IMPERIAL

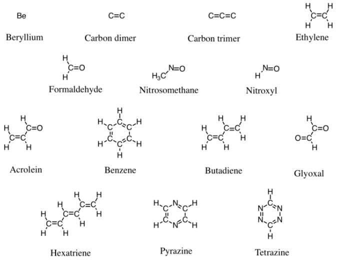
Global Acquisition Optimization for Structured Graph Bayesian Optimization

Jixiang Qing

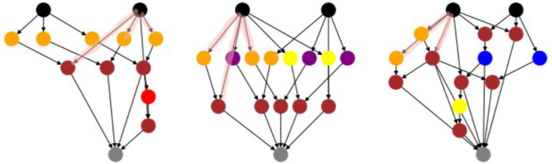
11/Sep/2025

Graph Representation and Optimization over Graphs

Graph representation is becoming popular in various domains:



(a) Sets of molecules [Loos et al., 2019]

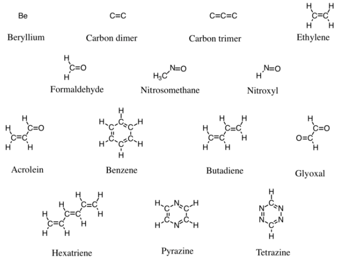


(b) Neural Architectures [Ru et al., 2020]

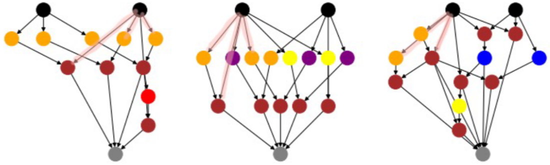
Figure: An illustration of graph structured data.

Graph Representation and Optimization over Graphs

Graph representation is becoming popular in various domains:



(a) Sets of molecules [Loos et al., 2019]



(b) Neural Architectures [Ru et al., 2020]

Figure: An illustration of graph structured data.

- **Graph Optimization**: Consider optimization over expensive black-box functions $f(G)$ defined over a space of graphs G differing in topology (structure, size) and attributes.

The issue of standard Bayesian Optimization for Graph Optimization

$$arg\ max_{\boldsymbol{x} \in \mathcal{X}}\ f(\boldsymbol{x})$$

Algorithm 1 Canonical BO Loop

Initialize dataset $\mathcal{D}_0 = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^{n_0}$
for $t = 1, \ldots N$ **do**
 Fit GP model $\mathcal{M}$ to $\mathcal{D}_{t-1}$

$$p(f\,|\,\boldsymbol{x}, D_{t-1}) = \mathcal{N}\left(\mu_t(\boldsymbol{x}), \sigma_t^2(\boldsymbol{x})\right)$$

 Maximize acquisition function $u(\cdot)$ to select the next promising point:

$$\boldsymbol{x}_t^* = \arg\max_{\boldsymbol{x} \in \mathcal{X}}\ u\left(\mu_t(\boldsymbol{x}), \sigma_t^2(\boldsymbol{x})\right)$$

 Query the true objective: $y_t = f(\boldsymbol{x}_t^*)$
 Update dataset: $\mathcal{D}_t = \mathcal{D}_{t-1} \cup \{(\boldsymbol{x}_t^*, y_t)\}$
end for

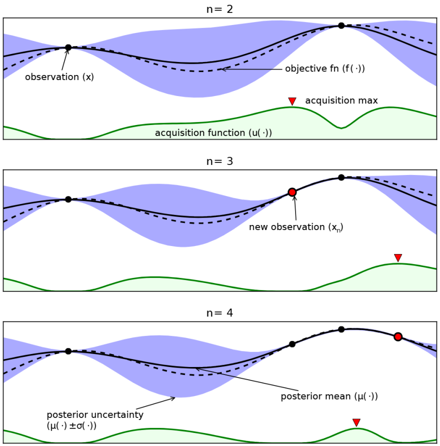


Figure: Illustration of Bayesian Optimization iterations (Figure from Shahriari et al. [2015])

The issue of standard Bayesian Optimization for Graph Optimization

$$arg\ max_{\boldsymbol{x} \in \mathcal{X}}\ f(\boldsymbol{x})$$

Algorithm 1 Canonical BO Loop

1: Initialize dataset $\mathcal{D}_0 = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^{n_0}$
2: **for** $t = 1, \ldots N$ **do**
3: Fit GP model $\mathcal{M}$ to $\mathcal{D}_{t-1}$

$$p(f|\boldsymbol{x}, D_{t-1}) = \mathcal{N}\left(\mu_t(\boldsymbol{x}), \sigma_t^2(\boldsymbol{x})\right)$$

4: Maximize acquisition function $u(\cdot)$ to select the next promising point:

$$\boldsymbol{x}_t^* = \arg\max_{\boldsymbol{x} \in \mathcal{X}} u\left(\mu_t(\boldsymbol{x}), \sigma_t^2(\boldsymbol{x})\right)$$

5: Query the true objective: $y_t = f(\boldsymbol{x}_t^*)$
6: Update dataset: $\mathcal{D}_t = \mathcal{D}_{t-1} \cup \{(\boldsymbol{x}_t^*, y_t)\}$
7: **end for**

Optimizing in Graph Space is Challenging

The issue of standard Bayesian Optimization for Graph Optimization

$$arg\ max_{\boldsymbol{x} \in \mathcal{X}}\ f(\boldsymbol{x})$$

Algorithm 1 Canonical BO Loop

Initialize dataset $\mathcal{D}_0 = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^{n_0}$
for $t = 1, \ldots N$ **do**
    Fit GP model $\mathcal{M}$ to $\mathcal{D}_{t-1}$

$$p(f \,|\, \boldsymbol{x}, D_{t-1}) = \mathcal{N}\left(\mu_t(\boldsymbol{x}), \sigma_t^2(\boldsymbol{x})\right)$$

