MOLECULAR DESIGN USING GRAPH BAYESIAN OPTIMIZATION WITH SHORTEST-PATH KERNELS

Yilin Xie[⋆], **Shiqiang Zhang**[⋆], **Jixiang Qing**, **Ruth Misener**[†], **Calvin Tsay**[†]
Department of Computing, Imperial College London, UK

ABSTRACT

Past decades have seen the great potential of generative machine learning in molecular design while also exposed the gap between research contributions and practical applications. Considering the expensive-to-evaluate nature of molecular properties and hard-to-satisfy validity of molecular structures, this paper tackles molecular design from graph Bayesian optimization viewpoint, i.e., generating feasible and optimal molecular graph with compatible features given limited number of evaluations. Our proposed method, BoGrape, uses shortest-path graph kernels to measure the similarity between molecules and utilizes mixed-integer programming to allow global exploration of molecular space while maintaining the feasibility of generated candidates. Preliminary results show promising performance of BoGrape.

1 INTRODUCTION

The natural graph representations of (bio)molecules enable the application of graph machine learning to molecular prediction and generation tasks (Gaudelet et al., 2021; Xia et al., 2019; Xiong et al., 2021). To increase the validity and quality of generated candidates, numerous labeled molecules are needed to learn the underlying structure-property relationships, making molecular design data-hungry, and thus expensive. From optimization perspectives, the aim of molecular design is to learn and then optimize an unknown objective using a few data points, leading researchers to Bayesian optimization (BO) (Frazier, 2018). However, the discrete molecular space hinders the direct implementation of BO techniques, which mostly operate on continuous inputs, motivating works that introduce a continuous latent space and apply BO on it (Jin et al., 2018; Maus et al., 2022; Rittig et al., 2022). Nevertheless, the validity of solutions is still measured on the original molecular space, whose counterpart on a learned latent space is hard to define explicitly.

Alternatively, molecular generation can be viewed as a constrained graph optimization problem, i.e., finding desired graph structures and features subject to given constraints. Considering the black-box objective and limited data, recent advances in graph BO seem to be promising alternatives. Existing graph BO approaches mostly adopt Gaussian processes (GPs) with various graph kernels to fit graph functions (Ramachandram et al., 2017; Borovitskiy et al., 2021; Ru et al., 2021; Zhi et al., 2023). However, these works are impractical for (bio)molecular generation since they either (i) limit the search space to a given fixed graph (Oh et al., 2019; Wan et al., 2023; Liang et al., 2024), directed labeled graphs (White et al., 2021; Ru et al., 2021; Wan et al., 2021), unlabeled graphs (Cui & Yang, 2018), etc., or (ii) rely on task-specific similarity metrics (Kandasamy et al., 2018). Additionally, acquisition functions in graph BO are mostly optimized using evolutionary algorithms (Kandasamy et al., 2018; Wan et al., 2021) or sampling (Ru et al., 2021; Wan et al., 2023), which cannot efficiently explore the search domain, flexibly handle constraints, or guarantee optimality.

This paper proposes **B**ayesian **o**ptimization over **gra**phs with shortest-**p**ath **e**ncoded, or BoGrape, as a paradigm for optimal molecular generation. GPs are chosen as the surrogates, using global acquisition function optimization methods introduced by Xie et al. (2024). We develop four variants of the classic shortest-path graph kernel (Borgwardt & Kriegel, 2005) for use in BoGrape. By precisely representing the shortest paths as decision variables, the acquisition function optimization is formulated as a mixed-integer program (MIP) with a mixed-feature search space, graph kernel, and

[⋆]Equal contributions. Corresponding authors: {yilin.xie22,s.zhang21}@imperial.ac.uk
[†]Equal contributions.

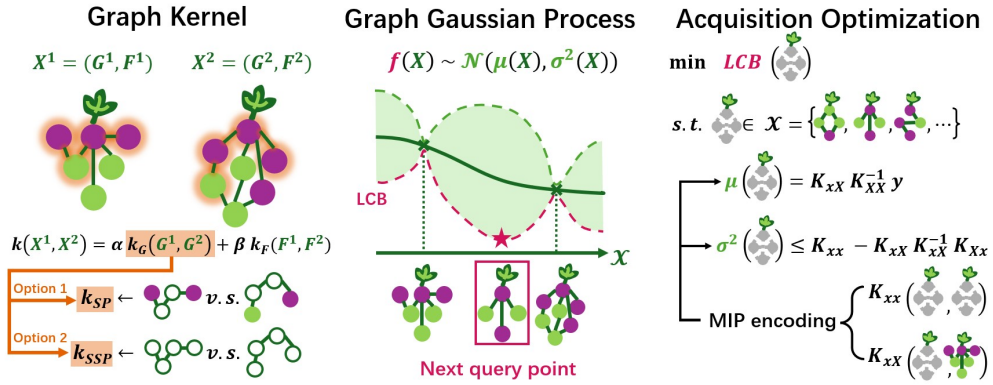


Figure 1: Key components of BoGrape. **Graph kernel** comprises k_G and k_F on the graph and feature levels respectively. **Graph GP** is then trained using the chosen kernel and samples. GP posterior is used to build the acquisition, e.g., LCB. Note that graph GP includes discrete graph domains; the continuous domain is only for illustration. **Acquisition optimization** is formulated as a MIP using the encoding of shortest paths and graph kernels. Solving the MIP gives the next query point.

relevant problem-specific constraints. Figure 1 illustrates the BoGrape pipeline. Numerical results on QM7 and QM9 datasets highlight the performance and sample efficiency of BoGrape in generating good candidate molecules.

2 Preliminaries

Bayesian optimization: BO is a derivative-free optimization framework designed to iteratively approach the optimum of an expensive-to-evaluate, black-box function $f : \mathcal{X} \to \mathbb{R}$ (Frazier, 2018). At each iteration, a surrogate model (usually GP) is trained on observed dataset, followed by selecting the next query point through optimizing an acquisition function, i.e., lower confidence bound (LCB).

Optimization over trained ML models: A trained ML model is encoded into an optimization formulation, enabling decision-making problems over model predictions. Many classic models are studied in this area, e.g., GPs (Schweidtmann et al., 2021), trees (Mistry et al., 2021; Thebelt et al., 2021; Ammari et al., 2023), neural networks (Anderson et al., 2020; Tsay et al., 2021), etc..

Global acquisition optimization: Xie et al. (2024) introduce a global acquisition optimization framework based on mixed-integer quadratic programming (MIQP) named PK-MIQP, which is useful because of its (i) compatibility with various kernels, and (ii) theoretical guarantee on regret bounds.

Graph kernels: Graph kernels extend the concept of kernels to graph domains and evaluate the similarity between graphs. Past research develops graph kernels using various graph patterns, e.g., neighborhoods, subgraphs, walks, paths. We refer to Vishwanathan et al. (2010); Borgwardt et al. (2020); Kriege et al. (2020); Nikolentzos et al. (2021) for more details about graph kernels.

3 Methodology

3.1 Shortest-path graph kernels

We focus on variants of the shortest-path (SP) kernel. For graph G, denote l_u as the label of node u, $e_{u,v}$ as the shortest path from u to v (which may not be unique), and $d_{u,v}$ as the shortest distance from node u to v (which is unique). Borgwardt & Kriegel (2005) define an SP kernel between graphs $G^1 = (V^1, E^1)$ and $G^2 = (V^2, E^2)$ as:

$$k_{SP}(G^1, G^2) = \sum_{u_1, v_1 \in V^1, u_2, v_2 \in V^2} k(e_{u_1,v_1}, e_{u_2,v_2})$$

$k(\cdot, \cdot)$ compares the labels and lengths of two shortest paths:

$$k(e_{u_1,v_1}, e_{u_2,v_2}) = k_v(l_{u_1}, l_{u_2}) \cdot k_e(d_{u_1,v_1}, d_{u_2,v_2}) \cdot k_v(l_{v_1}, l_{v_2})$$

where k_v is a kernel comparing node labels, k_e is a kernel comparing path lengths. Both k_v and k_e are usually chosen as Dirac kernels, giving the explicit representation of the SP kernel as:

$$k_{SP}(G^1, G^2) = \frac{1}{n_1^2 n_2^2} \sum_{u_1, v_1 \in V^1, u_2, v_2 \in V^2} \mathbf{1}(l_{u_1} = l_{u_2}, d_{u_1,v_1} = d_{u_2,v_2}, l_{v_1} = l_{v_2}) \tag{SP}$$

where $\frac{1}{n_1^2 n_2^2}$ is normalizing coefficient with n_1, n_2 as node number in graph G^1, G^2, respectively.

Note that each node may have more problem-specific features beyond a single label. From here on, we use $X = (G, F)$ to denote an attributed graph with G as the underlying labeled graph and F as node features. Intuitively, we can compare the features of two nodes instead of labels in k_v. However, this could unnecessarily reduce the number of matching paths between two graphs. Therefore, we borrow from Cui & Yang (2018) the idea to separate the implicit and explicit information of graphs, i.e., the kernel value between two attributed graphs X^1, X^2 becomes:

$$k(X^1, X^2) = \alpha \cdot k_G(G^1, G^2) + \beta \cdot k_F(F^1, F^2) \tag{1}$$

where k_G is any graph kernel, k_F is any kernel over features, and α, β are trainable parameters controlling the trade-off between graph similarity and feature similarity.

