

Bayesian Optimisation of Functions over Node-subsets on Graph



Machine Learning Research Group
Department of Engineering Science
University of Oxford

Research Problem

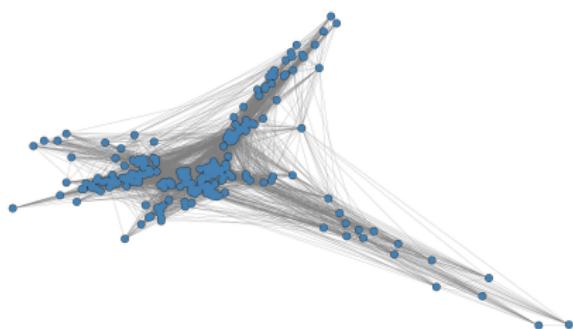
In network analysis, we are often interested in finding a subset \mathcal{S} of k nodes that leads to the maximisation of a black-box utility function $f(\mathcal{S})$ on a generic graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$ over the combinatorial space $\binom{\mathcal{V}}{k}$.

In the case of maximisation, it can be expressed as:

$$\mathcal{S}^* = \arg \max_{\mathcal{S} \in \binom{\mathcal{V}}{k}} f(\mathcal{S}). \quad (1)$$

Real-world examples

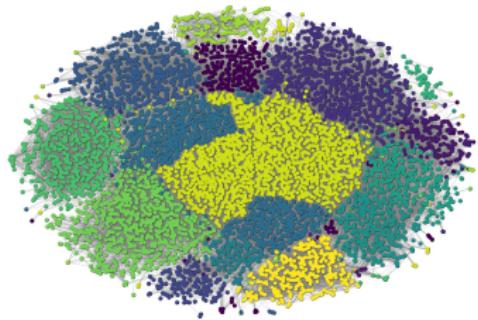
Proximity-contact network



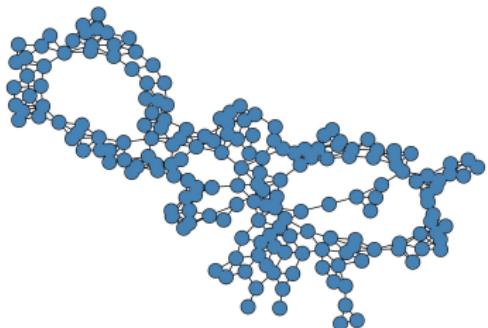
London road network



Social network



Molecule from DD-TUDataset



To optimise this utility function, however, is a non-trivial task.
Challenges include:

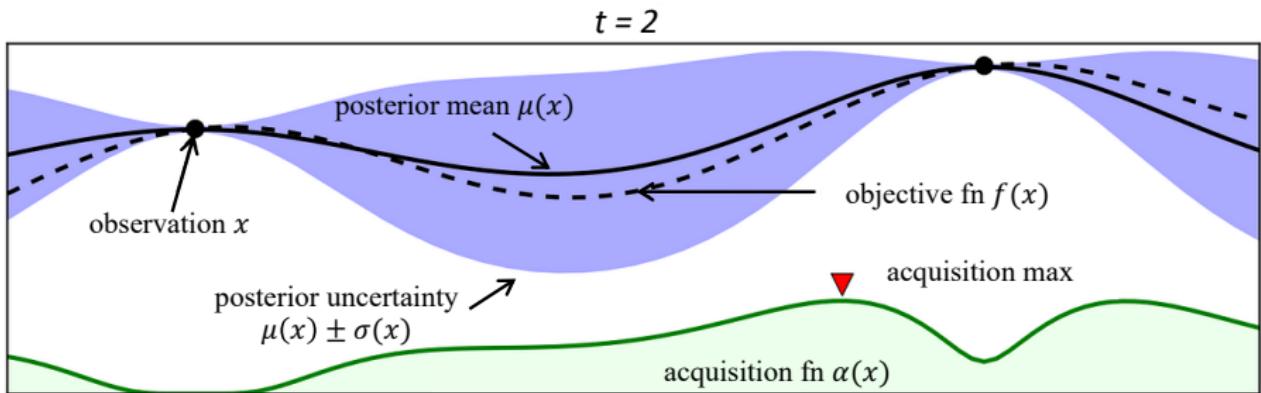
- ▶ Most conventional optimisation algorithms are designed for continuous space.
- ▶ The combinatorial search space $\binom{\mathcal{V}}{k}$ is massive and possesses an underlying graph structure.
- ▶ The original graph is not even fully observable in certain scenarios (e.g. offline social networks or contact networks).
- ▶ The objective functions are usually black-box and expensive to evaluate in many applications.