    Maximize acquisition function $u(\cdot)$ to select the next promising point:

$$\boldsymbol{x}_t^* = \arg\max_{\boldsymbol{x} \in \mathcal{X}} u\left(\mu_t(\boldsymbol{x}), \sigma_t^2(\boldsymbol{x})\right)$$

    Query the true objective: $y_t = f(\boldsymbol{x}_t^*)$
    Update dataset: $\mathcal{D}_t = \mathcal{D}_{t-1} \cup \{(\boldsymbol{x}_t^*, y_t)\}$
end for

Optimizing in Graph Space is Challenging

- Enumerating is infeasible.
  - Search Space is (super) exponentially large
  - Random combination of variables does not admit a feasible graph

The issue of standard Bayesian Optimization for Graph Optimization

$$arg\ max_{\boldsymbol{x} \in \mathcal{X}}\ f(\boldsymbol{x})$$

Algorithm 1 Canonical BO Loop

1: Initialize dataset $\mathcal{D}_0 = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^{n_0}$
2: **for** $t = 1, \ldots N$ **do**
3: Fit GP model $\mathcal{M}$ to $\mathcal{D}_{t-1}$

$$p(f \,|\, \boldsymbol{x}, D_{t-1}) = \mathcal{N}\left(\mu_t(\boldsymbol{x}), \sigma_t^2(\boldsymbol{x})\right)$$

4: Maximize acquisition function $u(\cdot)$ to select the next promising point:

$$\boldsymbol{x}_t^* = \arg\max_{\boldsymbol{x} \in \mathcal{X}} u\left(\mu_t(\boldsymbol{x}), \sigma_t^2(\boldsymbol{x})\right)$$

5: Query the true objective: $y_t = f(\boldsymbol{x}_t^*)$
6: Update dataset: $\mathcal{D}_t = \mathcal{D}_{t-1} \cup \{(\boldsymbol{x}_t^*, y_t)\}$
7: **end for**

Optimizing in Graph Space is Challenging

- Enumerating is infeasible.
  - Search Space is (super) exponentially large
  - Random combination of variables does not admit a feasible graph
- Existing heuristics (e.g., evolutionary algorithm) does not have optimality guarantee.

The issue of standard Bayesian Optimization for Graph Optimization

$$arg\ max_{\boldsymbol{x} \in \mathcal{X}}\ f(\boldsymbol{x})$$

Algorithm 1 Canonical BO Loop

Initialize dataset $\mathcal{D}_0 = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^{n_0}$
for $t = 1, \ldots N$ **do**
    Fit GP model $\mathcal{M}$ to $\mathcal{D}_{t-1}$

$$p(f \mid \boldsymbol{x}, D_{t-1}) = \mathcal{N}\left(\mu_t(\boldsymbol{x}), \sigma_t^2(\boldsymbol{x})\right)$$

    Maximize acquisition function $u(\cdot)$ to select the next promising point:

$$\boldsymbol{x}_t^* = \arg\max_{\boldsymbol{x} \in \mathcal{X}} u\left(\mu_t(\boldsymbol{x}), \sigma_t^2(\boldsymbol{x})\right)$$

    Query the true objective: $y_t = f(\boldsymbol{x}_t^*)$
    Update dataset: $\mathcal{D}_t = \mathcal{D}_{t-1} \cup \{(\boldsymbol{x}_t^*, y_t)\}$
end for

Optimizing in Graph Space is Challenging

- Enumerating is infeasible.
  - Search Space is (super) exponentially large
  - Random combination of variables does not admit a feasible graph
- Existing heuristics (e.g., evolutionary algorithm) does not have optimality guarantee.
- Hard to handle structured constraints (e.g., only optimize for all connected graphs).

Global Acquisition Optimization for Structured Graph

$$arg\ max_{\boldsymbol{x} \in \mathcal{X}}\ f(\boldsymbol{x})$$

Algorithm 1 Canonical BO Loop

Initialize dataset $\mathcal{D}_0 = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^{n_0}$
for $t = 1, \ldots N$ **do**
  Fit GP model $\mathcal{M}$ to $\mathcal{D}_{t-1}$

$$p(f \,|\, \boldsymbol{x}, D_{t-1}) = \mathcal{N}\left(\mu_t(\boldsymbol{x}), \sigma_t^2(\boldsymbol{x})\right)$$

  Maximize acquisition function $u(\cdot)$ to select the next promising point:

$$\boldsymbol{x}_t^* = \arg\max_{\boldsymbol{x} \in \mathcal{X}} u\left(\mu_t(\boldsymbol{x}), \sigma_t^2(\boldsymbol{x})\right)$$

  Query the true objective: $y_t = f(\boldsymbol{x}_t^*)$
  Update dataset: $\mathcal{D}_t = \mathcal{D}_{t-1} \cup \{(\boldsymbol{x}_t^*, y_t)\}$
end for

This Talk

Based on the shortest-path kernel [Borgwardt and Kriegel, 2005], we develop a **graph encoding** that enables **Mathematical Programming** for optimization over graph spaces, that:

