46:2003–2014, 08 2006.

[44] Moises Hassan, Robert D. Brown, Shikha Varma-O'Brien, and David Rogers. Cheminformatics analysis and learning in a data pipelining environment. *Molecular Diversity*, 10:283–299, 08 2006.

[45] Greg Landrum. Rdkit: A software suite for cheminformatics, computational chemistry, and predictive modeling, 2013.

[46] Aditya Raymond Thawani, Ryan-Rhys Griffiths, Arian Jamasb, Anthony Bourached, Penelope Jones, William McCorkindale, Alexander Aldrick, and Alpha Lee. The photoswitch dataset: A molecular machine learning benchmark for the advancement of synthetic chemistry. *ChemRxiv*, 07 2020.

[47] Naruki Yoshikawa, Kei Terayama, Masato Sumita, Teruki Homma, Kenta Oono, and Koji Tsuda. Population-based de novo molecule generation, using grammatical evolution. *Chemistry Letters*, 47:1431–1434, 11 2018.

[48] Zhenpeng Zhou, Steven Kearnes, Li Li, Richard N. Zare, and Patrick Riley. Optimization of molecules via deep reinforcement learning. *Scientific Reports*, 9, 07 2019.

[49] Sereina Riniker and Gregory A Landrum. Similarity maps - a visualization strategy for molecular fingerprints and machine-learning methods. *Journal of Cheminformatics*, 5, 09 2013.

[50] C Rasmussen and C Williams. Gaussian processes for machine learning, 2006.

[51] Ryan-Rhys Griffiths, Leo Klarner, Henry Moss, Aditya Ravuri, Sang Truong, Samuel Stanton, Gary Tom, Bojana Rankovic, Yuanqi Du, Arian Jamasb, Aryan Deshwal, Julius Schwartz, Austin Tripp, Gregory Kell, Simon Frieder, Anthony Bourached, Alex Chan, Jacob Moss, Chengzhi Guo, Johannes Durholt, Saudamini Chaurasia, Ji Park, Felix Strieth-Kalthoff, Alpha Lee, Bingqing Cheng, Philippe Schwaller, and Jian Tang. Gauche: A library for gaussian processes in chemistry, 2023.

[52] Thomas Beckers. An introduction to gaussian process models, 02 2021.

[53] Andrew Wilson, David Knowles, and Zoubin Ghahramani. Gaussian process regression networks, 10 2011.

[54] Carl Rasmussen and Zoubin Ghahramani. Occam's razor, 01 2000.

[55] Kashima Hisashi, Tsuda Koji, and Inokuchi Akihiro. Kernels for graphs. *MIT Press*, page 18, 2004.

[56] Cornelia Metzig, Gilles Bisson, Cecile Amblard, and Mirta Gordon. Graph kernels -a synthesis note on positive definiteness graph kernels -a synthesis note on positive definiteness, 2012.

[57] Giannis Nikolentzos, Giannis Siglidis, and Michalis Vazirgiannis. Graph kernels: a survey. *Journal of Artificial Intelligence Research*, 72:943–1027, 11 2021.

[58] Ping Li. Min-max kernels, 2015.

[59] S. Joshua Swamidass, Jonathan M Chen, Jocelyne Bruand, Peter Phung, Liva Ralaivola, and Pierre Baldi. Kernels for small molecules and the prediction of mutagenicity, toxicity and anti-cancer activity. *ResearchGate*, 21:i359–i368, 01 2005.

[60] James Young. Literature review: Graph kernels in chemoinformatics, 2022.

[61] László Babai. Graph isomorphism in quasipolynomial time, 01 2016.

[62] Jean-Philippe Vert. Graph kernels and applications in chemoinformatics, 2007.

[63] Arthur Gretton. Introduction to rkhs, and some simple kernel algorithms, 2019.

[64] Eric Xing, Sujay Kumar Jauhar, and Zhiguang Huo. 22 : Hilbert space embeddings of distributions.

[65] Pierre Mahe, Nobuhisa Ueda, Tatsuya Akutsu, Jean-Luc Perret, and Jean-Philippe Vert. Graph kernels for molecular structure-activity relationship analysis with support vector machines. *Journal of Chemical Information and Modeling*, 45:939–951, 05 2005.

[66] P. Jaccard. Jaccard, p. (1912). the distribution of the flora of the alpine zone. new phytologist, 11, 37-50. - references - scientific research publishing, 2015.

[67] T.T Tanimoto. "an elementary mathematical theory of classification and prediction", 2019.

[68] Andreia P Guerreiro, Carlos M Fonseca, and LuÃs Paquete. The hypervolume indicator. *ACM Computing Surveys*, 54:1–42, 07 2021.

[69] Michael Emmerich, Kaifeng Yang, Andre Deutz, Hao Wang, and Carlos M Fonseca. A multicriteria generalization of bayesian global optimization. *Springer optimization and its applications*, pages 229–242, 01 2016.

[70] Lyndon While, P Hingston, Luciano Barone, and S Huband. A faster algorithm for calculating hypervolume. *IEEE Transactions on Evolutionary Computation*, 10:29–38, 02 2006.

[71] Carlos M Fonseca and Manuel Luis Paquete, Lopez-Ibanez. An improved dimension-sweep algorithm for the hypervolume indicator. *IEEE Explore*, 09 2006.

[72] M.T.M. Emmerich, K.C. Giannakoglou, and B. Naujoks. Single- and multiobjective evolutionary optimization assisted by gaussian random field metamodels. *IEEE Transactions on Evolutionary Computation*, 10:421–439, 08 2006.

[73] Kaifeng Yang, Michael Emmerich, Andre Deutz, and Thomas Back. Multi-objective bayesian global optimization using expected hypervolume improvement gradient. *Swarm and Evolutionary Computation*, 44:945–956, 02 2019.

[74] G Meisters. Lebesgue measure on the real line, 1997.