We also propose a simplified shortest-path (SSP) kernel corresponding to an unlabeled SP kernel:

$$k_{SSP}(G^1, G^2) = \frac{1}{n_1^2 n_2^2} \sum_{u_1, v_1 \in V^1, u_2, v_2 \in V^2} \mathbf{1}(d_{u_1,v_1} = d_{u_2,v_2}) \tag{SSP}$$

Observe that both the SP and SSP kernels are linear kernels if we pre-process all shortest paths in each graph and count the number of each length of shortest path. To improve the representation ability, non-linear kernels are considered which demonstrate better empirical performance in exchange of the additional difficulties for optimization. Motivated by the practically strong performance of exponential kernels such as RBF kernel, Matérn kernel, graph diffusion kernel (Oh et al., 2019), etc., we propose the following two nonlinear graph kernels based on SP and SSP kernels:

$$k_{ESP}(G^1, G^2) = \exp(k_{SP}(G^1, G^2))/\sigma_k^2 \tag{ESP}$$

$$k_{ESSP}(G^1, G^2) = \exp(k_{SSP}(G^1, G^2))/\sigma_k^2 \tag{ESSP}$$

where variance σ_k^2 is added to control the magnitude of kernel value. Note that we could also add variance to SP and SSP kernels, but it would be absorbed by α, β.

3.2 Global acquisition optimization

We begin with the optimization formulation for the LCB acquisition function from Xie et al. (2024):

$$\min \mu - \beta_t^{1/2} \sigma \tag{3a}$$

$$\text{s.t. } \mu = K_{xX} K_{XX}^{-1} \boldsymbol{y} \tag{3b}$$

$$\sigma^2 \leq K_{xx} - K_{xX} K_{XX}^{-1} K_{Xx} \tag{3c}$$

$$K_{xX^i} = k(x, X^i),\ \forall 1 \leq i < t \tag{3d}$$

$$x = (G, F) \in \mathcal{X} = \mathcal{G} \times \mathcal{F} \tag{3e}$$

To maintain consistency with the general BO setting, we denote $x = (G, F)$ as the next sample and $X = \{(G^i, F^i), y^i\}_{i=1}^{t-1}$ as the prior samples at the t-th iteration. The difference is that now we need to optimize over both the graph domain $G \in \mathcal{G}$ and the feature domain $F \in \mathcal{F}$.

The advantages of formulation Eq. (3) are: (i) it is compatible with discrete variables, a key challenge of graph optimization, (ii) Eq. (3e) allows problem-specific constraints over the graph domain, (iii) Eq. (3d) is generic to the choice of graph kernel, and (iv) nonlinear kernels can be piecewise-linearly approximated and incorporated into Eq. (3) with theoretical guarantees on regret bounds.

3.3 Encoding of graph kernels

The explicit formulation of Eq.(3e) needs expressions of K_{xX^i} and K_{xx}, both of which involve the shortest path between any two nodes. Deriving shortest paths for a given graph is straightforward

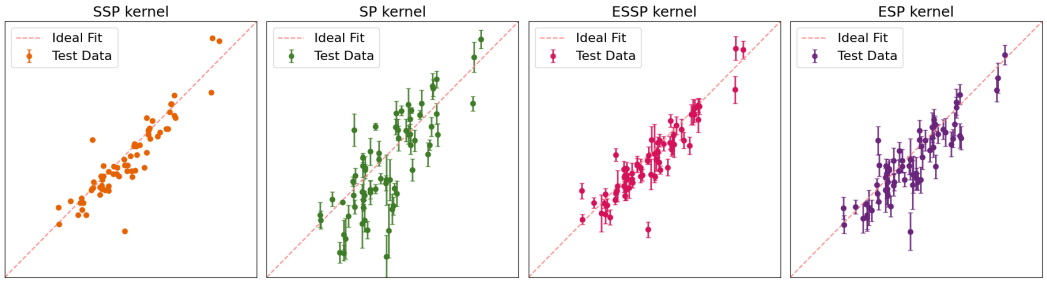


Figure 2: Predictive performance of GP with different kernels. 100 samples are randomly chosen from the QM7 dataset, 30 of which are used for training. The predictive mean with one standard deviation (predicted y) of the remaining 70 graphs are plotted against their real values (true y).

using classic shortest-path algorithms, while representing them into decision variables is quite complicated and thus the major contribution of this work. Due to space limit, we put all details about the encoding of shortest paths in Appendix A, where Theorem A.2 and Theorem A.3 guarantee the equivalence of our encoding for directed graphs. With shortest paths encoded, Appendix B formulates graph kernels introduced in Section 3.1. All encoding could be further simplified for undirected graphs, e.g., molecules (see Appendix B.3).

4 Experiments

In this preliminary work, we consider the performance of our method in both prediction and optimization tasks. Datasets QM7 (Blum & Reymond, 2009; Rupp et al., 2012) and QM9 (Ruddigkeit et al., 2012; Ramakrishnan et al., 2014) are chosen as real-world case studies, each of which consists of molecules with quantum mechanic properties. Each molecule is represented as a graph with $F = 15$ node features, including $L = 4$ labels. Since the maximal size of molecules is 7 and 9 for QM7 and QM9, respectively, we follow Zhang et al. (2023) and train a GNN as a predictor for each dataset.

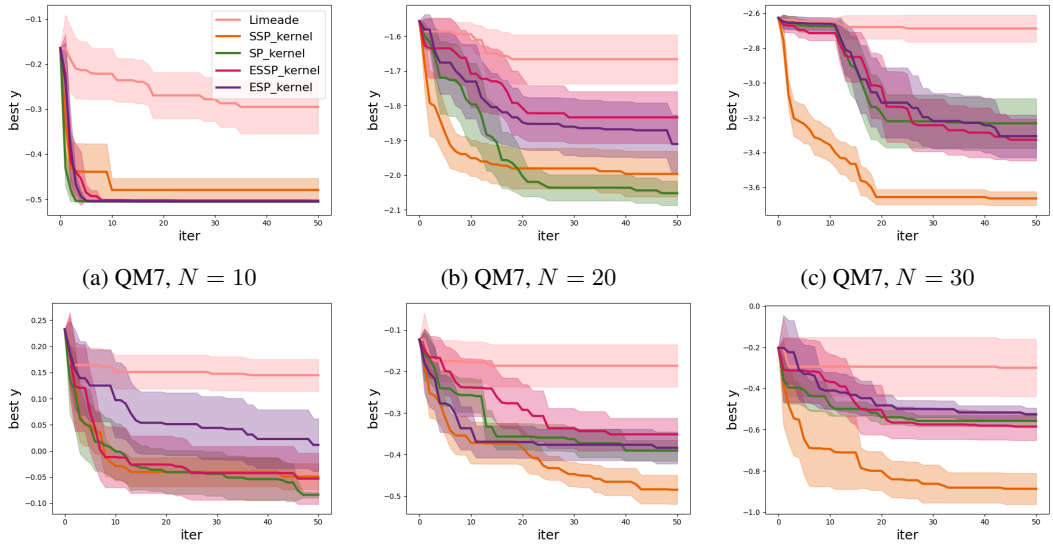
All experiments are performed on a 4.2 GHz Intel Core i7-7700K CPU with 16 GB memory. We use GPflow (Matthews et al., 2017) to implement GP models, PyG (Fey & Lenssen, 2019) to implement GNNs, and Gurobi v11.0.0 (Gurobi Optimization, LLC, 2024) to solve MIPs. Random sampling is a common baseline, but is excluded here since it rarely produces even feasible solutions. Instead, we use Limeade (Zhang et al., 2024b) to randomly generate feasible molecules, which is further enhanced by incorporating the composition constraints and symmetry-breaking constraints proposed by Zhang et al. (2023). Appendix C.2 compares random sampling and Limeade.

4.1 Molecular properties prediction

We first test the predictive performance of GPs with the four graph kernels. Based on the molecular size N, we consider two settings: (a) if the dataset includes molecules of size N, we randomly choose molecules from the dataset and use their real properties, and (b) for larger N, we use Limeade to generate molecules and use the trained GNN to predict their properties. To show the performance of different kernels on representing similarity between graphs, we apply setting (a) and perform a property prediction task using GPs equipped with the various kernels in Figure 2. For larger graph sizes, we apply setting (b) and report the root mean square errors and mean negative log likelihoods of GP regression in Tables 2 and 3 (see Appendix C.1). Figure 2 concludes the four graph kernels have comparable prediction performance in terms of accuracy, while two exponential kernels may more accurately quantify uncertainty (which is also observed in Table 3). When graph size N is larger, Table 2 shows that the more complicated kernels, i.e., SP and ESP, are generally better at predicting graph properties since they impose stronger criteria on comparing shortest-paths between two graphs.

4.2 Optimal molecular design

Now we have presented all the pieces needed to implement an end-to-end BO procedure. We employ the trained GNNs used in Section 4.1 as oracle predictors, i.e., the functions that we seek to optimize.