Global Acquisition Optimization for Structured Graph

$$arg\ max_{\boldsymbol{x} \in \mathcal{X}}\ f(\boldsymbol{x})$$

Algorithm 1 Canonical BO Loop

```
Initialize dataset 𝒟_0 = {(x_i, y_i)}_{i=1}^{n_0}
for t = 1, ... N do
    Fit GP model ℳ to 𝒟_{t-1}
        p(f | x, D_{t-1}) = 𝒩(μ_t(x), σ_t^2(x))
    Maximize acquisition function u(·) to select the next promising point:
        x_t^* = arg max_{x ∈ 𝒳} u(μ_t(x), σ_t^2(x))
    Query the true objective: y_t = f(x_t^*)
    Update dataset: 𝒟_t = 𝒟_{t-1} ∪ {(x_t^*, y_t)}
end for
```

Initialize dataset $\mathcal{D}_0 = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^{n_0}$

for $t = 1, \ldots N$ **do**

Fit GP model $\mathcal{M}$ to $\mathcal{D}_{t-1}$

$$p(f \,|\, \boldsymbol{x}, D_{t-1}) = \mathcal{N}\left(\mu_t(\boldsymbol{x}), \sigma_t^2(\boldsymbol{x})\right)$$

Maximize acquisition function $u(\cdot)$ to select the next promising point:

$$\boldsymbol{x}_t^* = \arg\max_{\boldsymbol{x} \in \mathcal{X}} u\left(\mu_t(\boldsymbol{x}), \sigma_t^2(\boldsymbol{x})\right)$$

Query the true objective: $y_t = f(\boldsymbol{x}_t^*)$

Update dataset: $\mathcal{D}_t = \mathcal{D}_{t-1} \cup \{(\boldsymbol{x}_t^*, y_t)\}$

end for

This Talk

Based on the shortest-path kernel [Borgwardt and Kriegel, 2005], we develop a **graph encoding** that enables **Mathematical Programming** for optimization over graph spaces, that:

- Can handle different graph structures.

Global Acquisition Optimization for Structured Graph

$$arg\ max_{\boldsymbol{x} \in \mathcal{X}}\ f(\boldsymbol{x})$$

Algorithm 1 Canonical BO Loop

1: Initialize dataset $\mathcal{D}_0 = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^{n_0}$
2: **for** $t = 1, \ldots N$ **do**
3: Fit GP model $\mathcal{M}$ to $\mathcal{D}_{t-1}$

$$p(f \,|\, \boldsymbol{x}, D_{t-1}) = \mathcal{N}\left(\mu_t(\boldsymbol{x}), \sigma_t^2(\boldsymbol{x})\right)$$

4: Maximize acquisition function $u(\cdot)$ to select the next promising point:

$$\boldsymbol{x}_t^* = \arg\max_{\boldsymbol{x} \in \mathcal{X}} u\left(\mu_t(\boldsymbol{x}), \sigma_t^2(\boldsymbol{x})\right)$$

5: Query the true objective: $y_t = f(\boldsymbol{x}_t^*)$
6: Update dataset: $\mathcal{D}_t = \mathcal{D}_{t-1} \cup \{(\boldsymbol{x}_t^*, y_t)\}$
7: **end for**

This Talk

Based on the shortest-path kernel [Borgwardt and Kriegel, 2005], we develop a **graph encoding** that enables **Mathematical Programming** for optimization over graph spaces, that:

- Can handle different graph structures.
- Guarantees global optimality of acquisition optimization.

Global Acquisition Optimization for Structured Graph

$$arg\ max_{\boldsymbol{x}\in\mathcal{X}}\ f(\boldsymbol{x})$$

Algorithm 1 Canonical BO Loop

Initialize dataset $\mathcal{D}_0 = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^{n_0}$
for $t = 1, \ldots N$ **do**
Fit GP model $\mathcal{M}$ to $\mathcal{D}_{t-1}$

$$p(f\,|\,\boldsymbol{x}, D_{t-1}) = \mathcal{N}\left(\mu_t(\boldsymbol{x}), \sigma_t^2(\boldsymbol{x})\right)$$

Maximize acquisition function $u(\cdot)$ to select the next promising point:

$$\boxed{\boldsymbol{x}_t^* = \arg\max_{\boldsymbol{x}\in\mathcal{X}} u\left(\mu_t(\boldsymbol{x}), \sigma_t^2(\boldsymbol{x})\right)}$$

Query the true objective: $y_t = f(\boldsymbol{x}_t^*)$
Update dataset: $\mathcal{D}_t = \mathcal{D}_{t-1} \cup \{(\boldsymbol{x}_t^*, y_t)\}$
end for

This Talk

Based on the shortest-path kernel [Borgwardt and Kriegel, 2005], we develop a **graph encoding** that enables **Mathematical Programming** for optimization over graph spaces, that:

- Can handle different graph structures.
- Guarantees global optimality of acquisition optimization.
- Can be used for molecular design and neural architecture search.

Outline

- Graph Concepts and Gaussian Process (GP) on Graphs

Outline

- Graph Concepts and Gaussian Process (GP) on Graphs
- Mixed Integer Programming and Graph Encoding

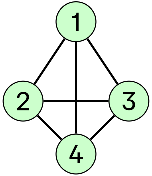
Outline

- Graph Concepts and Gaussian Process (GP) on Graphs
- Mixed Integer Programming and Graph Encoding
- Empirical Results

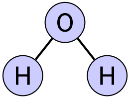
Preliminaries

Graphs

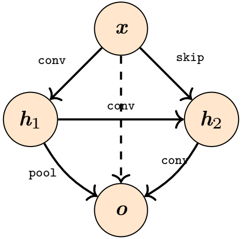
What is a graph? A mathematical structure with nodes (vertices) and edges (connections)