[75] M Fleischer. The measure of pareto optima applications to multi-objective metaheuristics. *Lecture notes in computer science*, pages 519–533, 01 2003.

[76] Luis Paquete, Carlos M. Fonseca, and Manuel Lopez-Ibanez. An optimal algorithm for a special case of klee's measure problem in three dimensions. *Core.ac.uk*, 2024.

[77] Justin Klekota and Frederick P Roth. Chemical substructures that enrich for biological activity. *Computer applications in the biosciences*, 24:2518–2525, 11 2008.

[78] Jakub Adamczyk and Piotr Ludynia. Scikit-fingerprints: easy and efficient computation of molecular fingerprints in python, 07 2024.

[79] Miguel Garcia-Ortegon, Gregor Simm, Austin J. Tripp, Jose Miguel Hernandez-Lobato, Andreas Bender, and Sergio Bacallado. Dockstring: easy molecular docking yields better benchmarks for ligand design, 2021.

[80] Joaquin Quinonero-Candela, Carl Rasmussen, Fabian Sinz, Olivier Bousquet, and Bernhard Scholkopf. Evaluating predictive uncertainty challenge, 2006.

[81] G Peter Lepage. A new algorithm for adaptive multidimensional integration. 05 2024.

[82] Harry Cunningham, Daniel Augusto De Souza, So Takao, Mark Van Der Wilk, and Marc Deisenroth. Actually sparse variational gaussian processes, 04 2023.

[83] Michalis Titsias. Variational learning of inducing variables in sparse gaussian processes, 04 2009.

[84] James Hensman, Nicolas Durrande, and Arno Solin. Variational fourier features for gaussian processes. *Journal of Machine Learning Research*, 18:1–52, 2018.

[85] Syrine Belakaria, Aryan Deshwal, and Janardhan Rao Doppa. Max-value entropy search for multi-objective bayesian optimization. *Advances in Neural Information Processing Systems*, 32, 2019.

[86] Sebastian Ament, Samuel Daulton, David Eriksson, Maximilian Balandat, and Eytan Bakshy. Unexpected improvements to expected improvement for bayesian optimization, 01 2024.

8 | Appendix

8.1 Source Code

Source code for all of the methods implemented in Chapter 3 and 4 for the project can be found in the GitHub repository: https://github.com/anabelyong/GP-MOBO.

8.2 Preliminary Mathematical Background

8.2.1 Cholesky Decomposition

The Cholesky decomposition, is useful in numerical methods including Gaussian Process (GP) regression. It is a specialization of the general LDU (lower-diagonal-upper) decomposition and is particularly applicable to symmetric, positive semi-definite matrices.

Decomposing Symmetric Matrices

Given a symmetric matrix A such that $A = A^T$, the Cholesky decomposition allows us to factor A as:

$$A = LL^T$$

where L is a lower triangular matrix. This factorization is useful because it reduces the complexity of operations on A from $\mathcal{O}(n^3)$ to $\mathcal{O}(n^2)$ by leveraging the structure of triangular matrices.

Connection to LDU Decomposition

The LDU decomposition of a matrix A:

$$A = LDU$$

where L is a lower triangular matrix, D is a diagonal matrix, and U is an upper triangular matrix. For symmetric matrices, it holds that $L = U^T$. Therefore, the LDU decomposition for a symmetric matrix can be rewritten as:

$$A = LDL^T$$

This is where the Cholesky decomposition steps in. We can simplify D by further noting that D can be expressed as the square of a diagonal matrix $\sqrt{D}$, i.e:

$$D = \sqrt{D} \cdot \sqrt{D}^T$$

Substituting this into the LDU decomposition gives:

$$A = L\sqrt{D} \cdot \sqrt{D}^T L^T = (L\sqrt{D})(L\sqrt{D})^T$$

Letting $L' = L\sqrt{D}$, we obtain the Cholesky decomposition:

$$A = L'L'^T$$

L' is the Cholesky factor of matrix A.

Positive Semi-Definiteness and Cholesky Decomposition

The Cholesky decomposition requires that the matrix A is positive semi-definite. This requirement ensures that all eigenvalues of A are non-negative, which in turn guarantees that the decomposition exists and is numerically stable.

The reason for this requirement can be intuitively understood by considering the quadratic form $\mathbf{x}^\top A\mathbf{x}$ for any non-zero vector $\mathbf{x}$:

$$x^T Ax = x^T L'L'^T x = ||L'^T x||^2 \geq 0$$

This expression confirms that $\mathbf{x}^\top A\mathbf{x} \geq 0$ if A is positive semi-definite, meaning the matrix A can be factored as $\mathbf{L}\mathbf{L}^\top$.

Applications in Gaussian Processes

In the context of Gaussian Processes, the Cholesky decomposition is particularly useful when sampling from a multivariate normal distribution. Given a multivariate Gaussian distribution $\mathbf{X} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$, where $\boldsymbol{\Sigma}$ is the covariance matrix, we can express the distribution as:

$$\mathbf{X} = \boldsymbol{\mu} + \mathbf{L}\mathbf{Z}$$

where $\mathbf{L}$ is the Cholesky factor of $\boldsymbol{\Sigma}$, and $\mathbf{Z}$ is a vector of independent standard normal

variables. This approach simplifies the sampling process and is essential for efficient GP regression implementations. The Cholesky decomposition not only simplifies matrix operations but also ensures numerical stability, making it a foundational tool in advanced statistical methods, including Gaussian Processes.