(a) QM7, $N = 10$ (b) QM7, $N = 20$ (c) QM7, $N = 30$

(d) QM9, $N = 10$ (e) QM9, $N = 20$ (f) QM9, $N = 30$

Figure 3: Bayesian optimization results on QM7 and QM9 with $N \in \{10, 20, 30\}$. Best objective value is plotted at each iteration. Mean with 0.5 standard deviation over 10 replications is reported.

At each iteration, we train a GP with a graph kernel and extract trained model parameters $\alpha, \beta, \sigma_k^2$ to calculate K_{XX} and represent K_{xx}, K_{xX_i} appearing in Eqs. (3b)–(3d). Eq. (3e) defines the search domain, which is already set up in Zhang et al. (2024a) for QM7 and QM9. Solving the final MIP suggests the next molecule to query. Algorithm 1 outlines BoGrape.

Algorithm 1 BoGrape at t-th iteration.

1: **Input:** dataset $X = \{(G^i, F^i), y^i\}_{i=1}^{t-1}$, graph kernel $\in$ {SSP, SP, ESSP, ESP}.
2: **Model training:** kernel parameters $\alpha, \beta, \sigma_k^2$ ▷ graph GP fit to X.
3: **Acquisition formulation:**
4: define objective in Eq. (3a) ▷ definition of LCB.
5: encode K_{xX^i} and K_{xx} in Eqs. (3b) – (3d) ▷ Appendix B.
6: search space $\mathcal{X}$ in Eq. (3e) ▷ problem-specific.
7: **Optimization:** optimal solution (G^t, F^t) ▷ solve Eq. (3) (with warm start).
8: **Output:** proposed sample (G^t, F^t).

In our experiments, 10 molecules generated by Limeade are used as the initial dataset, and 50 BO iterations are performed with 600s as the time limit for each iteration. To initialize the algorithm with some feasible solutions, i.e., good primal solutions, we use 20 molecules generated by Limeade together with previously sampled molecules to warm start Eq. (3) before solving it. For each BO run, we show the mean with 0.5 standard deviation of the best objective value over 10 replications.

Results in Figure 3 show that BoGrape outperforms Limeade regardless of which graph kernel is used. The SSP kernel displays the best performance in most cases, especially when the graph size N is large. This observation suggests that this simpler encoding reduces model complexity and produces better solutions within the given computational time. The SP kernel, which includes stricter comparison between graphs and has outstanding predictive performance as shown in Table 2, outperforms the SSP kernel in some cases, e.g. Figures 3a, 3b, 3d. Exponential kernels have good representation ability, at the trade-off that their formulations result in more complicated MIPs and require more computational resources for good performance. We hypothesize that BO with the exponential kernels may be more capable of providing high-quality solutions giving longer computational time.

5 CONCLUSION

This work proposes BoGrape to optimize black-box functions over graphs. Four shortest-path graph kernels are presented and tested on both prediction and Bayesian optimization tasks. The underlying mixed-integer formulation provides a flexible and general platform including mixed-feature search spaces, graph kernels, acquisition functions, and problem-specific constraints. BoGrape shows its ability to generate feasible and qualified molecular graphs, leveraging the data-insufficiency and validity requirement in practice. Numerical results show promising performance and suggest trade-offs between query-efficiency and computational time when choosing a suitable kernel.

ACKNOWLEDGMENTS

The authors gratefully acknowledge support from a Department of Computing Scholarship (YX), BASF SE, Ludwigshafen am Rhein (SZ), Engineering and Physical Sciences Research Council [grants EP/W003317/1, EP/X025292/1, and EP/Y028775/1] (RM, CT, JQ), a BASF/RAEng Research Chair in Data-Driven Optimisation (RM), and a BASF/RAEng Senior Research Fellowship (CT).

REFERENCES

Bashar L. Ammari, Emma S. Johnson, Georgia Stinchfield, Taehun Kim, Michael Bynum, William E. Hart, Joshua Pulsipher, and Carl D. Laird. Linear model decision trees as surrogates in optimization of engineering applications. *Computers & Chemical Engineering*, 178, 2023.

Ross Anderson, Joey Huchette, Will Ma, Christian Tjandraatmadja, and Juan Pablo Vielma. Strong mixed-integer programming formulations for trained neural networks. *Mathematical Programming*, 183(1):3–39, 2020.

Lorenz C. Blum and Jean-Louis Reymond. 970 million druglike small molecules for virtual screening in the chemical universe database GDB-13. *Journal of the American Chemical Society*, 131(25): 8732–8733, 2009.

Karsten Borgwardt, Elisabetta Ghisu, Felipe Llinares-López, Leslie O'Bray, Bastian Rieck, et al. Graph kernels: State-of-the-art and future challenges. *Foundations and Trends® in Machine Learning*, 13(5-6):531–712, 2020.

Karsten M Borgwardt and Hans-Peter Kriegel. Shortest-path kernels on graphs. In *International Conference on Data Mining*, 2005.

Viacheslav Borovitskiy, Iskander Azangulov, Alexander Terenin, Peter Mostowsky, Marc Peter Deisenroth, and Nicolas Durrande. Matern Gaussian processes on graphs. In *International Conference on Artificial Intelligence and Statistics*, 2021.

Jiaxu Cui and Bo Yang. Graph Bayesian optimization: Algorithms, evaluations and applications. *arXiv preprint arXiv:1805.01157*, 2018.

Matthias Fey and Jan E. Lenssen. Fast graph representation learning with PyTorch Geometric. In *ICLR 2019 Workshop on Representation Learning on Graphs and Manifolds*, 2019.

Robert W Floyd. Algorithm 97: Shortest path. *Communications of the ACM*, 5(6):345–345, 1962.

Peter I Frazier. A tutorial on Bayesian optimization. *arXiv preprint arXiv:1807.02811*, 2018.

Thomas Gaudelet, Ben Day, Arian R Jamasb, Jyothish Soman, Cristian Regep, Gertrude Liu, Jeremy BR Hayter, Richard Vickers, Charles Roberts, Jian Tang, et al. Utilizing graph machine learning within drug discovery and development. *Briefings in Bioinformatics*, 22(6):bbab159, 2021.

Gurobi Optimization, LLC. Gurobi optimizer reference manual, 2024. URL `https://www.gurobi.com`.

Wengong Jin, Regina Barzilay, and Tommi Jaakkola. Junction tree variational autoencoder for molecular graph generation. In *ICML*, 2018.

Kirthevasan Kandasamy, Willie Neiswanger, Jeff Schneider, Barnabas Poczos, and Eric P Xing. Neural architecture search with Bayesian optimisation and optimal transport. *NeurIPS*, 31, 2018.

Nils M Kriege, Fredrik D Johansson, and Christopher Morris. A survey on graph kernels. *Applied Network Science*, 5:1–42, 2020.

Huidong Liang, Xingchen Wan, and Xiaowen Dong. Bayesian optimization of functions over node subsets in graphs. *arXiv preprint arXiv:2405.15119*, 2024.

Alexander G. de G. Matthews, Mark van der Wilk, Tom Nickson, Keisuke. Fujii, Alexis Boukouvalas, Pablo León-Villagrá, Zoubin Ghahramani, and James Hensman. GPflow: A Gaussian process library using TensorFlow. *Journal of Machine Learning Research*, 18(40):1–6, 2017.

Natalie Maus, Haydn Jones, Juston Moore, Matt J Kusner, John Bradshaw, and Jacob Gardner. Local latent space Bayesian optimization over structured inputs. In *NeurIPS*, 2022.

Miten Mistry, Dimitrios Letsios, Gerhard Krennrich, Robert M. Lee, and Ruth Misener. Mixed-integer convex nonlinear optimization with gradient-boosted trees embedded. *INFORMS Journal on Computing*, 33(3):1103–1119, 2021.

Giannis Nikolentzos, Giannis Siglidis, and Michalis Vazirgiannis. Graph kernels: A survey. *Journal of Artificial Intelligence Research*, 72:943–1027, 2021.

Changyong Oh, Jakub Tomczak, Efstratios Gavves, and Max Welling. Combinatorial Bayesian optimization using the graph cartesian product. *NeurIPS*, 2019.

Dhanesh Ramachandram, Michal Lisicki, Timothy J Shields, Mohamed R Amer, and Graham W Taylor. Structure optimization for deep multimodal fusion networks using graph-induced kernels. *arXiv preprint arXiv:1707.00750*, 2017.

Raghunathan Ramakrishnan, Pavlo O. Dral, Matthias Rupp, and O. Anatole Von Lilienfeld. Quantum chemistry structures and properties of 134 kilo molecules. *Scientific Data*, 1(1):1–7, 2014.

Jan G. Rittig, Martin Ritzert, Artur M. Schweidtmann, Stefanie Winkler, Jana M. Weber, Philipp Morsch, Karl Alexander Heufer, Martin Grohe, Alexander Mitsos, and Manuel Dahmen. Graph machine learning for design of high-octane fuels. *AIChE Journal*, pp. e17971, 2022.

Binxin Ru, Xingchen Wan, Xiaowen Dong, and Michael Osborne. Interpretable neural architecture search via Bayesian optimisation with Weisfeiler-Lehman kernels. In *ICLR*, 2021.