Complete Graph
All nodes connected



(Undirected) Connected Graph
Path between any pair



DAG
Directed, no cycles

Applications:
- Molecules
- Social networks

Applications:
- Neural nets
- Workflows

Graph Concepts and Notations

Graph Definitions

- **Graph:** $G = (V, E)$ where:
  - V = set of nodes
  - $E \subseteq V \times V$ = edges
- **Adjacency:** $A \in \{0, 1\}^{n \times n}$
  - $A_{uv} = 1 \Leftrightarrow (u, v) \in E$

Graph Concepts and Notations

Graph Definitions

- **Graph:** $G = (V, E)$ where:
  - V = set of nodes
  - $E \subseteq V \times V$ = edges
- **Adjacency:** $A \in \{0, 1\}^{n \times n}$
  - $A_{uv} = 1 \Leftrightarrow (u, v) \in E$

Key Notation

- **Node exists:** $A_{vv} = 1 \Leftrightarrow v \in V$

Graph Concepts and Notations

Graph Definitions

- **Graph:** $G = (V, E)$ where:
  - V = set of nodes
  - $E \subseteq V \times V$ = edges
- **Adjacency:** $A \in \{0, 1\}^{n \times n}$
  - $A_{uv} = 1 \Leftrightarrow (u, v) \in E$

Key Notation

- **Node exists:** $A_{vv} = 1 \Leftrightarrow v \in V$
- **Distance:** $d_{u,v}$ = shortest path length

Graph Concepts and Notations

Graph Definitions

- **Graph:** $G = (V, E)$ where:
  - V = set of nodes
  - $E \subseteq V \times V$ = edges
- **Adjacency:** $A \in \{0, 1\}^{n \times n}$
  - $A_{uv} = 1 \Leftrightarrow (u, v) \in E$

Key Notation

- **Node exists:** $A_{vv} = 1 \Leftrightarrow v \in V$
- **Distance:** $d_{u,v}$ = shortest path length
- **Path indicator:** $\delta_{uv}^{w} = 1$ if w on path

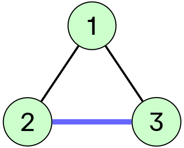
Graph Concepts and Notations

Graph Definitions

- **Graph:** $G = (V, E)$ where:
  - V = set of nodes
  - $E \subseteq V \times V$ = edges
- **Adjacency:** $A \in \{0,1\}^{n \times n}$
  - $A_{uv} = 1 \Leftrightarrow (u, v) \in E$

Key Notation

- **Node exists:** $A_{vv} = 1 \Leftrightarrow v \in V$
- **Distance:** $d_{u,v}$ = shortest path length
- **Path indicator:** $\delta_{uv}^{w} = 1$ if w on path
- **Path info:** $e_{u,v} = (d_{u,v}, l_u, l_v)$



Graph 1 (Cycle)

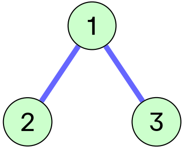
$V^1 = \{1,2,3\}$

$E^1 = \{(1,2),(1,3),(2,3)\}$

$d_{2,3} = 1$

$$A^1 = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

$\delta_{2,3}^1 = 0$



Graph 2 (Tree)

$V^2 = \{1,2,3\}$

$E^2 = \{(1,2),(1,3)\}$

$d_{2,3} = 2$

$$A^2 = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix}$$

$\delta_{2,3}^1 = 1$

Gaussian Processes over Graphs and Shortest-Path Kernel

Gaussian Process provide posterior predictive distribution as:

$$\mu(x_*) = k(x_*, X)\left[K(X,X) + \sigma_n^2 I\right]^{-1} y, \sigma^2(x_*) = k(x_*, x_*) - k(x_*, X)\left[K(X,X) + \sigma_n^2 I\right]^{-1} k(X, x_*)$$

Shortest-Path (SP) Kernel[Borgwardt & Kriegel, 2005] Compare all shortest paths between all node pairs in two graphs

General form:

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

Gaussian Processes over Graphs and Shortest-Path Kernel

Gaussian Process provide posterior predictive distribution as:

$$\mu(x_*) = k(x_*, X)\left[K(X, X) + \sigma_n^2 I\right]^{-1} y, \sigma^2(x_*) = k(x_*, x_*) - k(x_*, X)\left[K(X, X) + \sigma_n^2 I\right]^{-1} k(X, x_*)$$

Shortest-Path (SP) Kernel[Borgwardt & Kriegel, 2005] Compare all shortest paths between all node pairs in two graphs

General form:

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

Path comparison:

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

Gaussian Processes over Graphs and Shortest-Path Kernel

Gaussian Process provide posterior predictive distribution as:

$$\mu(x_*) = k(x_*, X)\left[K(X,X) + \sigma_n^2 I\right]^{-1} y, \sigma^2(x_*) = k(x_*, x_*) - k(x_*, X)\left[K(X,X) + \sigma_n^2 I\right]^{-1} k(X, x_*)$$

Shortest-Path (SP) Kernel[Borgwardt & Kriegel, 2005] Compare all shortest paths between all node pairs in two graphs

General form:

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

Path comparison:

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

With (Normalized) Dirac kernels (exact matching):

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

SP Kernel Variants

Handling Complex Graphs

Attributed Graphs: $X = (G, F)$

- G: Graph structure + labels
- F: Node features (continuous)

Composite kernel:

$$k(X^1, X^2) = \underbrace{\alpha k_G(G^1, G^2)}_{\text{structure}} + \underbrace{\beta k_F(F^1, F^2)}_{\text{features}}$$

SP Kernel Variants

Handling Complex Graphs

Attributed Graphs: $X = (G, F)$

- G: Graph structure + labels
- F: Node features (continuous)