8.2.2 Mercer's Theorem

Definition 12 ***Mercer's Theorem:*** *Let $\mathcal{C}$ be a compact subset of $\mathbb{R}^n$. To ensure that a continuous symmetric kernel function $K(x_1, x_2)$ defined on $\mathcal{C}$ can be represented as an inner product in some feature space, the following expansion must hold:*

$$K(x_1, x_2) = \sum_{k=1}^{\infty} \alpha_k \Phi_k(x_1) \Phi_k(x_2)$$

where $\alpha_k > 0$ are positive coefficients, and $\{\Phi_k(x)\}$ are the basis functions representing the implicit mapping from the input space $\mathcal{C}$ to the feature space. For the expansion to the valid, it is both necessary and sufficient that the kernel K is positive semi-definite, meaning that it satisfies the condition:

$$\int_{\mathcal{C}} \int_{\mathcal{C}} g(x_1) g(x_2) K(x_1, x_2) \delta x_1 \delta x_2 \geq 0$$

for all square-integrable functions $g \in \mathcal{L}_2(\mathcal{C})$

8.2.3 Positive Definite Kernel

Definition 13 ***Positive Definite Kernel:*** *A function $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is called positive definite if, for any finite set of molecular fingerprints $x_1, ..., x_n$ and any set of real numbers $\alpha_1, ..., \alpha_n$, the following holds:*

$$\sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j k(x_i, x_j) \geq 0$$

This condition ensures that the associated Gram matrix (a matrix of kernel evaluations between pairs of data points) is positive semi-definite.

To establish that a function is positive semidefinite is central to applying the Mercer theorem. The integral in many cases, shown in Appendix, cannot be evaluated explicitly, making the proof of positive definiteness nontrivial. However, using the closer properties

of positive definite functions defined below, we can show that this is a positive semidefinite symmetric kernel, ensuring that the kernel methods we are investigating can operate effectively within the RKHS framework.

Definition 14 ***Closure properties***

- *Closure under a sum: For two positive semidefinite symmetric kernels* $K_1, K_2 : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$*, the sum becomes:*

$$K = K_1 + K_2 : \mathcal{X} + \mathcal{X} \to \mathbb{R}$$

  is a positive semidefinite symmetric kernel.

- *Closure under a product: For two positive semidefinite symmetric kernels* $K_1, K_2 : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$*, the sum becomes:*

$$K = K_1 \times K_2 : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$$

  is a positive semidefinite symmetric kernel.

Additionally, Aronszajn's theorem notes that any positive definite kernel corresponds to an inner product in some Hilbert space $\mathcal{H}$, with a mapping $\Phi : \mathcal{X} \to \mathcal{H}$ such that:

$$k(x, x') = \langle \Phi(x), \Phi(x') \rangle_{\mathcal{H}}$$

This establishes the theoretical foundation for the kernel trick, as it guarantees that kernel methods can operate as if they were working in this high-dimensional space, even when space is not explicitly constructed.

8.2.4 Lebesgue Measure

The Lebesgue measure is a fundamental concept in measure theory, extending the intuitive notion of length, area, and volume to more complex sets beyond simple intervals. It was developed by Henri Lebesgue as a way to rigorously define the "size" of a set in a way that generalizes the concept of length to more abstract sets. Key properties of Lebesgue Measure are:

1. **Extends Length:** For any interval $I = [a, b]$ in the real line $\mathbb{R}$, the Lebesgue measure $\mu(I)$ coincides with the length of the interval, i.e. $\mu(I) = b - a$.
2. **Monotonicity:** If $A \subseteq B \subseteq \mathbb{R}$, then the Lebesgue measure is non-decreasing

$\mu(A) \leq \mu(B)$. This ensures that larger sets have a greater or equal measure compared to their subsets.

3. **Translation Invariance:** For any set $A \subseteq \mathbb{R}$ and any real number x_0, the measure of A remains the same if the set is translated by x_0. Formally, $\mu(A + x_0) = \mu(A)$, where $A + x_0 = \{x + x_0 : x \in A\}$.

4. **Countable Additivity:** is a countable collection of disjoint sets, then the measure of the union of these sets is the sum of their measures. Essentially, if $A_i \cap A_j = \emptyset$ for $i \neq j$, then:

$$\mu\left(\bigcup_{i=1}^{\infty} A_i\right) = \sum_{i=1}^{\infty} \mu(A_i)$$

Definition 15 ***Lebesgue Outer Measure:*** *For any subset $E \subseteq \mathbb{R}$, the concept of Lebesgue outer measure $\mu^*(E)$. This is defined as:*

$$\mu^*(E) = inf\left\{\sum_{k=1}^{\infty} l(I_k) : E \subseteq \bigcup_{k=1}^{\infty} I_k, \text{ where each } I_k \text{ is an interval}\right\}$$

Here, $\ell(I_k)$ denotes the length of the interval I_k, and the outer measure $\mu^*(E)$ is the infimum of sum of lengths of intervals covering the set E.

Definition 16 ***Lebesgue Measurable Sets:*** *A set $E \subseteq \mathbb{R}$ is Lebesgue measurable if, for every set $A \subseteq \mathbb{R}$, the following holds:*

$$\mu^*(A) = \mu^*(A \cap E) + \mu^*(A \cap E^C)$$

where E^c denotes the complement of E. The Lebesgue measure $\mu(E)$ of a measurable set E is then defined as the outer measure $\mu^*(E)$.

HV Indicator Relevance: These definitions and properties here are relevant for the Hypervolume Calculation, as it allows us to calculate this "volume" precisely, whether it is in 1 dimension (length), 2 dimensions(area) or higher dimensions (volume). The monotonicity property ensures that as the set of Pareto-optimal solutions expands, the hypervolume (measure of dominated region) increases or remains the same but never decreases. This property ensures that the hypervolume indicator correctly reflects improvements in the Pareto front.

Countable additivity ensures that the total hypervolume is simply the sum of the measures of these individual regions. This property is fundamental when calculating the

hypervolume, as it guarantees that the measure of the entire dominated region can be computed by adding up the measures of smaller, disjoint parts Conclusively, the hypervolume indicator can be thought of as a measure of the "outer" region dominated by the Pareto front, bounded by the reference point. The Lebesgue outer measure helps in defining this measure rigorously, ensuring that the hypervolume is calculated as the smallest possible "volume" that covers the entire dominated region.