Lars Ruddigkeit, Ruud Van Deursen, Lorenz C. Blum, and Jean-Louis Reymond. Enumeration of 166 billion organic small molecules in the chemical universe database gdb-17. *Journal of Chemical Information and Modeling*, 52(11):2864–2875, 2012.

Matthias Rupp, Alexandre Tkatchenko, Klaus-Robert Müller, and O. Anatole Von Lilienfeld. Fast and accurate modeling of molecular atomization energies with machine learning. *Physical Review Letters*, 108(5):058301, 2012.

Artur M Schweidtmann, Dominik Bongartz, Daniel Grothe, Tim Kerkenhoff, Xiaopeng Lin, Jaromił Najman, and Alexander Mitsos. Deterministic global optimization with Gaussian processes embedded. *Mathematical Programming Computation*, 13(3):553–581, 2021.

Alexander Thebelt, Jan Kronqvist, Miten Mistry, Robert M. Lee, Nathan Sudermann-Merx, and Ruth Misener. Entmoot: A framework for optimization over ensemble tree models. *Computers & Chemical Engineering*, 151:107343, 2021.

Calvin Tsay, Jan Kronqvist, Alexander Thebelt, and Ruth Misener. Partition-based formulations for mixed-integer optimization of trained ReLU neural networks. In *NeurIPS*, 2021.

Stefan Vigerske and Ambros Gleixner. SCIP: Global optimization of mixed-integer nonlinear programs in a branch-and-cut framework. *Optimization Methods and Software*, 33(3):563–593, 2018.

S Vichy N Vishwanathan, Nicol N Schraudolph, Risi Kondor, and Karsten M Borgwardt. Graph kernels. *The Journal of Machine Learning Research*, 11:1201–1242, 2010.

Xingchen Wan, Henry Kenlay, Binxin Ru, Arno Blaas, Michael Osborne, and Xiaowen Dong. Adversarial attacks on graph classifiers via Bayesian optimisation. In *NeurIPS*, 2021.

Xingchen Wan, Pierre Osselin, Henry Kenlay, Binxin Ru, Michael A Osborne, and Xiaowen Dong. Bayesian optimisation of functions on graphs. *NeurIPS*, 2023.

Colin White, Willie Neiswanger, and Yash Savani. Bananas: Bayesian optimization with neural architectures for neural architecture search. In *AAAI*, 2021.

Xiaolin Xia, Jianxing Hu, Yanxing Wang, Liangren Zhang, and Zhenming Liu. Graph-based generative models for *de Novo* drug design. *Drug Discovery Today: Technologies*, 32:45–53, 2019.

Yilin Xie, Shiqiang Zhang, Joel Paulson, and Calvin Tsay. Global optimization of Gaussian process acquisition functions using a piecewise-linear kernel approximation. *arXiv preprint arXiv:2410.16893*, 2024.

Jiacheng Xiong, Zhaoping Xiong, Kaixian Chen, Hualiang Jiang, and Mingyue Zheng. Graph neural networks for automated *de novo* drug design. *Drug Discovery Today*, 26(6):1382–1393, 2021.

Shiqiang Zhang, Juan S. Campos, Christian Feldmann, David Walz, Frederik Sandfort, Miriam Mathea, Calvin Tsay, and Ruth Misener. Optimizing over trained GNNs via symmetry breaking. In *NeurIPS*, 2023.

Shiqiang Zhang, Juan S Campos, Christian Feldmann, Frederik Sandfort, Miriam Mathea, and Ruth Misener. Augmenting optimization-based molecular design with graph neural networks. *Computers & Chemical Engineering*, 186:108684, 2024a.

Shiqiang Zhang, Christian W Feldmann, Frederik Sandfort, Miriam Mathea, Juan S Campos, and Ruth Misener. Limeade: Let integer molecular encoding aid. *arXiv preprint arXiv:2411.16623*, 2024b.

Yin-Cong Zhi, Yin Cheng Ng, and Xiaowen Dong. Gaussian processes on graphs via spectral kernel learning. *IEEE Transactions on Signal and Information Processing over Networks*, 2023.

A ENCODING OF SHORTEST PATHS

A.1 NOTATIONS

We provide details for all variables introduced in this paper in Table 1. Recall that the search domain considered here consists of all connected graphs with node number ranging from n_0 to n, each node has M binary features with the first L node features as the one-hot encoding of node label. Use $[n]$ to denote set $\{0, 1, \ldots, n-1\}$.

Table 1: All variables introduced in the optimization formulation for graph kernels.

Variables	Domain	Description
$A_{u,v},\ u, v \in [n]$	$\{0,1\}$	the existence of edge from u to v
$d_{u,v},\ u, v \in [n]$	$[n+1]$	the length of shortest path from u to v
$\delta^w_{u,v},\ u, v, w \in [n]$	$\{0,1\}$	if w appears at the shortest path from u to v
$d^s_{u,v},\ u, v \in [n], s \in [n+1]$	$\{0,1\}$	indicator: $\mathbf{1}(d_{u,v} = s)$
$D_s,\ s \in [n]$	$[n^2+1]$	# shortest paths with length s
$D^c_s,\ s \in [n], c \in [n^2+1]$	$\{0,1\}$	indicator: $\mathbf{1}(D_s = c)$
$p^{s,l_1,l_2}_{u,v},\ u, v, s \in [n], l_1, l_2 \in [L]$	$\{0,1\}$	indicator: $\mathbf{1}(F_{u,l_1} = 1, d_{u,v} = s, F_{v,l_2} = 1)$
$P_{s,l_1,l_2},\ s \in [n], l_1, l_2 \in [L]$	$[n^2+1]$	# shortest paths with length s and labels l_1, l_2
$P^c_{s,l_1,l_2},\ s \in [n], l_1, l_2 \in [L], c \in [n^2+1]$	$\{0,1\}$	indicator: $\mathbf{1}(P_{s,l_1,l_2} = c)$
$N^m,\ m \in [M]$	$[N+1]$	sum of m-th feature over all nodes
$N^c_m,\ m \in [M], c \in [M+1]$	$\{0,1\}$	indicator: $\mathbf{1}(N_m = c)$

A.2 SHORTEST PATH ENCODING FOR GRAPHS WITH FIXED SIZE

For the sake of exposition, we start with all connected graphs G with fixed size, i.e., node number n is given. The optimization formulation for graph kernels involves constant graph information and their variable counterparts. For each variable Var, we use $Var(G)$ to denote its value on a given graph G. For example, $d_{u,v}(G)$ is the shortest distance from node u to node v in graph G.

If graph G is given, $A_{u,v}, d_{u,v}, \delta^w_{u,v}$ in Table 1 can be computed using classic shortest-path algorithms, such as the Floyd–Warshall algorithm (Floyd, 1962). In graph optimization tasks, however, we need to encode the relationships between these variables as constraints. Motivated by the Floyd–Warshall algorithm, we first present the constraints in Eq. (4) and then prove that there exists a bijective between the feasible domain given by these constraints and all connected graphs with size n.

$$A_{v,v} = 1, \qquad \forall v \in [n] \tag{4a}$$

$$d_{v,v} = 0, \qquad \forall v \in [n] \tag{4b}$$

$$d_{u,v} \begin{cases} = 1, & \text{if } A_{u,v} = 1 \\ > 1, & \text{if } A_{u,v} = 0 \end{cases}, \qquad \forall u, v \in [n],\ u \neq v \tag{4c}$$

$$d_{u,v} \begin{cases} = d_{u,w} + d_{w,v}, & \text{if } \delta^w_{u,v} = 1 \\ < d_{u,w} + d_{w,v}, & \text{if } \delta^w_{u,v} = 0 \end{cases}, \qquad \forall u, v, w \in [n],\ u \neq v \tag{4d}$$

$$\delta^w_{v,v} = \begin{cases} 1, & \text{if } w = v \\ 0, & \text{if } w \neq v \end{cases}, \qquad \forall v, w \in [n] \tag{4e}$$

$$\delta^u_{u,v} = \delta^v_{u,v} = 1, \qquad \forall u, v \in [n],\ u \neq v \tag{4f}$$

$$\sum_{w \in [n]} \delta^w_{u,v} \begin{cases} = 2, & \text{if } A_{u,v} = 1 \\ > 2, & \text{if } A_{u,v} = 0 \end{cases}, \qquad \forall u, v \in [n],\ u \neq v \tag{4g}$$

Eq. (4) are necessary conditions that $A_{u,v}, d_{u,v}, \delta^w_{u,v}$ should satisfy:

- Eq. (4a) initializes the diagonal elements.