Composite kernel:

$$k(X^1, X^2) = \underbrace{\alpha k_G(G^1, G^2)}_{\text{structure}} + \underbrace{\beta k_F(F^1, F^2)}_{\text{features}}$$

Simplified SP (SSP) Kernel

Drops label requirements:

$$k_{SSP} = \frac{1}{n_1^2 n_2^2} \sum_{u_1, v_1, u_2, v_2} \mathbb{1}\{d_{u_1, v_1} = d_{u_2, v_2}\}$$

✓ Less sparse

SP Kernel Variants

Handling Complex Graphs

Attributed Graphs: $X = (G, F)$

- G: Graph structure + labels
- F: Node features (continuous)

Composite kernel:

$$k(X^1, X^2) = \underbrace{\alpha k_G(G^1, G^2)}_{\text{structure}} + \underbrace{\beta k_F(F^1, F^2)}_{\text{features}}$$

Nonlinear Extensions

Exponential variants:

$$k_{ESP}(G^1, G^2) = \exp\left(\frac{k_{SP}(G^1, G^2)}{\sigma_k^2}\right)$$

$$k_{ESSP}(G^1, G^2) = \exp\left(\frac{k_{SSP}(G^1, G^2)}{\sigma_k^2}\right)$$

✓ More expressive × Harder to optimize

Simplified SP (SSP) Kernel

Drops label requirements:

$$k_{SSP} = \frac{1}{n_1^2 n_2^2} \sum_{u_1, v_1, u_2, v_2} \mathbb{1}\{d_{u_1, v_1} = d_{u_2, v_2}\}$$

✓ Less sparse

SP Kernel Variants

Handling Complex Graphs

Attributed Graphs: $X = (G, F)$

- G: Graph structure + labels
- F: Node features (continuous)

Composite kernel:

$$k(X^1, X^2) = \underbrace{\alpha k_G(G^1, G^2)}_{\text{structure}} + \underbrace{\beta k_F(F^1, F^2)}_{\text{features}}$$

Nonlinear Extensions

Exponential variants:

$$k_{ESP}(G^1, G^2) = \exp\left(\frac{k_{SP}(G^1, G^2)}{\sigma_k^2}\right)$$

$$k_{ESSP}(G^1, G^2) = \exp\left(\frac{k_{SSP}(G^1, G^2)}{\sigma_k^2}\right)$$

✓ More expressive × Harder to optimize

Simplified SP (SSP) Kernel

Drops label requirements:

$$k_{SSP} = \frac{1}{n_1^2 n_2^2} \sum_{u_1, v_1, u_2, v_2} \mathbb{1}\{d_{u_1, v_1} = d_{u_2, v_2}\}$$

✓ Less sparse

Summary of Variants

Kernel	Labels in k_G?	Nonlinear?
SP	Yes	No
SSP	No	No
ESP	Yes	Yes
ESSP	No	Yes

Acquisition Maximization in Graph Spaces: The Optimizers

Mixed Integer Programming with Auxiliary Variables

MIP Standard Form

$$\min_{x,z} c^T x + d^T z \quad \text{s.t.} \quad Ax + Bz \leq b, \quad Ex + Fz = g, \quad x \in \mathbb{R}^n, \quad z \in \{0,1\}^m$$

Mixed Integer Programming with Auxiliary Variables

MIP Standard Form

$$\min_{x,z} c^T x + d^T z \quad \text{s.t.} \quad Ax + Bz \leq b, \quad Ex + Fz = g, \quad x \in \mathbb{R}^n, \quad z \in \{0,1\}^m$$

Example: Optimizing $f(x, z) = x^2 - 2\sqrt{z}$ where $x \in \{0, 1, 2, 3\}$, $z \in [0, 4]$ Using MIP

MIP Formulation:

$$\min_{x,z,y,w,\lambda_i} \quad y - 2w \qquad \text{(linear objective via auxiliaries)}$$

$$\text{s.t.} \quad x = \sum_{i=0}^{3} i \cdot \lambda_i, \quad \sum_{i=0}^{3} \lambda_i = 1$$

$$y = x^2 \qquad \text{(encoded via binary indicators)}$$

$$w \geq 0 \qquad \text{by definition of } \sqrt{z}$$

$$w^2 \leq z \qquad \text{(relaxed to linear inequalities)}$$

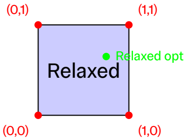
$$\lambda_i \in \{0, 1\}, \quad z \in [0, 4]$$

How MIP Solvers Work: Branch-and-Bound

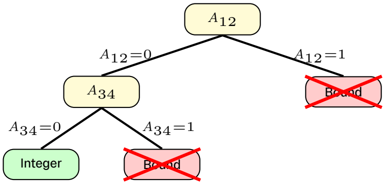
Core Idea: Relaxation Provides Bounds

Original: $\min f(A)$ where $A_{ij} \in \{0, 1\}$

Relaxed: $\min f(A)$ where $A_{ij} \in [0, 1]$



Relaxed solution gives **lower bound**



Branch-and-Bound Algorithm

1. **Relax:** Allow $A_{ij} \in [0, 1]$
   - Solve LP (polynomial time)
   - Get lower bound on optimum
2. **Branch:** If $A_{ij} = 0.5$:
   - Left: Fix $A_{ij} = 0$
   - Right: Fix $A_{ij} = 1$
3. **Prune:** Cut branch if:
   - Bound $\geq$ best integer found
   - Infeasible subproblem
4. **Repeat:** Until all branches explored
   - Select next node (heuristic)
   - Continue branching