8.2.5 Klee's Measure Problem

Klee's Measure Problem, a huge problem in computational geometry, involves determining the measure (such as length, area or volume) of the union of a collection of axis-aligned rectangles (or more generally, hyperrectangles) in d-dimensional space. The problem is stated as follows:

Definition 17 ***Klee's Measure Problem:*** *Let $R = \{R_1, R_2, ..., R_n\}$ be a set of n axis-aligned hyperrectangles in $\mathbb{R}^d$. Each hyperrectangle R_i is defined by its lower and upper bounds in each dimension. The objective of Klee's Measure Problem is to compute the volume of the union of these hyperrectangles, denoted by $V(R)$, where the volume is defined as the measure of the region covered by at least one hyperrectangle in R. This measure is expressed as:*

$$V(R) = \mu_d \left(\bigcup_{i=1}^{n} R_i \right)$$

where μ_d denotes the Lebesgue measure in d-dimensions.

The challenge in solving KMP arises from the potential overlap among hyperrectangles, as simply summing the volumes of the individual hyperrectangles would overestimate the total volume due to overlapping regions. The problem is known to have a complexity of $\mathcal{O}(n \log n + n^{d/2} \log n)$ in general d-dimensional space, making it computationally challenging for higher dimensions.

8.2.6 Gaussian Random Fields (GRFs)

A Gaussian random field (GRF is a collection of random variables indexed by a set of points in space, typically denoted as $Y(x)$, where x belongs to some spatial domain $\mathbb{R}^d$. Any finite collection of these random variables follows a multivariate Gaussian distribution. This property makes GRFs a powerful tool in modeling spatial phenomena where the underlying stochastic process is assumed to be Gaussian.

Definition 18 ***GRFs:*** *Let $\{Y(x) : x \in \mathbb{R}^d\}$ be a random field and for any finite set of points $\{x_1, x_2, ..., x_n\} \subset \mathbb{R}^d$, the joint distribution of $(Y(x_1), Y(x_2), ..., Y(x_n))$ is a multivariate Gaussian. This can be expressed as:*

$$Y = (Y(x_1), Y(x_2), ..., Y(x_n)) \sim \mathcal{N}(\mu, C)$$

where $\mu = (\mu(x_1), \mu(x_2), ..., \mu(x_n))$ is the mean vector and C is the covariance matrix with entries $C_{ij} = Cov(Y(x_i), Y(x_j))$

Covariance Structure and Homogeneity

An important component of GRFs, is their covariance structure, which dictates how the values of the field are correlated across space. The covariance function $C(x_i, x_j) = Cov(Y(x_i), Y(x_j))$ captures the spatial dependence between two points x_i and x_j. In the case of homogeneous fields, the covariance function depends only on the relative distance between the points $C(x_i, x_j) = C(r)$ where $r = ||x_i - x_j||$.

The covariance function is central to the GRF's smoothness properties and its related to the power spectrum $P(k)$ through the Fourier transform:

$$C(r) = \int_{\mathbb{R}^d} P(k) e^{ik \cdot r} \delta k$$

where P(k) is the power spectral density function, which describes the distribution of variance as a function of spatial frequency k.

8.3 Molecular Objectives Definitions

- $f(m) = -\text{DockingScore}(\text{PPARD}, m)$, represents the negative docking score for the Peroxisome Proliferator-Activated Receptor Delta (PPARD) score. A higher value of f_1 indicates a stronger binding affinity between molecule m and PPARD target.
- $f(m) = -\text{QED}(m)$ is Quantitative Estimate for Drug-likeness (QED) score. This metric evaluates how "drug-like" a molecule is, based on factors such as molecular weight, lipophilicity (logP) and number of hydrogen bond donors/acceptors. Higher QED values suggest molecule possesses properties commonly associated with effective drugs.
- $f(m) = \text{sim}(m, \text{celecoxib})$ measures similarity of molecule m to Celecoxib, a well-known drug, using a fingerprint-based similarity metric. This is to ensure the

molecules retain high degree fo structural similarity to an existing successful drug, maintaining potential efficacy.

- $f(m)$ = Fexofenadine MPO measures multiple objectives for Fexofenadine by scalarizing the objectives below with geometric mean:
  - $f(m) = \text{sim}(m, \text{fexofenadine, AP})$: Measures similarity to Fexofenadine based on aromatic properties (AP).
  - $f(m) = \text{TPSA}(m)$: Topological Polar Surface Area, representing molecule polarity and influencing permeability and bioavailability.
  - $f(m) = \log P(m)$: Logarithmic partition coefficient, reflecting molecule hydrophobicity or lipophilicity.
- $f(m)$ = Amlodipine MPO measures multiple objectives for Amlodipine by scalarizing the objectives below with geometric mean:
  - $f(m) = \text{sim}(m, \text{amlodipine, ECFP4})$: Measures similarity to Amlodipine using ECFP4 fingerprints.
  - $f(m) = \text{NumberRings}(m)$: Measures the number of rings in the molecule, applying Gaussian smoothing to match the optimal number for drug-like properties.
- $f(m)$ = Perindopril MPO measures multiple objectives for Perindopril by scalarizing the objectives below with geometric mean:
  - $f(m) = \text{sim}(m, \text{perindopril, ECFP4})$: Measures similarity to Perindopril using ECFP4 fingerprints.
  - $f(m) = \text{NumberAromaticRings}(m)$: Measures the number of aromatic rings in the molecule, applying Gaussian smoothing to reflect drug-like structures.