- Eq. (4b) initializes the shortest distance from v to itself.
- Eq. (4c) forces the shortest distance from node u and v be 1 if edge $u \to v$ exists, and larger than 1 otherwise.
  Rewrite Eq. (4c) as:
  $$\begin{aligned} d_{u,v} &\le 1 + n \cdot (1 - A_{u,v}), && \forall u, v \in [n],\ u \neq v \\ d_{u,v} &\ge 2 - A_{u,v}, && \forall u, v \in [n],\ u \neq v \end{aligned}$$
  where n is a big-M coefficient using $d_{u,v} \le n - 1$.
- Eq. (4d) is the triangle inequality for distance matrix d.
  Rewrite Eq. (4d) as:
  $$\begin{aligned} d_{u,v} &\le d_{u,w} + d_{w,v} - (1 - \delta^w_{u,v}), && \forall u, v, w \in [n] \\ d_{u,v} &\ge d_{u,w} + d_{w,v} - 2n \cdot (1 - \delta^w_{u,v}), && \forall u, v, w \in [n] \end{aligned}$$
  where $2n$ is a big-M coefficient since $d_{u,w} + d_{w,v} < 2n$.
- Eq. (4e) initializes $\delta^w_{v,v}$ by definition.
- Eq. (4f) initializes $\delta^u_{u,v}$ and $\delta^v_{u,v}$ by definition.
- Eq. (4g) ensures that there is at least one node at the shortest path from node u to v if there is no edge from node u to v. Otherwise, no node except for u and v could appear at the shortest path from u to v.
  Rewrite Eq. (4g) as:
  $$\begin{aligned} \sum_{w \in [n]} \delta^w_{u,v} &\le 2 + (n-2) \cdot (1 - A_{u,v}), && \forall u, v \in [n],\ u \neq v \\ \sum_{w \in [n]} \delta^w_{u,v} &\ge 2 + (1 - A_{u,v}), && \forall u, v \in [n],\ u \neq v \end{aligned}$$
  where $n - 2$ is a big-M coefficient since $\sum_{w \in [n]} \delta^w_{u,v} \le n$.

Replacing disjunctive constraints accordingly in Eq. (4) gives the final formulation Eq. (MIP-SP):

$$\begin{cases} A_{v,v} = 1, & \forall v \in [n] \\ d_{v,v} = 0, & \forall v \in [n] \\ d_{u,v} \le 1 + n \cdot (1 - A_{u,v}), & \forall u, v \in [n],\ u \neq v \\ d_{u,v} \ge 2 - A_{u,v}, & \forall u, v \in [n],\ u \neq v \\ d_{u,v} \le d_{u,w} + d_{w,v} - (1 - \delta^w_{u,v}), & \forall u, v, w \in [n] \\ d_{u,v} \ge d_{u,w} + d_{w,v} - 2n \cdot (1 - \delta^w_{u,v}), & \forall u, v, w \in [n] \\ \delta^v_{v,v} = 1, & \forall v \in [n] \\ \delta^w_{v,v} = 0, & \forall v, w \in [n],\ v \neq w \\ \delta^u_{u,v} = \delta^v_{u,v} = 1, & \forall u, v \in [n],\ u \neq v \\ \sum_{w \in [n]} \delta^w_{u,v} \le 2 + (n-2) \cdot (1 - A_{u,v}), & \forall u, v \in [n],\ u \neq v \\ \sum_{w \in [n]} \delta^w_{u,v} \ge 2 + (1 - A_{u,v}), & \forall u, v \in [n],\ u \neq v \end{cases} \tag{MIP-SP}$$

Lemma A.1. *$(A_{u,v}(G), d_{u,v}(G), \delta^w_{u,v}(G))$ is a feasible solution of Eq. (MIP-SP) with size $n = n(G)$ given any connected graph G.*

Proof. Trivial to verify by definition. □

Theorem A.2. *Given any $n \in \mathbb{Z}^+$, for any feasible solution $(A_{u,v}, d_{u,v}, \delta^w_{u,v})$ of Eq. (MIP-SP) with size n, there exists a unique graph G such that:*

$$(A_{u,v}(G), d_{u,v}(G), \delta^w_{u,v}(G)) = (A_{u,v}, d_{u,v}, \delta^w_{u,v})$$

i.e., there is a bijective between the feasible domain of Eq. (MIP-SP) with size n and the set consisting of all connected graphs with n nodes.

Proof. If such G exists, it is unique since $A_{u,v}$ gives the existence of every edge. Thus it suffices to show that $(d_{u,v}(G), \delta^w_{u,v}(G)) = (d_{u,v}, \delta^w_{u,v})$ for G defined with $A_{u,v}$.

We are going to prove it by induction on the shortest distance sd from node u to v in graph G. Specifically, we want to show that for any $0 \leq sd < n$, and for any pair of (u, v) such that $\min(d_{u,v}(G), d_{u,v}) = sd$, we have $d_{u,v}(G) = d_{u,v}$ and $\delta^w_{u,v}(G) = \delta^w_{u,v},\ \forall w \in [n]$.

For $sd = 0$, $\min(d_{u,v}(G), d_{u,v}) = 0$ if and only if $u = v$. For any $v \in [n]$, it is obvious to have:

$$\begin{aligned} d_{v,v}(G) &= 0 = d_{v,v} \\ \delta^v_{v,v}(G) &= 1 = \delta^v_{v,v} \\ \delta^w_{v,v}(G) &= 0 = \delta^w_{v,v},\ \forall w \neq v \end{aligned}$$

For $sd = 1$, consider every pair (u, v) such that $d_{u,v}(G) = 1$, we have $A_{u,v} = A_{u,v}(G) = 1$, then it is easy to obtain:

$$\begin{aligned} d_{u,v}(G) &= 1 = d_{u,v} \\ \delta^w_{u,v}(G) &= 1 = \delta^w_{u,v},\ \forall w \in \{u, v\} \\ \delta^w_{u,v}(G) &= 0 = \delta^w_{u,v},\ \forall w \notin \{u, v\} \end{aligned}$$

where $\delta^w_{u,v} = 0,\ \forall w \notin \{u, v\}$ since:

$$\sum_{w \notin \{u,v\}} \delta^w_{u,v} = \sum_{w \in [n]} \delta^w_{u,v} - \delta^u_{u,v} - \delta^v_{u,v} = 0$$

On the contrary, $d_{u,v} = 1$ gives $A_{u,v} = 1$, thus $A_{u,v}(G) = 1$ and $\delta^w_{u,v}(G) = \delta^w_{u,v},\ \forall w$ by definition.

Now assume that for any pair of (u, v) such that $\min(d_{u,v}(G), d_{u,v}) \leq sd$, we have $d_{u,v}(G) = d_{u,v}$ and $\delta^w_{u,v}(G) = \delta^w_{u,v},\ \forall w$. Since $\delta^w_{u,v}(G) = \delta^w_{u,v},\ \forall w \in \{u, v\}$ always holds by definition, we only consider $w \notin \{u, v\}$.

Part 1: We first consider every pair of (u, v) such that $d_{u,v}(G) = sd + 1$. Since $sd + 1 \geq 2$, we know that $A_{u,v} = A_{u,v}(G) = 0$ and there exists $w \notin \{u, v\}$ on the shortest path from node u to v in graph G.

Case 1.1: For every $w \notin \{u, v\}$ such that $\delta^w_{u,v}(G) = 1$, since $d_{u,w}(G) \leq sd$ and $d_{w,v}(G) \leq sd$, we have:

$$d_{u,v} \leq d_{u,w} + d_{w,v} = d_{u,w}(G) + d_{w,v}(G) = d_{u,v}(G) = sd + 1$$

The equality has to hold, otherwise, $d_{u,v} \leq sd$ gives $d_{u,v}(G) = d_{u,v} \leq sd$ by assumption. Therefore, $\delta^w_{u,v} = 1 = \delta^w_{u,v}(G)$.

Case 1.2: For every $w \notin \{u, v\}$ such that $\delta^w_{u,v}(G) = 0$, if $\delta^w_{u,v} = 1$, then $d_{u,w} + d_{w,v} = d_{u,v} = sd + 1$, which means that $d_{u,w} \leq sd$ and $d_{w,v} \leq sd$. By assumption, we have $d_{u,w}(G) = d_{u,w}, d_{w,v}(G) = d_{w,v}$ and then:

$$d_{u,w}(G) + d_{w,v}(G) = d_{u,w} + d_{w,v} = d_{u,v} = d_{u,v}(G)$$

which contradicts to $\delta^w_{u,v}(G) = 0$. Thus $\delta^w_{u,v} = 0$.

Part 2: Then we consider every pair of (u, v) such that $d_{u,v} = sd + 1$. Similarly, we have $A_{u,v} = A_{u,v}(G) = 0$.

Case 2.1: For every $w \notin \{u, v\}$ such that $\delta^w_{u,v} = 1$, since $d_{u,w} \leq sd$ and $d_{w,v} \leq sd$, we have $d_{u,w}(G) = d_{u,v}$ and $d_{w,v}(G) = d_{w,v}$, then:

$$d_{u,v}(G) \leq d_{u,w}(G) + d_{w,v}(G) = d_{u,w} + d_{w,v} = d_{u,v} = sd + 1$$

This equality also has to hold, otherwise, $d_{u,v}(G) \leq sd$, by assumption $d_{u,v} = d_{u,v}(G) \leq sd$, which is a contradiction.