Efficiency: Early pruning via tight bounds
Complexity: Best $\mathcal{O}(N)$, Worst $\mathcal{O}(2^N)$
Modern solvers: Cuts, heuristics, parallelization

From Graph BO to Mixed Integer Programming

The Challenge

Goal: Optimize UCB acquisition function $u(G) = \mu(G) + \beta\sigma(G)$ over graphs

- **Search space:** Graph topology + node/edge labels (binary $+$ categorical variable)
- **Objective:** GP acquisition function (continuous, nonlinear)

From Graph BO to Mixed Integer Programming

The Challenge

Goal: Optimize UCB acquisition function $u(G) = \mu(G) + \beta\sigma(G)$ over graphs

- **Search space:** Graph topology + node/edge labels (binary $+$ categorical variable)
- **Objective:** GP acquisition function (continuous, nonlinear)

Our MIP Formulation [Xie et al., 2024]

$$\max_{G,\mu,\sigma} \quad \mu + \beta\sigma \qquad \text{(acquisition function)} \tag{1}$$

$$\text{s.t.} \quad \mu = K_{GX}K_{XX}^{-1}y \qquad \text{(GP mean)} \tag{2}$$

$$\sigma^2 \leq K_{GG} - K_{GX}K_{XX}^{-1}K_{XG} \qquad \text{(GP variance)} \tag{3}$$

$$G \in \mathcal{G}_{\text{specified}} \qquad \text{(graph constraints)} \tag{4}$$

where:

- Blue variables: auxiliary (continuous) variables
- Black variables: are decisions (discrete)

Challenge

⚠️ Solving the optimization is not as simple as:

```
solve_MIP(acq_func, x_space, z_space)
```

Challenge

⚠ Solving the optimization is not as simple as:

```
solve_MIP(acq_func, x_space, z_space)
```

Valid graphs are a tiny fraction of all adjacency matrices

- Arbitrary A does not
  - define a valid graph.
  - define a graph in the space of our interest (e.g., $\mathcal{G}_{connected}, \mathcal{G}_{DAG}$)

Challenge

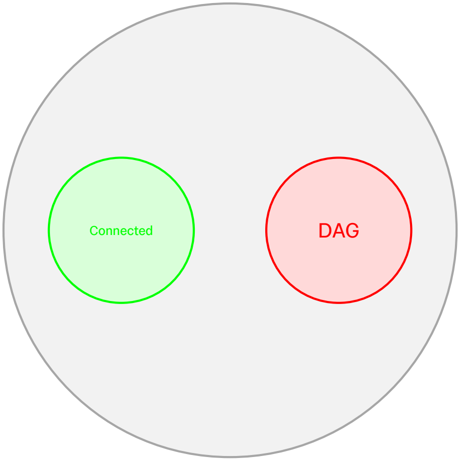
⚠️ Solving the optimization is not as simple as:

```
solve_MIP(acq_func, x_space, z_space)
```

Valid graphs are a tiny fraction of all adjacency matrices

- Arbitrary A does not
  - define a valid graph.
  - define a graph in the space of our interest (e.g., $\mathcal{G}_{connected}$, $\mathcal{G}_{DAG}$)

⚠️ Need explicit constraints to stay in valid graph space

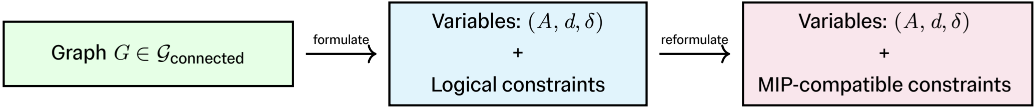


All matrices $\{0, 1\}^{n \times n}$

Graph Encoding:
Enabling MIP for Graph-Structured Optimization

Graph Encoding: From Graphs to MIP Variables

Graph Encoding: Represent graph properties as MIP variables and constraints



Graph Encoding: From Graphs to MIP Variables

Graph Encoding: Represent graph properties as MIP variables and constraints

Graph $G \in \mathcal{G}_{\text{connected}}$ —formulate→ Variables: (A, d, δ) + Logical constraints —reformulate→ Variables: (A, d, δ) + MIP-compatible constraints

Properties We Must Encode

Basic Structure	For Shortest Path	Graph Type Constraints
▪ Edge existence: $A_{uv} \in \{0, 1\}$ ▪ Node presence: $A_{vv} \in \{0, 1\}$ ▪ Var size: $A_{uv} \leq min\{A_{uu}, A_{vv}\}$	▪ Shortest distance: $d_{u,v} \in [0, n+1]$ ▪ Path indicator: $\delta^w_{uv} \in \{0, 1\}$ ▪ Triangle: $d_{u,v} \leq d_{uw} + d_{wv}$ ▪ If $A_{uv} = 1$ then $d_{u,v} = 1$ ▪ If $\delta^w_{uv} = 1$ then $d_{u,v} = d_{uw} + d_{wv}$	▪ Connectivity: $d_{u,v} < n$ ▪ Undirected: $A_{uv} = A_{vu}$ ▪ DAG: $d_{u,v} + d_{vu} \geq n$ (no cycles)

MIP Reformulation

Logical Constraint	MIP Reformulation (via Big-M)
$A_{uv} = 1 \Rightarrow d_{u,v} = 1$	$d_{u,v} \leq 1 + n(1 - A_{uv})$ $d_{u,v} \geq 1 - n(1 - A_{uv})$
$A_{uv} = 0 \Rightarrow d_{u,v} \geq 2$	$d_{u,v} \geq 2 - A_{uv}$
$\delta_{uv}^{w} = 1 \Rightarrow d_{u,v} = d_{uw} + d_{wv}$	$d_{u,v} \leq d_{uw} + d_{wv} + n(1 - \delta_{uv}^{w})$ $d_{u,v} \geq d_{uw} + d_{wv} - n(1 - \delta_{uv}^{w})$