8.4 GP-MOBO Implementation Details

8.4.1 Oracle Utility Function Example

```python
# Create "oracles" for various objectives
QED_ORACLE = Oracle("qed")
CELECOXIB_ORACLE = Oracle("celecoxib-rediscovery")
LOGP_ORACLE = Oracle("logp")


def evaluate_objectives(smiles_list: list[str]) -> np.ndarray:
    # Initialize arrays for each objective
    f1 = np.array([-DOCKSTRING_DATASET["PPARD"].get(s, np.nan) for s in smiles_list])
    f2 = np.array(QED_ORACLE(smiles_list))
    f3 = np.array(CELECOXIB_ORACLE(smiles_list))

    # Filter out NaN values from f1 and corresponding entries in other arrays
    valid_indices = ~np.isnan(f1)
    f1 = f1[valid_indices]
    f2 = f2[valid_indices]
    f3 = f3[valid_indices]


    # Ensure all arrays have the same shape
    if not (len(f1) == len(f2) == len(f3)):
        raise ValueError("All input arrays must have the same shape")

    out = np.stack([f1, f2, f3])  # 3xN
    return out.T  # transpose, Nx3
```

Figure 8.1: Oracle Function for Toy MPO Experimental Setup

8.4.2 Hypervolume Computation Test Cases

The test cases for Hypervolume Indicator and Expected Hypervolume Improvement were available in BoTorch's https://github.com/pytorch/botorch/blob/main/test/utils/multi_objective/test_hypervolume.py. The results from our implementation treating the data as a numpy array, instead of using tensorial data such as BoTorch is as shown:

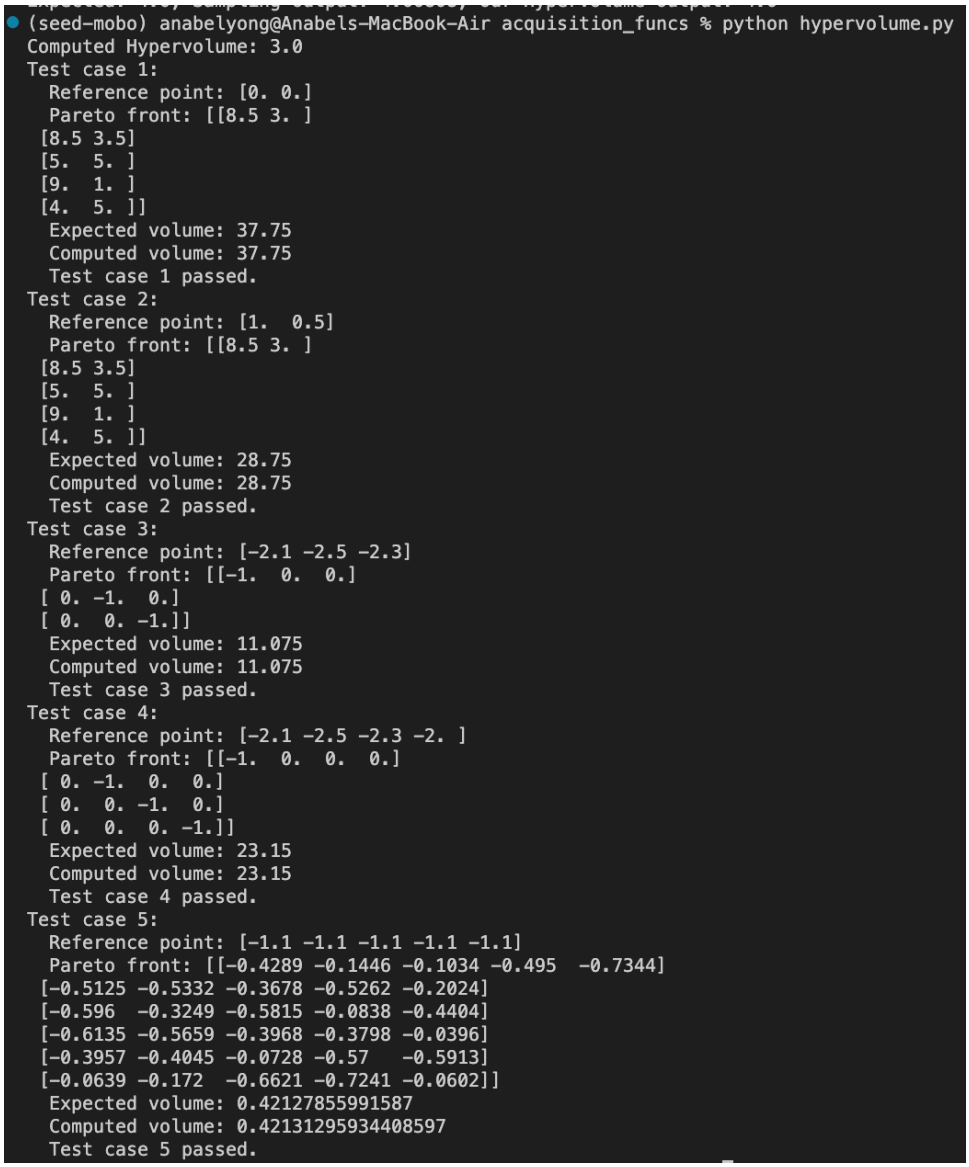

Figure 8.2: **Hypervolume Test Cases available from BoTorch passed by our EHVI implementation**

8.4.3 Negative Log Predictive Density (NLPD)

The predictive density of a test observation $\tilde{y}$ given training data (x,y) and test data $\tilde{x}$ can be expressed as:

$$p(\tilde{y}|\tilde{x}, x, y) = \int p(\tilde{y}, \tilde{x}, \theta) \cdot p(\theta|x, y)\delta\theta$$

where θ represents the GP's parameters. This integral is typically computed through Monte Carlo methods:

$$p(\tilde{y}|\tilde{x}, x, y) \approx \frac{1}{M} \sum_{m=1}^{M} p(\tilde{y}|\tilde{x}, \theta^m)$$

where θ^m being draws from the posterior distribution given the training data. Taking the logarithm of the predictive density provides the log predictive density, which is averaged over all test cases:

$$\log p(\tilde{y}|\tilde{x}, x, y) \approx -\log M + \log \sum_{m=1}^{M} \exp(\log p(\tilde{y}, \tilde{x}, \theta^{(m)}))$$

The NLPD is the negative of this average, giving us a measure of how well the model's predictive distribution captures the true outcomes.