Preliminaries

BO is a gradient-free optimisation algorithm that aims to find the global optimal point x^* of a back-box function $f : \mathcal{X} \rightarrow \mathbb{R}$ over the search space \mathcal{X} , which, in the case of maximisation, can be written as $x^* = \arg \max_{x \in \mathcal{X}} f(x)$.

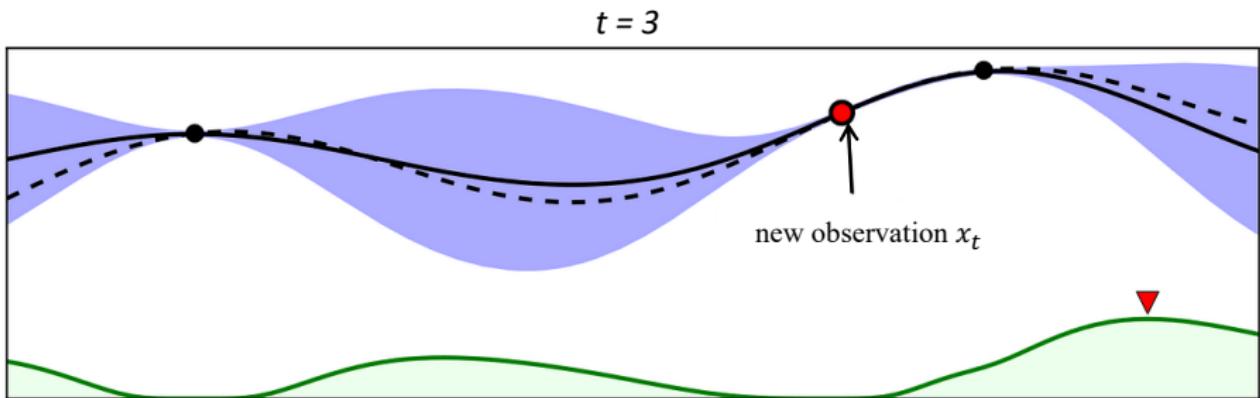
The framework consists of two steps at each iteration:

- ▶ Fitting a surrogate model.
- ▶ Maximizing an acquisition function.



Gaussian Processes surrogate:

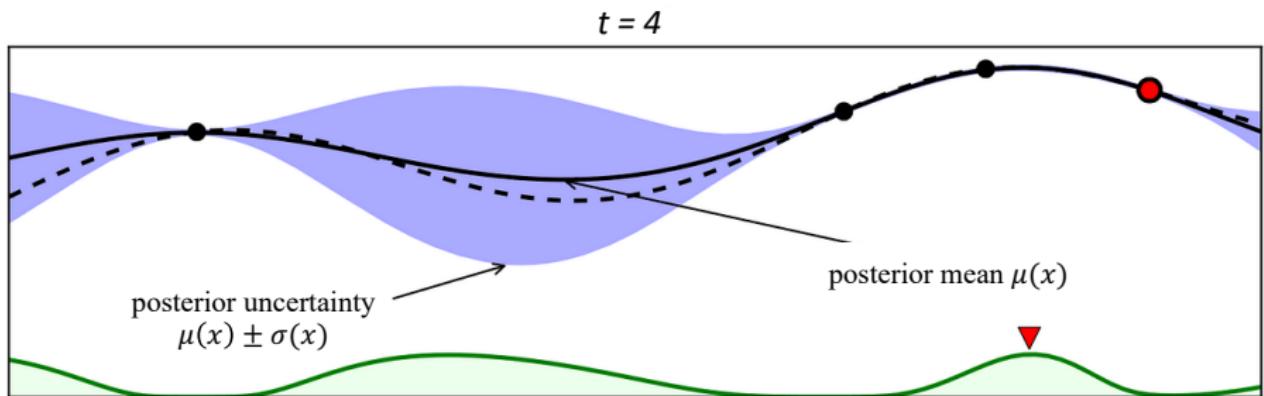
$$f(x) \sim \mathcal{GP}(\mu(x), k(x, x')).$$



Expected Improvement acquisition:

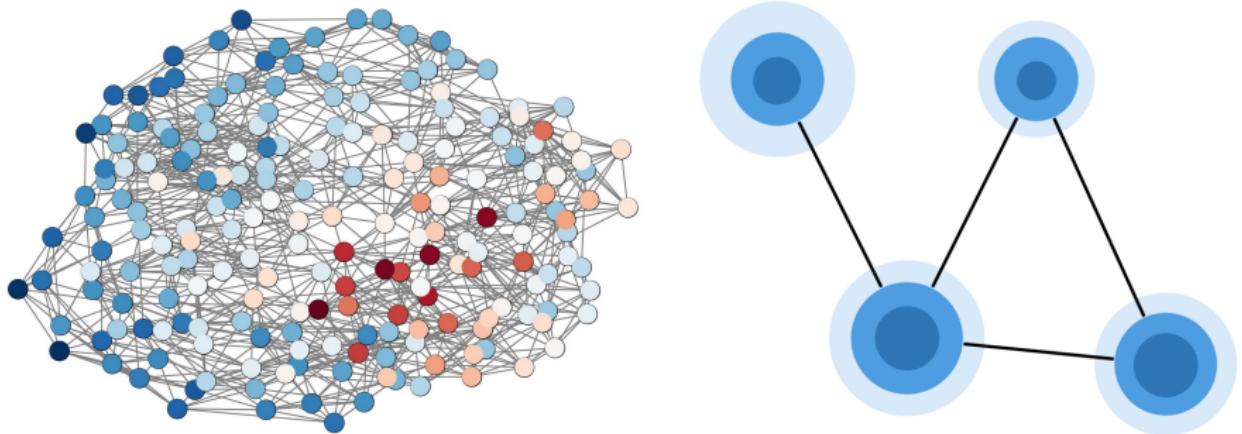
$$\text{EI}_{1:t}(x') = \mathbb{E}[(f(x') - f(x_{1:t}^*))^+],$$

with $[\alpha]^+ = \max(\alpha, 0)$ and $x_t^* = \arg \max_{x_i \in x_{1:t}} f(x_i)$.



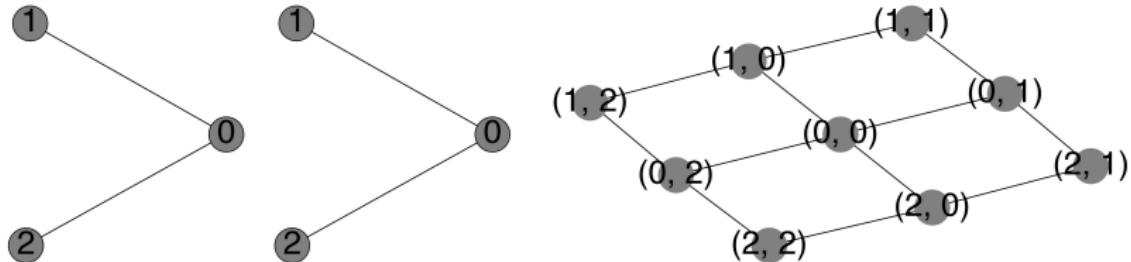
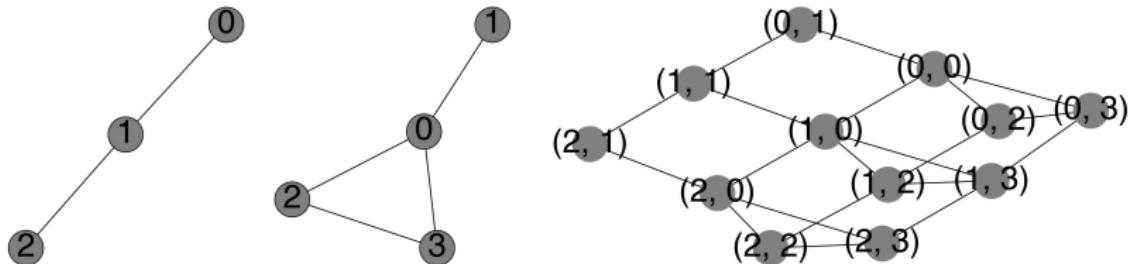
Previous work (Wan et al., NeurIPS 2023) investigated BO of functions on a single node [1], which requires Gaussian Processes defined on a graph as the surrogate model.

