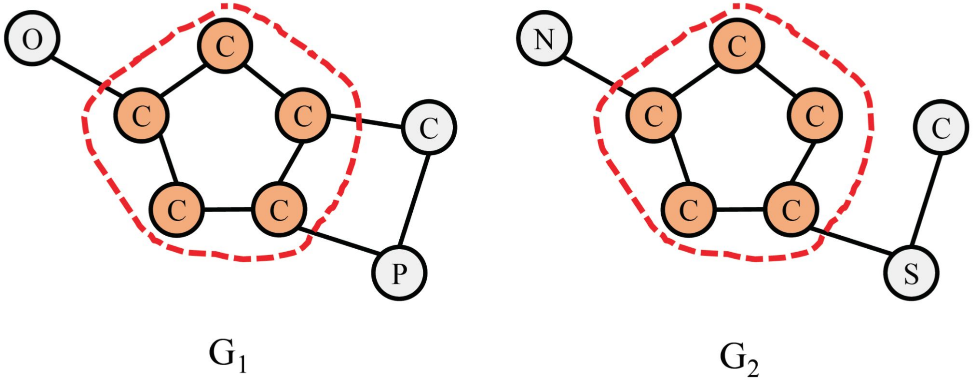
GLSearch: Maximum Common Subgraph Detection via Learning to Search

Derek Xu*,Yunsheng Bai*, Yizhou Sun, Wei Wang
University of California Los Angeles, California, USA

Presenter: Derek Xu
06/18/2021

Maximum Common Subgraph (MCS) Detection

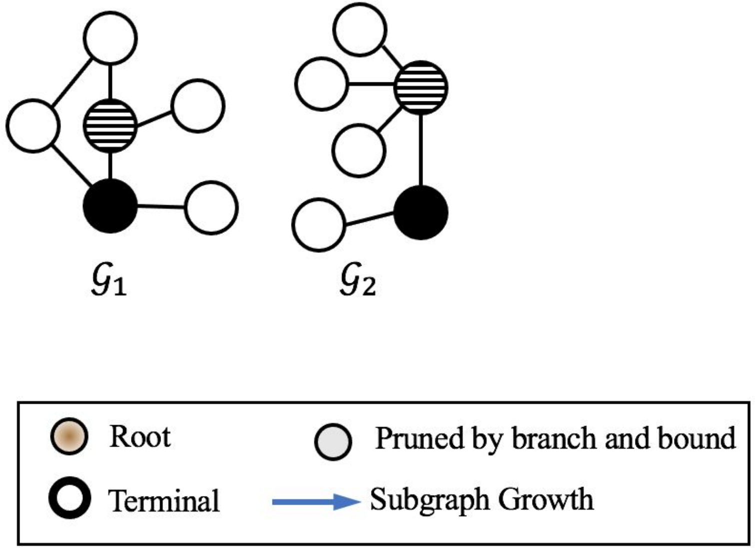
1. Applications:
   a. Software analysis
   b. Graph database
   c. Cloud computing
   d. Drug synthesis



2. Challenging (NP-hard)
   a. Subgraphs must be isomorphic to each other
   b. Found subgraphs should be as large as possible
   c. Connected and induced subgraph