8.5 Additional Results

8.5.1 Example of Training Dataset for Both GP-MOBO and GP-BO

Known SMILES	f_1	f_2	f_3	**GMean**
O=C(NC1=C2C(NC=C2)=NC=C1) C3CCC(CC3)C(N)C	8.3	0.8107	0.1102	0.9051
S(=O)(=O)(/N=C/1/C=C(C2C(=O)CC(CC2=O)(C)C) C(=O)C=3C1=CC=CC3)C4=CC=C(C=C4)C	9.5	0.6628	0.2823	1.2114
O=C1N(C2CCCCCC2)CC(=O)N(C1C3=CC(OC) =C(OC)C=C3)CC4=CC=CC=C4	9.0	0.6324	0.1643	0.9779
O(C=1C=C(CNC=2N(C=3C(N2)=CC=CC3) CCN4CCCCC4)C=CC1)C	9.0	0.6840	0.1615	0.9981
O1N=C(C=C1CC(C)C)C(=O)NCCC2=CC=CC=C2	8.8	0.8794	0.1327	1.0089
ClC1=CC=C(N2C(=NN=C2SC(C(=O) NC3=CC=4OCOC4C=C3)C)C(N(C)C)C)C=C1	8.7	0.5094	0.2015	0.9630
O[C@]1([C@@]2([C@H]([C@H]3[C@H]([C@@H](O) C2)[C@@]4(C(=CC3)CC(=O)C=C4)C)CC1)C) C(=O)COC(=O)C	7.8	0.5540	0.0563	0.6243
O=N(=O)C1=CC(/C(=N/NC=2N=C(C(=NN2) C=3C=CC=CC3)C4=CC=CC=C4)/C)=CC=C1	10.2	0.2738	0.1926	0.8133
ClC1=CC=C(S(=O)(=O)C=2C(=CC(=NC2NC) C)C=C1	8.2	0.9453	0.2755	1.2878
ClC1=CC(NC=2N=C(N=C(N2)N) CN3CCN(CC3)CC4=CC=5OCOC5C=C4)=CC=C1C	10.2	0.5653	0.1517	0.9564

Table 8.1: Initial Training Set: Known SMILES and Corresponding Objective Values (f_1, f_2, f_3) for multi-objective GP-MOBO setup and their Geometric Mean of f_1, f_2, f_3.

Known SMILES	**GMean**
C1=C(C2=C(C=C1O) OC(C(C2=O)=O)C3=CC=C(C(=C3)O)O)O	0.9159
O=S(=O)(N1CCNCCC1) C2=CC=CC=3C2=CC=NC3	1.2550
C=1C=C2S/C(/N(CC) C2=CC1OC)=CC(=O)C	1.0040
C=1(N=C(C=2C=NC=CC2) C=CN1)NC=3C=C(NC(C4=CC=C(CN5CCN(CC5) C)C=C4)=O)C=CC3C	0.9521
C1=CC=2C(=CNC2C=C1) C=3C=CN=CC3	1.0327
N1(C2=C(C(N)=NC=N2) C=N1)C3=CC=CC=C3	0.8999
C1(=C2C(C=CC=C2)=NC=N1) NC3=CC(OC)=CC=C3	0.9412
N1C(N(C(C2=CC=CC=C12)=O) CCN3CCC(CC3)=C(C=4C=CC(=CC4)F) C=5C=CC(=CC5)F)=S	1.1289
C1(O[C@@H](CC(C(=CC([C@H]([C@H](C([C@@H] (C[C@@H](C=CC=CC=C([C@H](C[C@H]2O[C@] (C(C(N3[C@H]1CCCC3)=O)=O)(O)[C@@H](CC2)C)OC) C)C)=O)OC)O)C)C)=O)[C@@H](C[C@H]4C[C@@H](OC) [C@H](O)CC4)C)=O	0.9731
O=C1C=2C=3C(=NNC3C=CC2) C4=C1C=CC=C4	0.9411

Table 8.2: Initial Training Set: Known SMILES and Corresponding Geometric Mean Values provided for the single-objective GP BO setup

8.5.2 Dataset BEST SMILES (Top 20 SMILES) in Toy MPO Setup

Dataset Best SMILES for Toy MPO DockSTRING setup	**Value of Best SMILES**
S(C1=CC=C(C=C1)N2C(C3=CC=C(C)C=C3)=CC(C(F)(F)F)=N2)(N)(=O)=O	1.9478373773604283
FC(F)(F)C1=CC(N2N=CC(=C2N)C=3C=CC(=CC3)C)=CC=C1	1.567558036817745
ClC1=CC=C(C=2C(=O)N(NC=3C=CC(=CC3)C)C(=O)C2)C=C1	1.466675340058482
N1(N=C(C=2C=CC(=CC2)C)C=C1N)C3=CC=C(C=C3)C	1.4641374959896714
S(=O)(=O)(NNC1=NC=2C(N=C1C(F)(F)F)=CC=CC2)C3=CC=C(C=C3)C	1.4512344549267704
S(=O)(=O)(N)C1=CC=C(NC(=O)NC=2C=CC(=CC2)C(F)(F)F)C=C1	1.439286157552866
FC(F)(F)C1=CC(N2CCN(CC2)C(=O)CC3=CC=C(C=C3)C)=CC=C1	1.4249646907867288
S(=O)(=O)(NNC=1C=CC(=CC1)C)C2=CC=C(C=C2)C	1.421004073331502
C1(=CC=C(S(NC2=CC=C(C=C2)C(NC3=NOC(=C3)C)=O)(=O)=O)C=C1)C	1.4127307356041288
CCNC(=O)C=1C=CC(=CC1)N2C(=CC(=N2)C)C3=CC=CC=C3	1.4116668811297421
S(=O)(=O)(N1N=C(N)C(=C1)C2=CC=C(F)C=C2)C3=CC=C(OCC)C=C3	1.4061813943772048
S(=O)(=O)(N)C1=CC=C(C=2C(=NOC2C)C=3C=CC=CC3)C=C1	1.4043521001657435
O=C(N1N=C(N=C1N)C=2C=CC(=CC2)C)CC3=CC=CC=C3	1.3989038673335519
S(=O)(=O)(N1CCN(CC1)C=2C(=CC=CC2)C(F)(F)F)C3=CC=C(C=C3)C	1.3957869324271062
ClC1=CC(CN2CC(=NS(=O)(=O)C3=CC=C(C=C3)C)C=CC2=O)=CC=C1Cl	1.3941200508505909
S(=O)(=O)(C=1C(=CC(=NC1NC)C)C)C2=CC=C(C=C2)C	1.3939248840045841
S(=O)(=O)(NCC)C=1C=CC(NC(=O)C=2N(N=C(C2)C(F)(F)F)C)=CC1	1.3908884961286663
S(=O)(=O)(NCCN1C=2C(C=C1C)=CC=CC2)C3=CC=C(C=C3)C	1.3878057287883785
FC(F)(F)C1=CN(CC=2C=CC(=CC2)C(=O)NC3=CC(=CC=C3)C)C(=O)C=C1	1.3767026956086061
S(=O)(=O)(NC1=C2CCCC2=NC(O)=C1)C3=CC=C(C=C3)C	1.3749559507452085