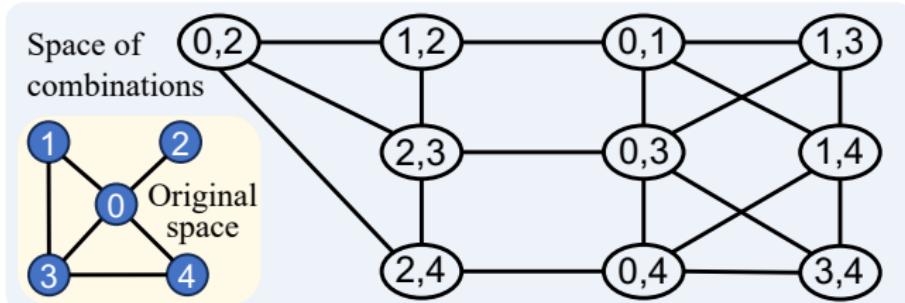
$$f(v) \sim \mathcal{GP}(\mu(v), k(v, v')).$$



Methodology

Examples of a Cartesian product of two graphs:

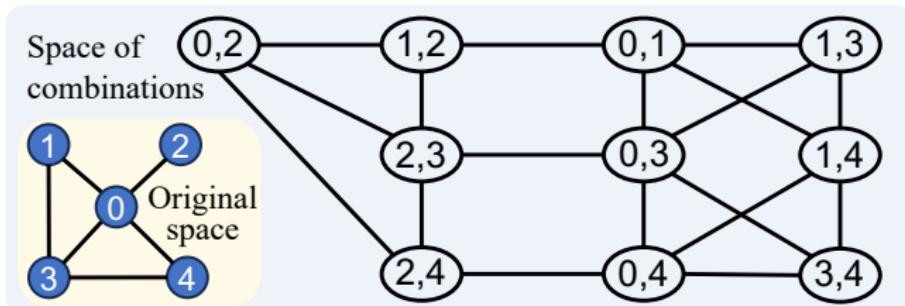




The combinatorial operation for combinations of k nodes on an underlying graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is given by:

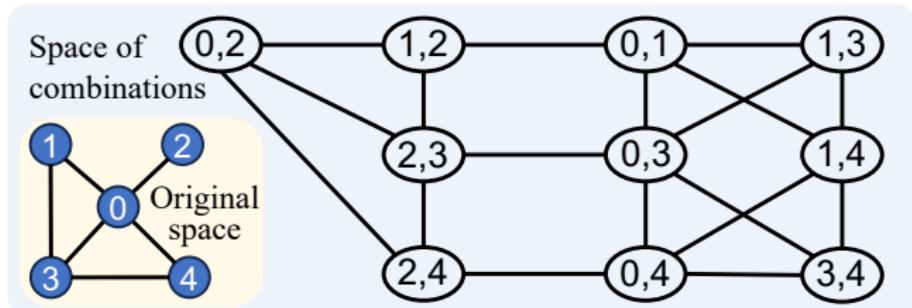
$$\hat{\mathcal{G}}^{ k } = \square_{i=1}^k \mathcal{G}, \quad (2)$$

which leads to a combo-graph $\hat{\mathcal{G}}^{ k } = \{\hat{\mathcal{V}}, \hat{\mathcal{E}}\}$ of size $|\hat{\mathcal{V}}| = \binom{|\mathcal{V}|}{k}$ with each combo-node $\hat{v}_i = (v_i^{(1)}, v_i^{(2)}, \dots, v_i^{(k)}) \in \hat{\mathcal{V}}$ being a combination of k nodes from graph \mathcal{G} without replacement.



The combo-edges $\hat{\mathcal{E}}$ in the combo-graph are defined as follows:

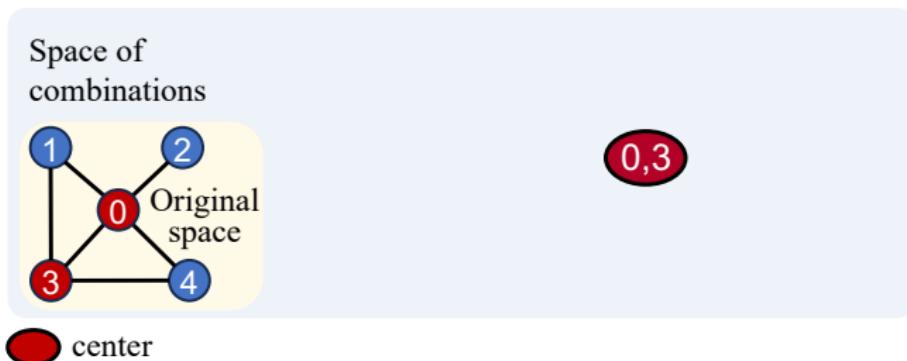
Assume $\hat{v}_1 = (v_1^{(1)}, v_1^{(2)}, \dots, v_1^{(k)})$ and $\hat{v}_2 = (v_2^{(1)}, v_2^{(2)}, \dots, v_2^{(k)})$ are two arbitrary combo-nodes in the combo-graph $\mathcal{G}^{ k }$, then the combo-edge $(\hat{v}_1, \hat{v}_2) \in \hat{\mathcal{E}}$ iff $\exists j$ such that $\forall i \neq j$, $v_1^{(i)} = v_2^{(i)}$ and $(v_1^{(j)}, v_2^{(j)}) \in \mathcal{E}$.