Case 2.2: For every $w \notin \{u, v\}$ such that $\delta^w_{u,v} = 0$, if $\delta^w_{u,v}(G) = 1$, then $d_{u,w}(G) = d_{w,v}(G) = d_{u,v}(G) = sd + 1$, which means that $d_{u,w}(G) \leq sd$ and $d_{w,v}(G) \leq sd$. Therefore,

$$d_{u,w} + d_{w,v} = d_{u,w}(G) + d_{w,v}(G) = d_{u,v}(G) = d_{u,v}$$

which contradicts to $\delta_{u,v} = 0$. □

A.3 SHORTEST PATH ENCODING FOR GRAPH WITH UNKNOWN SIZE

The formulation becomes more complicated when the graph size is unknown (but bounded). Denote n_0 and n as the minimal and maximal number of nodes, respectively, and use $A_{v,v}$ to represent the existence of node v. We need to assign proper values to $d_{u,v}$ and $\delta^w_{u,v}$ when either of u or v does not exist. Moreover, we extend the domain of $d_{u,v}$ from $[n]$ to $[n+1]$ and use n to denote infinity.

We extend constraints listed in Eq. (4) to handle changeable graph size:

$$A_{v,v} \geq A_{v+1,v+1}, \qquad \forall v \in [n-1] \tag{5a}$$

$$\sum_{v\in[n]} A_{v,v} \geq n_0, \tag{5b}$$

$$2A_{u,v} \leq A_{u,u} + A_{v,v}, \qquad \forall u, v \in [n], u \neq v \tag{5c}$$

$$d_{v,v} = 0, \qquad \forall v \in [n] \tag{5d}$$

$$d_{u,v} \begin{cases} = 1, & A_{u,v} = 1 \\ > 1, & A_{u,v} = 0 \end{cases}, \qquad \forall u, v \in [n],\ u \neq v \tag{5e}$$

$$d_{u,v} \begin{cases} < n, & \text{if } A_{u,u} = A_{v,v} = 1 \\ = n, & \text{if } \min\{A_{u,u}, A_{v,v}\} = 0 \end{cases}, \qquad \forall u, v \in [n],\ u \neq v \tag{5f}$$

$$d_{u,v} \begin{cases} = d_{u,w} + d_{w,v}, & \text{if } \delta^w_{u,v} = 1 \\ < d_{u,w} + d_{w,v}, & \text{if } \delta^w_{u,v} = 0 \end{cases}, \qquad \forall u, v, w \in [n],\ u \neq v \tag{5g}$$

$$\delta^w_{v,v} = \begin{cases} 1, & \text{if } w = v \\ 0, & \text{if } w \neq v \end{cases}, \qquad \forall v, w \in [n] \tag{5h}$$

$$\delta^u_{u,v} = \delta^v_{u,v} = 1, \qquad \forall u, v \in [n],\ u \neq v \tag{5i}$$

$$\sum_{w\in[n]} \delta^w_{u,v} \begin{cases} = 2, & \text{if } A_{u,v} = 1 \\ > 2, & \text{if } A_{u,v} = 0,\ A_{u,u} = A_{v,v} = 1 \\ = 2, & \text{if } \min\{A_{u,u}, A_{v,v}\} = 0 \end{cases}, \qquad \forall u, v \in [n],\ u \neq v \tag{5j}$$

where:

- Eq. (5a) forces nodes with smaller indexes exist.
- Eq. (5b) gives the lower bound of the number of existed nodes.
- Eq. (5c) means that there is no edge from node u to v if any of them does not exist.
- Eq. (5d) initializes the shortest distance from one node to itself, even it does not exist.
- Eq. (5e) forces the shortest distance from node u and v be 1 if there is one edge from u to v, and larger that 1 otherwise.
  Rewrite Eq. (5e) as:
  $$\begin{aligned} d_{u,v} &\leq 1 + n \cdot (1 - A_{u,v}), && \forall u, v \in [n],\ u \neq v \\ d_{u,v} &\geq 2 - A_{u,v}, && \forall u, v \in [n],\ u \neq v \end{aligned}$$
  where n is a big-M coefficient using the fact that $d_{u,v} \leq n$.
- Eq. (5f) sets the shortest distance from node u to v as n, i.e., ∞, if any of them does not exist. Otherwise, the shortest distance is less than n.
  Rewrite Eq. (5f) as:
  $$\begin{aligned} d_{u,v} &\geq n \cdot (1 - A_{u,u}), && \forall u, v \in [n],\ u \neq v \\ d_{u,v} &\geq n \cdot (1 - A_{v,v}), && \forall u, v \in [n],\ u \neq v \end{aligned}$$
- Eq. (5g) is the triangle inequality for the distance matrix d.
  Rewrite Eq. (5g) as:
  $$\begin{aligned} d_{u,v} &\leq d_{u,w} + d_{w,v} - (1 - \delta^w_{u,v}), && \forall u, v, w \in [n] \\ d_{u,v} &\geq d_{u,w} + d_{w,v} - 2n \cdot (1 - \delta^w_{u,v}), && \forall u, v, w \in [n] \end{aligned}$$
  where $2n$ is a big-M coefficient since $d_{u,w} + d_{w,v} \leq 2n$.

- Eq. (5h) initializes $\delta^w_{v,v}$ by definition, even node v does not exist.
- Eq. (5i) initializes $\delta^u_{u,v}$ and $\delta^v_{u,v}$ by definition, even node u or v does not exist.
- Eq. (5j) makes sure that there is at least on node at the shortest path from node u to v if there is no edge from node u and v and these two nodes both exist. Otherwise, only $\delta^u_{u,v}$ and $\delta^v_{u,v}$ equal to 1.

  Rewrite Eq. (5j) as:

$$\begin{aligned}
&\sum_{w\in[n]} \delta^w_{u,v} \le 2 + (n-2)\cdot(1-A_{u,v}), && \forall u,v\in[n],\ u\neq v\\
&\sum_{w\in[n]} \delta^w_{u,v} \le 2 + (n-2)\cdot A_{u,u}, && \forall u,v\in[n],\ u\neq v\\
&\sum_{w\in[n]} \delta^w_{u,v} \le 2 + (n-2)\cdot A_{v,v}, && \forall u,v\in[n],\ u\neq v\\
&\sum_{w\in[n]} \delta^w_{u,v} \ge A_{u,u} + A_{v,v} + (1-A_{u,v}), && \forall u,v\in[n],\ u\neq v
\end{aligned}$$

  where $n-2$ is a big-M coefficient since $\sum_{w\in[n]} \delta^w_{u,v} \le n$.

To conclude, the formulation for shortest paths of all connected graphs with at least n_0 nodes and at most n nodes is:

$$\begin{cases}
A_{v,v} \ge A_{v+1,v+1}, & \forall v\in[n-1]\\
\sum_{v\in[n]} A_{v,v} \ge n_0, &\\
2A_{u,v} \le A_{u,u} + A_{v,v}, & \forall u,v\in[n],\ u\neq v\\
d_{v,v} = 0, & \forall v\in[n]\\
d_{u,v} \le 1 + n\cdot(1-A_{u,v}), & \forall u,v\in[n],\ u\neq v\\
d_{u,v} \ge 2 - A_{u,v}, & \forall u,v\in[n],\ u\neq v\\
d_{u,v} \ge n\cdot(1-A_{u,u}), & \forall u,v\in[n],\ u\neq v\\
d_{u,v} \ge n\cdot(1-A_{v,v}), & \forall u,v\in[n],\ u\neq v\\
d_{u,v} \le d_{u,w} + d_{w,v} - (1-\delta^w_{u,v}), & \forall u,v,w\in[n]\\
d_{u,v} \ge d_{u,w} + d_{w,v} - 2n\cdot(1-\delta^w_{u,v}), & \forall u,v,w\in[n]\\
\delta^w_{v,v} = \begin{cases}1, & \text{if } w=v\\ 0, & \text{if } w\neq v\end{cases}, & \forall v,w\in[n]\\
\delta^u_{u,v} = \delta^v_{u,v} = 1, & \forall u,v\in[n],\ u\neq v\\
\sum_{w\in[n]} \delta^w_{u,v} \le 2 + (n-2)\cdot(1-A_{u,v}), & \forall u,v\in[n],\ u\neq v\\
\sum_{w\in[n]} \delta^w_{u,v} \le 2 + (n-2)\cdot A_{u,u}, & \forall u,v\in[n],\ u\neq v\\
\sum_{w\in[n]} \delta^w_{u,v} \le 2 + (n-2)\cdot A_{v,v}, & \forall u,v\in[n],\ u\neq v\\
\sum_{w\in[n]} \delta^w_{u,v} \ge A_{u,u} + A_{v,v} + (1-A_{u,v}), & \forall u,v\in[n],\ u\neq v
\end{cases} \tag{MIP-SP-plus}$$

Theorem A.3. *There is a bijective between the feasible domain of Eq. (MIP-SP-plus) with size* $[n_0, n]$ *and all connected graphs with number of nodes in* $[n_0, n]$.