Table 8.3: Dataset Best SMILES and their corresponding Values of Best SMILES in Toy MPO Setup

8.5.3 Dataset BEST SMILES (Top 20 SMILES) in Fexofenadine MPO

Best 20 SMILES for Fexofenadine MPO from GUACAMOL	Best SMILES Value
O=C(CCC(=O)NC(CO)C(O)c1ccc([N+](=O)[O-])cc1)NCCCNCCCCN(Cc1ccccc1)Cc1ccccc1	0.72892475
O=C(O)CC(O)(CSCCCCCCCc1ccc2ccccc2c1)C(=O)O	0.72478167
CNC(=O)N1CCC(NC(=O)c2ccc(Oc3ccc(C#CC4(O)CN5CCC4CC5)cc3)cc2)CC1	0.72041918
CN1C(=O)C(C(O)C2CCCCC2)NC(=O)C12CCN(Cc1ccc(Oc3ccc(C(=O)O)cc3)cc1)CC2	0.71171583
O=C(Cc1cc2ccccc2[nH]1)N1CCC(Nc2ncc(C(O)=NO)cn2)(c2ccccc2)CC1	0.70899797
COc1ccc2nccc(C(O)CN3CCC(NCc4cc5cccnc5[nH]4)CC3)c2n1	0.70524978
CC(C)(C)NC(=O)C1CN(Cc2cccnc2)CCN1C[S+](O-)CC(Cc1ccccc1)C(=O)NC1c2ccccc2CC1O	0.70424961
O=C(CN(c1cccc([N+](=O)[O-])c1)S(=O)(=O)c1ccccc1)N1CCCCC1	0.70213893
O=C(c1ccc(O)cc1OCC(O)CN1CCC2(CC1)Cc1cc(Cl)ccc1O2)N1CCOCC1	0.70144999
O=C(O)c1ccc(-c2noc(C3CCN(C(=O)NC4CC4c4ccccc4)CC3)n2)cc1	0.70001582
CCC(=O)N(c1ccccc1)C1(C(=O)OC)CCN(CCn2c(=O)c3ccccc3n(CC)c2=O)CC1	0.69978857
CC(O)C1C(=O)N2C(C(=O)[O-])=C(c3ccc(C[n+]4ccc(N5CCCCC5)cc4)cc3)CC12	0.69792347
O=C(c1ccc(NS(=O)(=O)c2cccc3c2OCCO3)cc1)N1CCC(O)(Cc2ccccc2)CC1	0.69737852
O=C(O)CC1c2ccccc2C(=O)N(CC(=O)NCCCCNc2nc3ccccc3[nH]2)c2ccccc21	0.69677083
O=C(O)CNC(C(=O)N1CCCC1C(=O)NCC#Cc1c[nH]cn1)C(c1ccccc1)c1ccccc1	0.69641193
NCc1ccc(Cl)cc1CNC(=O)C1CCCN1C(=O)C1(O)c2ccccc2-c2c1ccc[n+]2[O-]	0.69526927
Cn1c(=O)c(C(=O)NCC2CCN(Cc3ccccc3)CC2)c(O)c2cc(O)c(O)cc21	0.69161915
O=C(C=Cc1ccc([N+](=O)[O-])cc1)Nc1ccc(N2CCN(CC(O)(Cn3cncn3)c3ccc(F)cc3F)CC2)c(F)c1	0.68930971
CCOCCN(CC(O)CN1CCCC2(CC(=O)c3cc(O)ccc3O2)C1)S(=O)(=O)c1c(C)cccc1C	0.68897279
O=C(O)CN1CCC(c2c(C=Cc3ccc4ccccc4n3)nc3c(N4CCOCC4)ccnn23)CC1	0.68839601