Note when $k = 1$, the problem reduces to the previous work [1] that uses BO for functions of a single node.

However, when k and $|\mathcal{V}|$ are relatively large, the combinatorial space could be huge (e.g. $\binom{1000}{8} \approx 2.4 \times 10^{19}$).

Local modelling is needed!



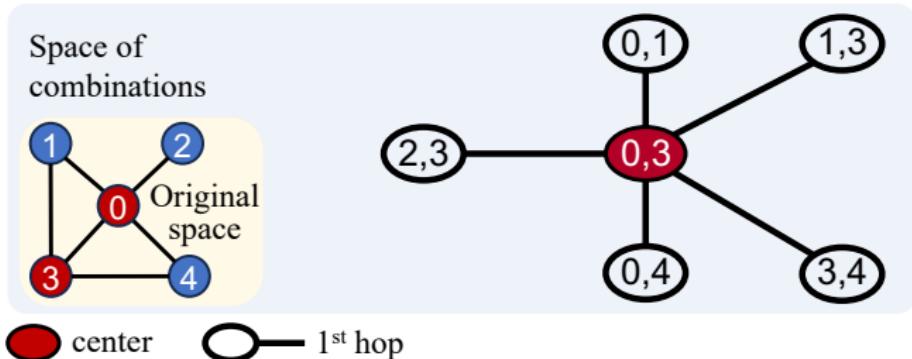
Goal: Construct an ego-subgraph at a given centre \hat{v}^* of size Q with maximum hop ℓ_{\max} from the underlying graph \mathcal{G} .

Algorithm:

1. Initialise $\tilde{\mathcal{G}} = \{\tilde{\mathcal{V}}, \tilde{\mathcal{E}}\}$ with only \hat{v}^* and then loop through neighbours of each element node $v \in \hat{v}^*$.

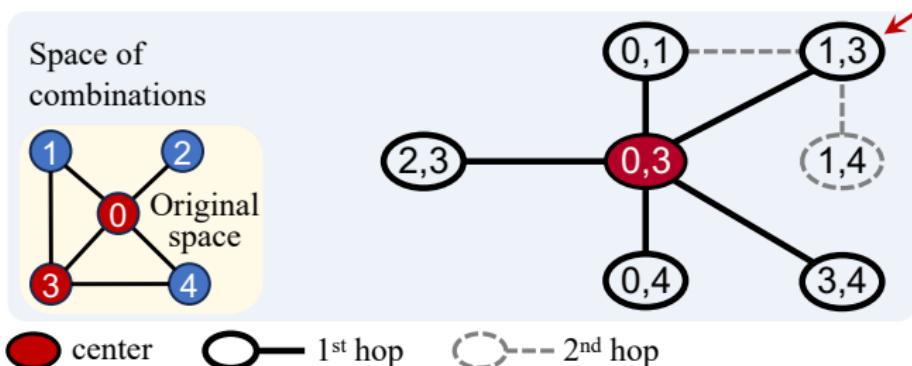


2. If a neighbour $\mathcal{N}_i(v)$ of v is not in \hat{v}^* , create a new combo-node \hat{v}' by substituting $\mathcal{N}_i(v)$ with v in \hat{v}^* , and accordingly create a combo-edge by connecting \hat{v}' to \hat{v}^* .