Proof of Theorem A.3. Fix the node number as n_1 with $n_0 \le n_1 \le n$, Eqs. 5a – 5b force:

$$A_{v,v} = \begin{cases}1, & v\in[n_1]\\ 0, & v\in[n]\backslash[n_1]\end{cases}$$

substituting which to other constraints give us:

$$\begin{aligned}
&A_{u,v} = A_{v,u} = 0, && \forall u \in [n_1],\ v \in [n] \backslash [n_1],\ u \neq v \\
&d_{v,v} = 0, && \forall v \in [n] \backslash [n_1] \\
&d_{u,v} = d_{v,u} = n, && \forall u \in [n_1],\ v \in [n] \backslash [n_1],\ u \neq v \\
&\delta^w_{u,v} = \delta^w_{v,u} = \begin{cases} 1, & w \in \{u, v\} \\ 0, & w \notin \{u, v\} \end{cases}, && \forall u \in [n_1],\ v \in [n] \backslash [n_1]
\end{aligned}$$

One can easily check that all constraints associated with non-existed nodes are satisfied. Removing those constraints turns Eq. (MIP-SP-plus) into Eq. (MIP-SP) with size n_1. □

B ENCODING OF GRAPH KERNELS

For the encoding of different graph kernels, we first rewrite Eq. (3d) using Eq. (1) as:

$$k_{xX_i} = k(x, X_i) = \alpha \cdot k_G(G, G^i) + \beta \cdot k_F(F, F^i)$$

where $k_G(\cdot, \cdot)$ is formulated in Appendix B.1 and $k_F(\cdot, \cdot)$ is encoded in Appendix B.2. When only considering undirected graphs, those encoding could be further simplified as shown in Appendix B.3.

W.l.o.g., assume that each node has M features, i.e., $F^i \in \mathbb{R}^{n(G^i) \times M}$, and the first L features denote the one-hot encoding of its label, i.e., $\sum_{l \in [L]} F^i_l = 1,\ \forall 1 \leq i < t$.

B.1 GRAPH LEVEL

With the shortest distances $d_{u,v}$ as decision variables, formulating $k_G(G, G_i)$ is straightforward for SP and SSP kernels:

$$k_{SSP}(G, G^i) = \frac{1}{n^2 n_i^2} \sum_{u_1, v_1 \in [n]} \sum_{u_2, v_2 \in [n(G^i)]} d_{u_1, v_1}^{d_{u_2, v_2}(G^i)} = \frac{1}{n^2 n_i^2} \sum_{u,v,s \in [n]} D_s(G^i) \cdot d^s_{u,v}$$

where $n_i := n(G^i)$ is the number of nodes of G^i and $d^s_{u,v} = \mathbf{1}(d_{u,v} = s)$ are indicator variables of a big-M formulation:

$$\sum_{s \in [n+1]} d^s_{u,v} = 1, \ \sum_{s \in [n+1]} s \cdot d^s_{u,v} = d_{u,v},\ \forall u, v \in [n]$$

Remark B.1. We introduce $d^s_{u,v}$ for $s \in [n+1]$ instead of $s \in [n]$ to include the cases with unknown graph size, where $d_{u,v} = n$ means the shortest path from node u to v does not exist. But $d^n_{u,v}$ is not used in evaluating the kernel.

Similarly, introducing indicator variables $p^{s,l_1,l_2}_{u,v}$ as:

$$p^{s,l_1,l_2}_{u,v} = \mathbf{1}(F_{u,l_1} = 1, d_{u,v} = s, F_{v,l_2} = 1), \forall u, v, s \in [n],\ l_1, l_2 \in [L]$$

and counting the numbers of each type of paths in G^i:

$$P_{s,l_1,l_2}(G^i) = |\{(u, v) \mid u, v \in [n_i], l_u(G^i) = l_1, d_{u,v}(G^i) = s, l_v(G^i) = l_2\}|$$

the SP kernel is formulated as:

$$k_{SP}(G, G^i) = \frac{1}{n^2 n_i^2} \sum_{u,v,s \in [n], l_1, l_2 \in [L]} P_{s,l_1,l_2}(G^i) \cdot p^{s,l_1,l_2}_{u,v}$$

There are several ways to handle the exponential kernels: (i) directly use (local) nonlinear solvers, losing optimality guarantees, (ii) piecewise linearize the exponential function following Xie et al. (2024), or (iii) utilize nonlinear MIP functionalities in established solvers such as Gurobi (Gurobi Optimization, LLC, 2024) or SCIP (Vigerske & Gleixner, 2018). We choose to use Gurobi, which by default employs a dynamic piecewise linear approximation of the exponential function.

Note that K_{xx} in Eq. (3c) is not constant with a non-stationary kernel, making it the most complicated term in the whole formulation. By definition, $k_{SSP}(G, G)$ has the following quadratic form:

$$k_{SSP}(G, G) = \frac{1}{n^4} \sum_{s \in [n]} D_s^2$$

where $D_s = \sum_{u,v \in [n]} d_{u,v}^s,\ \forall s \in [n]$. Reusing the indicator trick and introducing $D_s^c = \mathbf{1}(D_s = c),\ \forall s \in [n], c \in [n^2 + 1]$, the quadratic form is equivalently linearized as:

$$K_{SSP}(G, G) = \frac{1}{n^4} \sum_{s \in [n], c \in [n^2+1]} c^2 \cdot D_s^c$$

where indicator variables should satisfy:

$$\sum_{c \in [n^2+1]} D_s^c = 1, \sum_{c \in [n^2+1]} c \cdot D_s^c = D_s,\ \forall s \in [n]$$

Repeating the procedure for SP kernel, we have:

$$K_{SP}(G, G) = \frac{1}{n^4} \sum_{s \in [n], l_1, l_2 \in [L], c \in [n^2+1]} c^2 \cdot P_{s,l_1,l_2}^c$$

where indicator variables $P_{s,l_1,l_2}^c = \mathbf{1}(P_{s,l_1,l_2} = c)$ satisfy:

$$\sum_{c \in [n^2+1]} P_{s,l_1,l_2}^c = 1, \sum_{c \in [n^2+1]} c \cdot P_{s,l_1,l_2}^c = P_{s,l_1,l_2},\ \forall s \in [n], l_1, l_2 \in [L]$$

B.2 FETURE LEVEL

Assume that each graph G has a binary feature matrix $F \in \{0, 1\}^{n(G) \times M}$, we need to formulate $k_F(F, F^i)$ and $k_F(F, F)$ properly. k_F could be defined in multiple ways, here we propose a permutational-invariant kernel considering the pair-wise similarity among node features. Given two feature matrices F^1, F^2 corresponding to graphs G^1, G^2 respectively, define k_F as:

$$k_F(F^1, F^2) := \frac{1}{n_1 n_2 M} \sum_{v_1 \in V^1, v_2 \in V^2} F_{v_1}^1 \cdot F_{v_2}^2 = \frac{1}{n_1 n_2 M} \sum_{m \in [M]} N_m(F^1) \cdot N_m(F^2)$$

where $N_m(F) = |\{v \mid v \in G,\ F_{v,m} = 1\}|,\ \forall m \in [M]$, $\frac{1}{n_1 n_2 M}$ is the normalized coefficient.

Similar to Appendix B.1, we have:

$$k_F(F, F^i) = \frac{1}{n n_i M} \sum_{m \in [M]} N_m(F^i) \cdot N_m$$

where $N_m = \sum_{v \in [n]} F_{v,m},\ \forall m \in [M]$, and

$$k_F(F, F) = \frac{1}{n^2 M} \sum_{m \in [M]} N_m^2 = \frac{1}{n^2 M} \sum_{m \in [M], c \in [M+1]} c^2 \cdot N_m^c$$

where indicators $N_m^c = \mathbf{1}(N_m = c),\ \forall m \in [M], c \in [M + 1]$ satisfy:

$$\sum_{c \in [M+1]} N_m^c = 1, \sum_{c \in [M+1]} c \cdot N_m^c = N_m,\ \forall m \in [M]$$

B.3 SIMPLIFY PATH ENCODING OVER UNDIRECTED GRAPHS

For undirected graphs, we first add the following constraints to guarantee symmetry:

$$\begin{cases} A_{u,v} = A_{v,u}, & \forall u, v \in [n],\ u < v \\ d_{u,v} = d_{v,u}, & \forall u, v \in [n],\ u < v \\ \delta_{u,v}^w = \delta_{v,u}^w, & \forall u, v, w \in [n],\ u < v \end{cases}$$

Since the inverse of any shortest path from node u to v is also a shortest path from node v to u, for SSP and ESSP kernels, $D_s,\ \forall s \in [n]$ are even and we can fix odd indicators as zero:

$$D_s^c = \begin{cases} 1, & \text{if } c \text{ is even} \\ 0, & \text{if } c \text{ is odd} \end{cases},\ \forall s \in [n],\ c \in [n^2+1]$$