MIP Reformulation

Logical Constraint	MIP Reformulation (via Big-M)
$A_{uv} = 1 \Rightarrow d_{u,v} = 1$	$d_{u,v} \leq 1 + n(1 - A_{uv})$ $d_{u,v} \geq 1 - n(1 - A_{uv})$
$A_{uv} = 0 \Rightarrow d_{u,v} \geq 2$	$d_{u,v} \geq 2 - A_{uv}$
$\delta_{uv}^{w} = 1 \Rightarrow d_{u,v} = d_{uw} + d_{wv}$	$d_{u,v} \leq d_{uw} + d_{wv} + n(1 - \delta_{uv}^{w})$ $d_{u,v} \geq d_{uw} + d_{wv} - n(1 - \delta_{uv}^{w})$

- Inactivate constraint when binary variable $= 0$, otherwise constraint becomes tight,
- The reformulation is not unique, in appropriate reformulation result change properties (e.g., bijectiveness).

Theoretical Guarantee: Bijection Property

Theorem (Bijection between MIP solutions and connected graphs [Xie et al., 2025a])

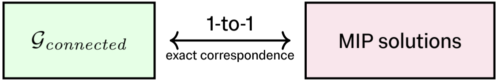
For any feasible solution (A, d, δ) of our MIP formulation with n nodes, there exists a unique connected graph G with the same (A, d, δ), and vice versa.

Theoretical Guarantee: Bijection Property

Theorem (Bijection between MIP solutions and connected graphs [Xie et al., 2025a])

For any feasible solution (A, d, δ) of our MIP formulation with n nodes, there exists a unique connected graph G with the same (A, d, δ), and vice versa.

What This Means



Implications

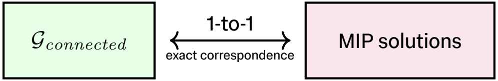
- **No missing graphs:** Every connected graph can be found
- **No invalid solutions:** Every MIP solution is a real graph
- **Global optimality:** MIP provably finds the best graph

Theoretical Guarantee: Bijection Property

Theorem (Bijection between MIP solutions and connected graphs [Xie et al., 2025a])

For any feasible solution (A, d, δ) of our MIP formulation with n nodes, there exists a unique connected graph G with the same (A, d, δ), and vice versa.

What This Means



Implications

- **No missing graphs:** Every connected graph can be found
- **No invalid solutions:** Every MIP solution is a real graph
- **Global optimality:** MIP provably finds the best graph

Implications

- **Variable-size graphs**: Same bijection holds [Xie et al., 2025a]
- **DAGs**: Extended with acyclicity constraints [Xie et al., 2025b]

Acquisition Function Maximization in Graph Space: The Final Formulation

Our Complete MIP Formulation

$$\max_{G,\mu,\sigma} \quad \mu + \beta\sigma \qquad \text{(acquisition function)} \tag{5}$$

$$\text{s.t.} \quad \mu = K_{GX} K_{XX}^{-1} y \qquad \text{(GP mean)} \tag{6}$$

$$\sigma^2 \leq K_{GG} - K_{GX} K_{XX}^{-1} K_{XG} \qquad \text{(GP variance)} \tag{7}$$

$$G \in \mathcal{G}_{\text{connected}} \qquad \text{(graph constraints)} \tag{8}$$

Acquisition Function Maximization in Graph Space: The Final Formulation

Our Complete MIP Formulation

$$\max_{G,\mu,\sigma} \quad \mu + \beta\sigma \qquad \text{(acquisition function)} \tag{5}$$

$$\text{s.t.} \quad \mu = K_{GX}K_{XX}^{-1}y \qquad \text{(GP mean)} \tag{6}$$

$$\sigma^2 \leq K_{GG} - K_{GX}K_{XX}^{-1}K_{XG} \qquad \text{(GP variance)} \tag{7}$$

$$G \in \mathcal{G}_{\text{connected}} \qquad \text{(graph constraints)} \tag{8}$$

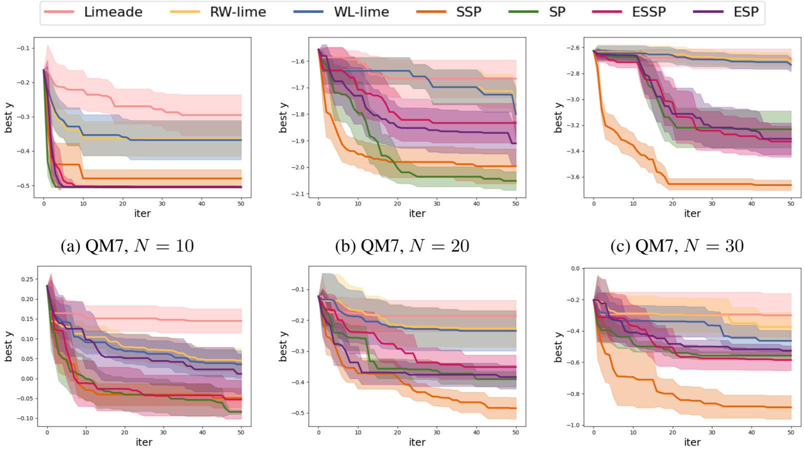
How This Becomes MIP-Solvable

Component	**MIP Implementation**
Graph G	Variables (A, d, δ) with linear constraints
$G \in \mathcal{G}_{\text{connected}}$	Logical constraints linearized via Big-M
Kernel K_{GX}	Function of $(d_{u,v})$ - linearized [Xie et al., 2024]
GP computations μ, σ	Auxiliary continuous variables [Xie et al., 2024]
Products like $K_{GX}K_{XX}^{-1}$	McCormick envelopes [Xie et al., 2024]