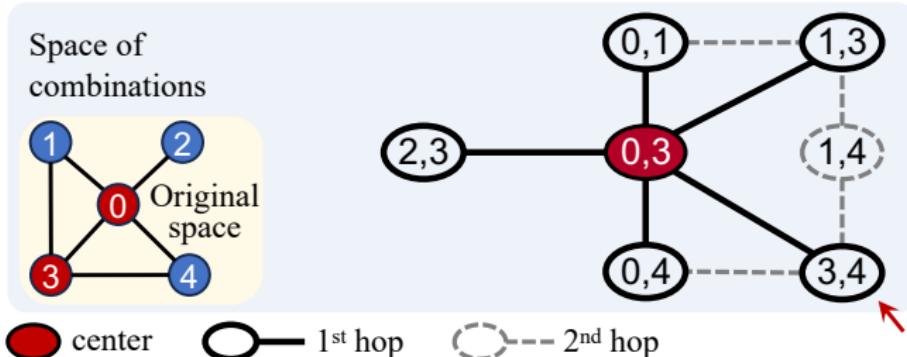


2. If a neighbour $\mathcal{N}_i(v)$ of v is not in \hat{v}^* , create a new combo-node \hat{v}' by substituting $\mathcal{N}_i(v)$ with v in \hat{v}^* , and accordingly create a combo-edge by connecting \hat{v}' to \hat{v}^* .

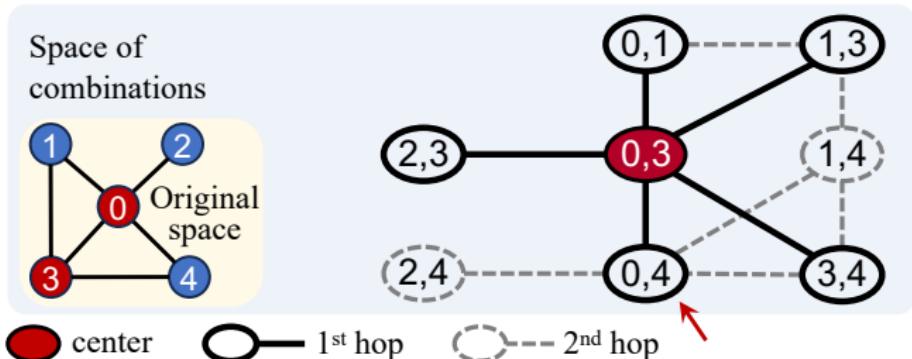
Remark: After finding the combo-neighbours of \hat{v}^* at hop $\ell = 1$, $\tilde{\mathcal{G}}$ becomes a star-network at centre \hat{v}^* .



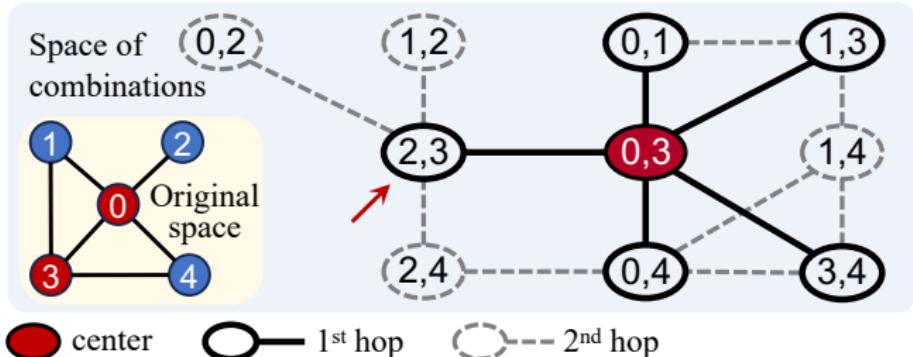
3. Repeat the above star-sampling procedures for every newly found combo-node to find their combo-neighbours at hop $\ell + 1$, which meanwhile also finds the edges among combo-nodes within hop ℓ .



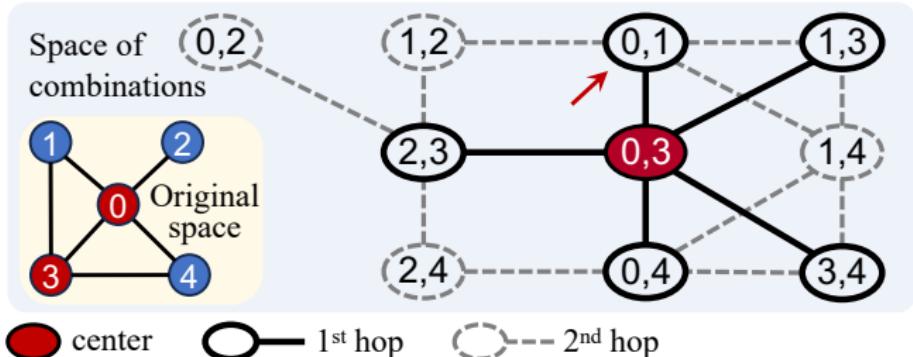
3. Repeat the above star-sampling procedures for every newly found combo-node to find their combo-neighbours at hop $\ell + 1$, which meanwhile also finds the edges among combo-nodes within hop ℓ .