Similarly, for SP and ESP kernels, we have:

$$P_{s,l_1,l_2} = P_{s,l_2,l_1},\ \forall s \in [n],\ f_1, f_2 \in [L]$$

C Additional numerical results

C.1 Kernel performance

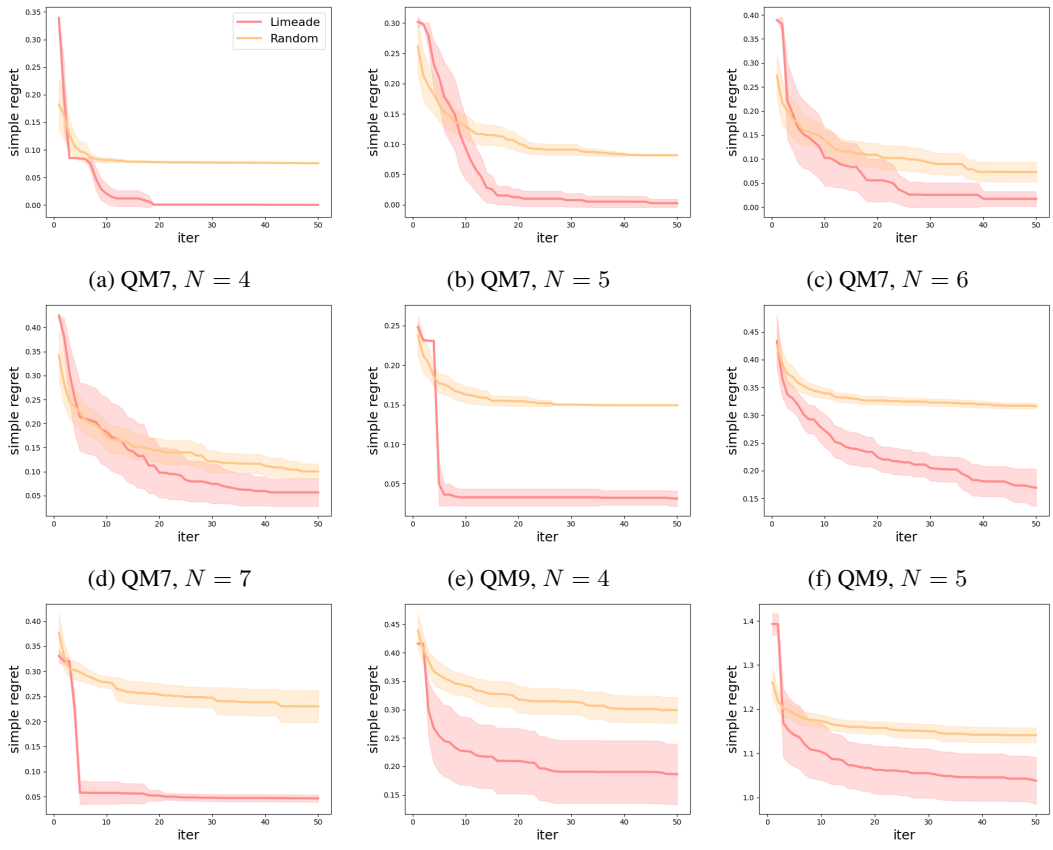
Table 2: Model performance of GPs equipped with different graph kernels. For each graph size N, we use Limeade to random generate 20 and 100 molecules for training and testing, respectively, root mean square error (RMSE) of predictive error is reported over 30 replications.

Dataset	N	SSP	SP	ESSP	ESP
QM7	10	0.30(0.08)	0.28(0.06)	0.29(0.08)	**0.26(0.07)**
	15	0.30(0.09)	**0.21(0.05)**	0.23(0.07)	**0.21(0.06)**
	20	0.32(0.14)	**0.22(0.08)**	0.26(0.08)	0.23(0.08)
	25	**0.19(0.08)**	**0.19(0.07)**	0.25(0.08)	0.22(0.08)
	30	0.28(0.19)	**0.26(0.15)**	0.34(0.19)	0.31(0.17)
QM9	10	1.20(0.47)	**0.67(0.13)**	0.85(0.24)	0.68(0.12)
	15	1.27(0.68)	0.45(0.16)	0.78(0.30)	**0.44(0.16)**
	20	1.41(0.69)	0.56(0.16)	0.82(0.35)	**0.55(0.15)**
	25	0.57(0.41)	**0.34(0.20)**	0.45(0.31)	0.35(0.21)
	30	0.28(0.19)	**0.20(0.19)**	0.25(0.26)	0.23(0.24)

Table 3: Model performance of GPs equipped with different graph kernels. For each graph size N , we use Limeade to random generate 20 and 100 molecules for training and testing, respectively, mean negative log likelihood (MNLL) is reported over 30 replications.

Dataset	N	SSP	SP	ESSP	ESP
QM7	10	6098.53(3475.83)	4.56(5.47)	0.59(0.57)	**0.19(0.46)**
	15	472.75(414.04)	1.06(1.71)	1.12(1.31)	**0.12(0.52)**
	20	440.53(418.42)	2.92(5.07)	2.47(3.37)	**0.27(0.60)**
	25	309.44(344.72)	10.06(16.12)	2.77(3.58)	**0.06(0.74)**
	30	581.49(621.67)	197.80(254.26)	1.64(2.33)	**0.87(1.71)**
QM9	10	116087.28(62728.04)	2.00(0.93)	2.58(1.05)	**1.23(0.42)**
	15	15756.72(18864.45)	1.26(0.98)	1.66(0.98)	**0.63(0.77)**
	20	7618.60(6860.77)	1.36(0.82)	2.45(2.13)	**0.97(0.42)**
	25	1065.96(1907.33)	1.15(2.92)	0.62(2.78)	**-0.59(1.91)**
	30	61.50(148.76)	-0.06(4.93)	-1.28(3.50)	**-1.88(2.82)**

All GPs are trained by maximizing the log marginal likelihood. There are two trainable parameters, i.e., α, β, for the SP and SSP kernels, and one extra variance σ_k^2 for the two exponential kernels. During GP training, we set bounds for kernel parameters to $[0.01, 100]$ with 1 as their initial values, and set noise variance σ_ϵ^2 as 10^{-6}. Table 2 and Table 3 reports RMSE and MNLL for GPs with different kernels and molecular size, respectively.



(a) QM7, $N = 4$ (b) QM7, $N = 5$ (c) QM7, $N = 6$

(d) QM7, $N = 7$ (e) QM9, $N = 4$ (f) QM9, $N = 5$

(g) QM9, $N = 6$ (h) QM9, $N = 7$ (i) QM9, $N = 8$

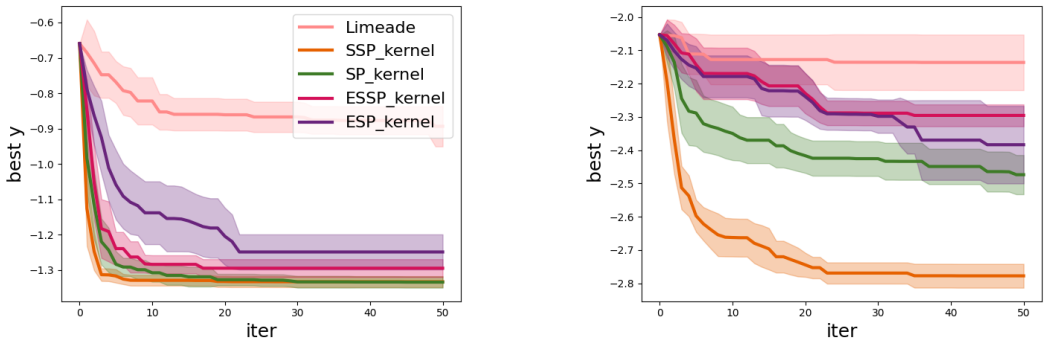
Figure 4: Performance of random sampling and Limeade over QM7 and QM9 datasets with different graph size N. Simple regret is plotted at each iteration. Mean with 0.5 standard deviation over 10 replications is reported.

C.2 Random sampling v.s. Limeade

Randomly sample feasible graphs is not trivial because the graph structure and features should be reasonable and compatible with each other, e.g., satisfying structural feasibility, dataset-specific constraints, etc. in molecular generation task. Here we consider random sampling over QM7 and QM9, to guarantee the feasibility of samples and compare it with Limeade. Figure 4 plots the regret curve over 50 iterations for both sample methods. In all cases, Limeade outperforms random sampling, showing the limitations of random sampling. Therefore, we choose Limeade as our sampling baseline.

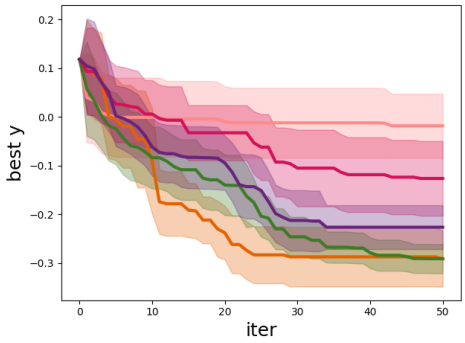
C.3 Additional optimal molecular design results

Results for $N \in \{15, 25\}$ are reported in Figure 5, supporting our analysis in Section 4.2.

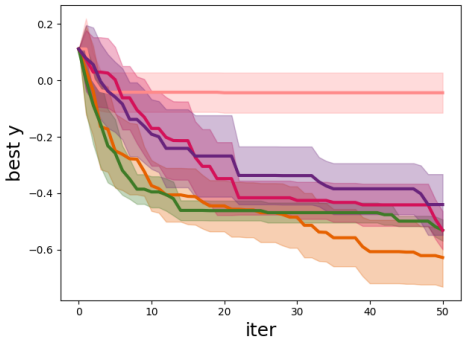


(a) QM7, $N = 15$

(b) QM7, $N = 25$



(c) QM9, $N = 15$



(d) QM9, $N = 25$

Figure 5: Bayesian optimization results on QM7 and QM9 with $N \in \{15, 25\}$. Best objective value is plotted at each iteration. Mean with 0.5 standard deviation over 10 replications is reported.