Table 8.4: Best 20 SMILES and their corresponding Values for Fexofenadine MPO

8.5.4 Dataset BEST SMILES (Top 20 SMILES) in Amlodipine MPO

Best 20 SMILES for Amlodipine MPO from GUACAMOL	Best SMILES Value
COC(=O)C1=C(C)NC(C)=C(C(=O)OCc2cccc(F)c2)C1c1cccc([N+](=O)[O-])c1	0.61237244
CCOC(=O)C1=C(C)N=C(C)C(=C(O)OCC)C1c1nc2ccccc2[nH]1	0.58747999
CCOC(=O)C1=C(C)NC(C)=C(C(=O)OCC)C1c1cc(C(C)CC)c2oc(=O)c(C(=O)OC)cc2c1	0.58387421
COc1cc(C2NC(=O)NC(C)=C2C(C)=O)ccc1OCc1ccccc1Cl	0.57983351
CCOC(=O)C1=C(C)NC(=S)NC1c1ccc(NC(=O)Nc2ccc(OC)cc2)cc1	0.5585696
CCOC(=O)c1c(C)[nH]c(C)c1C(=O)CSc1nnnn1-c1ccccc1	0.5547002
COC(=O)c1c(SCC(=O)Nc2cccc(Cl)c2C)[nH]c2ccccc2c1=O	0.5500191
CCOC(=O)C1=C(C)OC(N)=C(C(=O)OCC)C12C(=O)Nc1ccc(Br)cc12	0.54461929
COC(=O)C1=C(C)N=C(C)C(=C(O)OC)C1C1=CCN(C(=O)Oc2ccccc2)C=C1	0.54189556
COC(OC)C1=C(C(=O)OCC=Cc2ccccc2)C(c2ccc(Cl)c(Cl)c2)C(C(=O)O)=CN1	0.54151
CCC1c2ccccc2CN1CNC(=O)c1cc(Cl)c(N)cc1OC	0.53802759
CCOC(=O)c1cnc2c(ccc3ccccc32)c1Cl	0.53708616
CCOC(=O)c1ccc(OCc2nc3ccccc3n(C)c2=O)cc1	0.53568323
CCOC(=O)C1=C(Nc2ccccc2)CC(C)(O)C(C(=O)OCC)C1c1ccc(Br)cc1	0.53555738
CCOC(=O)C1C(C(=O)OCC)C12C(=O)N(C)c1ccccc12	0.53329511
COC(=O)c1c(NC(=O)CCCOc2ccccc2)sc2c1CCC(C)C2	0.53229065
CCOc1ccccc1C1CC(=O)Nc2cc(OC)c(OC)cc21	0.53215208
CCOC(=O)C1=C(C)N(c2cccc(C(F)(F)F)c2)C(=O)N(C)C1c1ccc(C#N)cc1C(=O)N(C)CCCOC	0.52812079
CCOC(=O)c1c[nH]c2c(ccc3nc(Cl)cc(C)c32)c1=O	0.52704628
CCOC(=O)c1cnc2ccc(OC)cc2c1NCCc1ccccc1	0.52660319

Table 8.5: Best 20 SMILES and their corresponding Values for Amlodipine MPO

8.5.5 Dataset BEST SMILES (Top 20 SMILES) in Perindopril MPO

Best 20 SMILES for Perindopril MPO from GUACAMOL	**Best SMILES Value**
CCCC(NC(=O)C(N)Cc1ccc(O)cc1)C(=O)N1CCCC1C(=O)NCC(=O)NC(Cc1ccccc1)C(=O)N1CCCC1C(=O)O	0.48795004
CCOC(=O)C(Cc1ccc(O)cc1)NC(=O)C1(NC(=O)C(SC(=O)c2ccccn2)C(C)C)CCCC1	0.48760869
CCOC(=O)C(CNC(C)=O)c1cn(C(=O)OCC)c2ccccc12	0.47756693
CC(C)CC(C(=O)NC1CCCCC1)N(Cc1cccs1)C(=O)c1snc(C(N)=O)c1N	0.47050403
CCOC(=O)C(C)(C)Oc1ccc(N(CC2CCCC2)C(=O)Nc2nccs2)cc1	0.46770717
COC(C)(C)C(O)C(=O)N1C(C(=O)NCc2cc(Cl)ccc2-n2cnnn2)CC2CC21	0.46589083
CCOCCOC(=O)Nc1cc2nc(C3CCCCC3)[nH]c2cc1N(C)C	0.46164354
CCOC(=O)C(CCCCNC(=O)C(C)n1c([N+](=O)[O-])cnc1C)NC(=O)Cn1cc([N+](=O)[O-])nc1C	0.4612656
CCCc1cc(=O)oc2c(C)c(OCC(=O)N3CCC(C(=O)O)CC3)ccc12	0.45976311
CC(C)C(O)CC(O)C(CC1CCCCC1)NC(=O)C(Cc1c[nH]cn1)NC(=O)c1ccc[nH]1	0.45841567
CC(=O)NC(Cc1ccccc1)C(=O)N1CCCC1C(=O)NC(CCCn1ccnc1)B1OC2CC3CC(C3(C)C)C2(C)O1	0.45812285
CCOC(=O)c1c(OC2CCCCC2C)cc(CCc2ccccc2)[nH]c1=O	0.45607017
CC(C)C(=O)N1CCC(C(=O)NC(C(=O)NC(CCCCN)C(=O)OC(C)(C)C)C(C)c2c[nH]c3ccccc23)CC1	0.45584231
CC(C)(C)OC(=O)NCCCCC1NC(=O)C2Cc3c([nH]c4ccccc34)C(C3CCCCC3)N2C1=O	0.45329841
CCOC(=O)N1CCC(NC(=O)C2CCN(Cc3nc(-c4ccc(CC)cc4)oc3C)CC2)CC1	0.45191299
COCC(C)n1c(SCC(=O)NC2CCCCC2)nc2ccccc2c1=O	0.45083482
CCN(CC)S(=O)(=O)c1cccc(-c2nnc(SCC(=O)NC3CCCCC3C)n2N)c1	0.45056356
CCOC(=O)C=CC(=O)Nc1ccccc1CCCN1CCC23CCCCC2C1Cc1ccc(O)cc13	0.45022517
CCCCCOC(=O)N1CCN(C(=O)C(CCC(=O)O)NC(=O)c2cc(OCC3CCN(C)CC3)nc(-c3ccccc3)n2)CC1	0.44986771
CCOC(=O)C(Cc1ccccc1)NC(=O)C(C)(C)C(CC(C)C)NC(=O)c1ccc(C#N)cc1	0.44881939

Table 8.6: Best 20 SMILES and their corresponding Values for Perindopril MPO