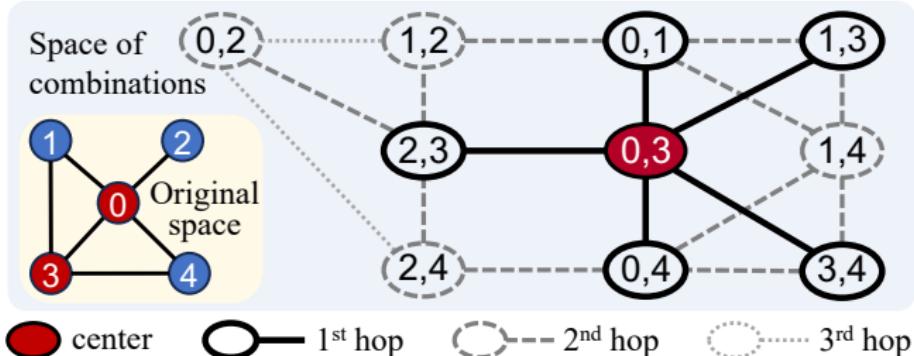
3. Repeat the above star-sampling procedures for every newly found combo-node to find their combo-neighbours at hop $\ell + 1$, which meanwhile also finds the edges among combo-nodes within hop ℓ .



3. Repeat the above star-sampling procedures for every newly found combo-node to find their combo-neighbours at hop $\ell + 1$, which meanwhile also finds the edges among combo-nodes within hop ℓ .



3. Repeat the above star-sampling procedures for every newly found combo-node to find their combo-neighbours at hop $\ell + 1$, which meanwhile also finds the edges among combo-nodes within hop ℓ .

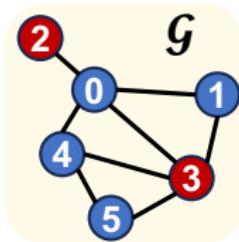
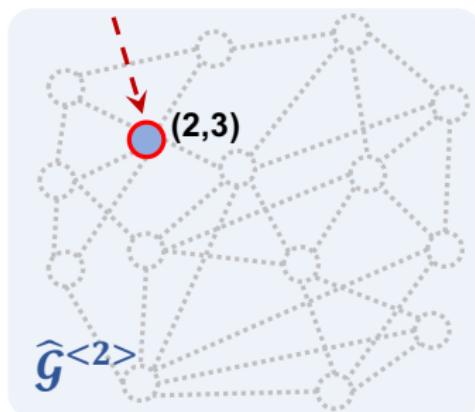


4. Stop the algorithm until the subgraph size limit Q or the maximum hop ℓ_{\max} is reached.

(t-1)

Initial combo-node
 \hat{v}_{t-1}^* from restart or
 last query

Combinatorial
 Space

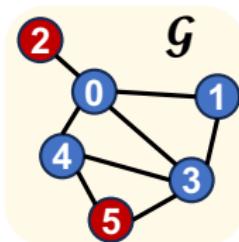
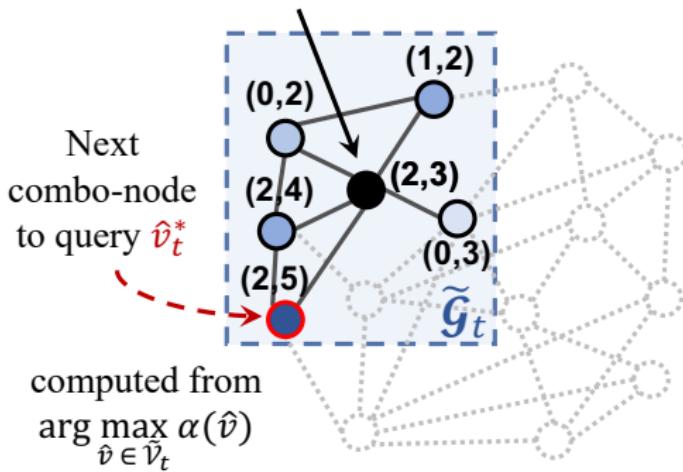


Original Space

- unexplored structure
- explored structure
- queried combo-node
- surrogate prediction
- next query

(t)