Result: Thousands of linear constraints + binary/continuous variables → Solved by branch-and-bound (Gurobi) → Global optimal graph

Empirical Investigation

Molecular Optimization

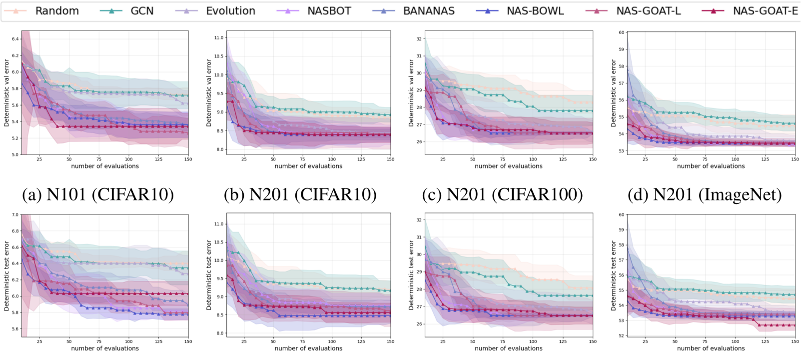


(a) QM7, $N = 10$ (b) QM7, $N = 20$ (c) QM7, $N = 30$

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

Figure: Bayesian optimization results on QM7 and QM9.

Neural Architecture Search Results



(a) N101 (CIFAR10) (b) N201 (CIFAR10) (c) N201 (CIFAR100) (d) N201 (ImageNet)

(e) N101 (CIFAR10) (f) N201 (CIFAR10) (g) N201 (CIFAR100) (h) N201 (ImageNet)

Figure: Numerical results of Graph BO on NAS-Bench-101 (N101) ($N \leq 7$) and NAS-Bench-201 (N201) ($N = 4$). (**Top**) Deterministic validation error. (**Bottom**) The corresponding test error. Median with one standard deviation over 20 replications is plotted.

Takeaway

We enable exact Acquisition Function Optimization over graph spaces via MIP

Takeaway

We enable exact Acquisition Function Optimization over graph spaces via MIP

Key Contributions

- **Graph encoding:** First MIP formulation for connected graphs and DAGs

Takeaway

We enable exact Acquisition Function Optimization over graph spaces via MIP

Key Contributions

- **Graph encoding:** First MIP formulation for connected graphs and DAGs
- **Theoretical guarantee:** Proved bijection between MIP solutions and graphs

Takeaway

We enable exact Acquisition Function Optimization over graph spaces via MIP

Key Contributions

- **Graph encoding:** First MIP formulation for connected graphs and DAGs
- **Theoretical guarantee:** Proved bijection between MIP solutions and graphs

Takeaway

We enable exact Acquisition Function Optimization over graph spaces via MIP

Key Contributions

- **Graph encoding:** First MIP formulation for connected graphs and DAGs
- **Theoretical guarantee:** Proved bijection between MIP solutions and graphs
- **Empirical validation:** State-of-the-art performance on
  - Molecular design (connected graphs)
  - Neural architecture search (DAGs)

Practical Impact

Able to conduct small to medium scale $N \leq 30$ (with acquisition optimization taking 1-10 minutes per iteration) graph BO with connected graph or DAG, supporting discrete edge feature and node label.

References

K. M. Borgwardt and H.-P. Kriegel. Shortest-path kernels on graphs. In Fifth IEEE international conference on data mining (ICDM'05), pages 8–pp. IEEE, 2005.

P.-F. Loos, M. Boggio-Pasqua, A. Scemama, M. Caffarel, and D. Jacquemin. Reference energies for double excitations. Journal of chemical theory and computation, 15(3):1939–1956, 2019.

B. Ru, X. Wan, X. Dong, and M. Osborne. Interpretable neural architecture search via bayesian optimisation with weisfeiler-lehman kernels. arXiv preprint arXiv:2006.07556, 2020.

B. Shahriari, K. Swersky, Z. Wang, R. P. Adams, and N. De Freitas. Taking the human out of the loop: A review of bayesian optimization. Proceedings of the IEEE, 104(1):148–175, 2015.

Y. Xie, S. Zhang, J. Paulson, and C. Tsay. Global optimization of gaussian process acquisition functions using a piecewise-linear kernel approximation. arXiv preprint arXiv:2410.16893, 2024.

Y. Xie, S. Zhang, J. Qing, R. Misener, and C. Tsay. Bograpе: Bayesian optimization over graphs with shortest-path encoded. arXiv preprint arXiv:2503.05642, 2025a.

Y. Xie, S. Zhang, J. Qing, R. Misener, and C. Tsay. Global optimization of graph acquisition functions for neural architecture search. arXiv preprint arXiv:2505.23640, 2025b.

IMPERIAL

Thank you!
Questions?

Contact: j.qing@imperial.ac.uk

[1] Xie, Y., Zhang, S., Qing, J., Misener, R., & Tsay, C. (2025). BoGrape: Bayesian Optimization over Graphs with Shortest-Path Encoded. arXiv:2503.05642

[2] Xie, Y., Zhang, S., Qing, J., Misener, R., & Tsay, C. (2025). Global optimization of graph acquisition functions for neural architecture search. arXiv:2505.23640