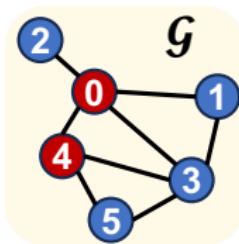
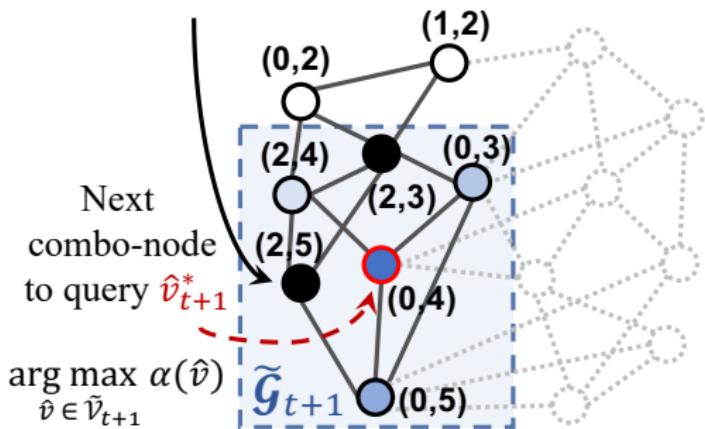
Context combo-subgraph \tilde{g}_t at iter t,
centered at last queried combo-node \hat{v}_{t-1}^*



- unexplored structure ○— explored structure ● queried combo-node
- surrogate prediction ● next query

(t+1)

$\tilde{\mathcal{G}}_{t+1}$ centered at last
queried combo-node \hat{v}_t^*

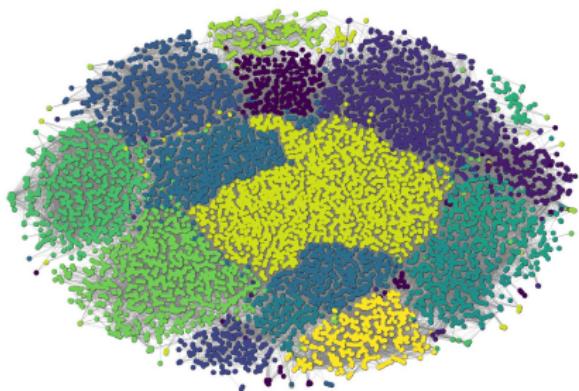


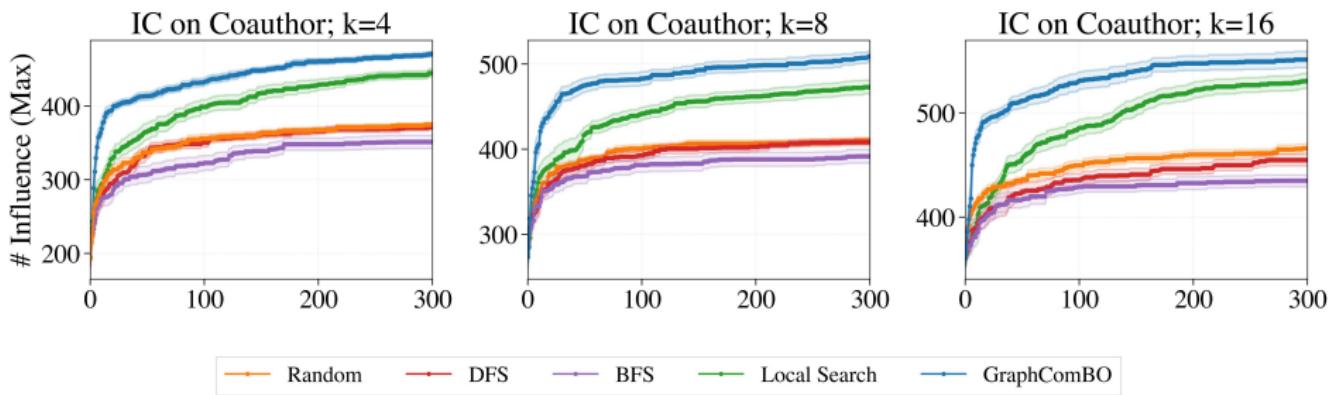
Original Space

- unexplored structure ○— explored structure ● queried combo-node
- surrogate prediction ● next query

Real-world Experiments

- ▶ Each node can be either activated or inactivated at each step.
- ▶ Initializing with only k seed nodes activated.
- ▶ When node v becomes active, it is given a single chance to activate each inactive neighbour w with probability p .
- ▶ The process runs until no more activation is possible.





Thank You 😊

- [1] Xingchen Wan, Pierre Osselin, Henry Kenlay, Binxin Ru, Michael A Osborne, and Xiaowen Dong. Bayesian optimisation of functions on graphs. *Advances in Neural Information Processing Systems*, 2023.