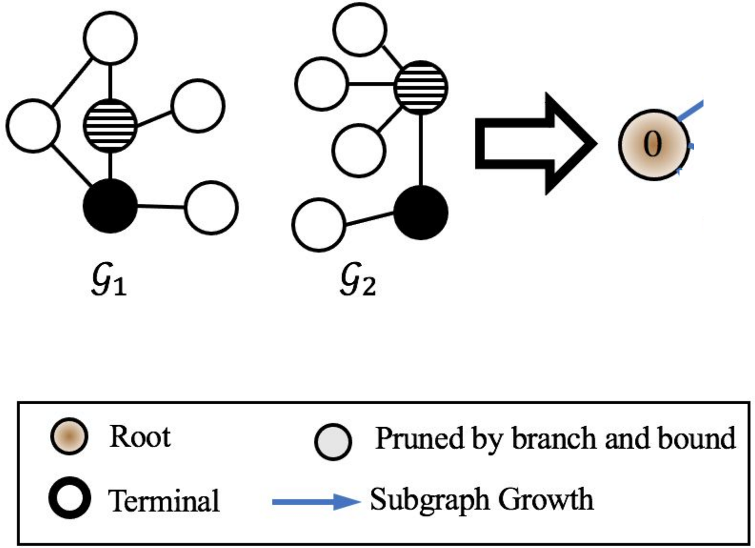
Existing Work on MCS Detection

State-of-the-art branch and bound search algorithm for MCS



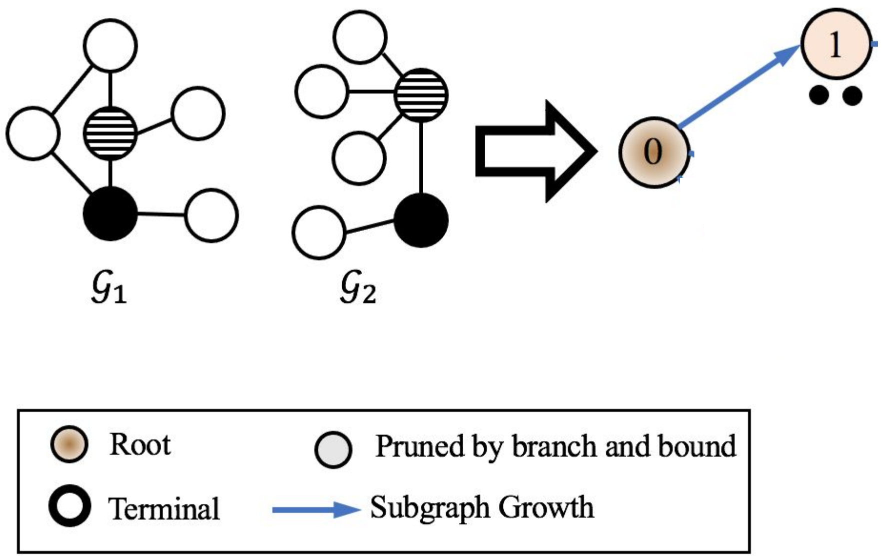
Existing Work on MCS Detection

State-of-the-art branch and bound search algorithm for MCS



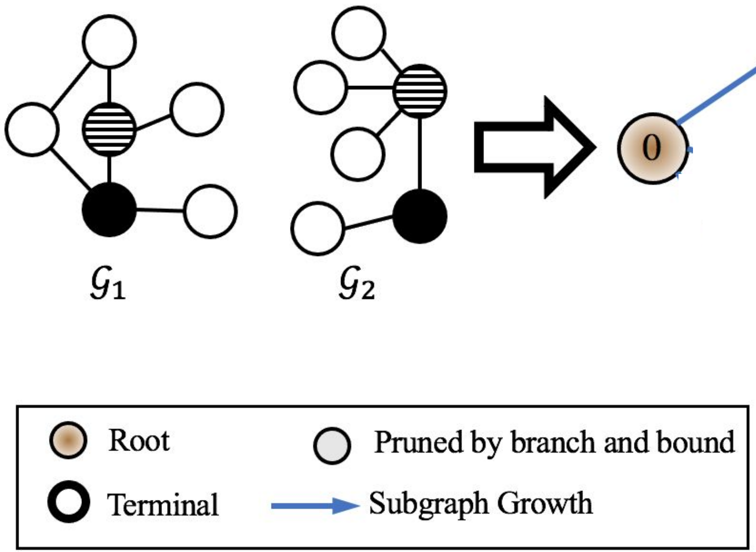
Existing Work on MCS Detection

State-of-the-art branch and bound search algorithm for MCS



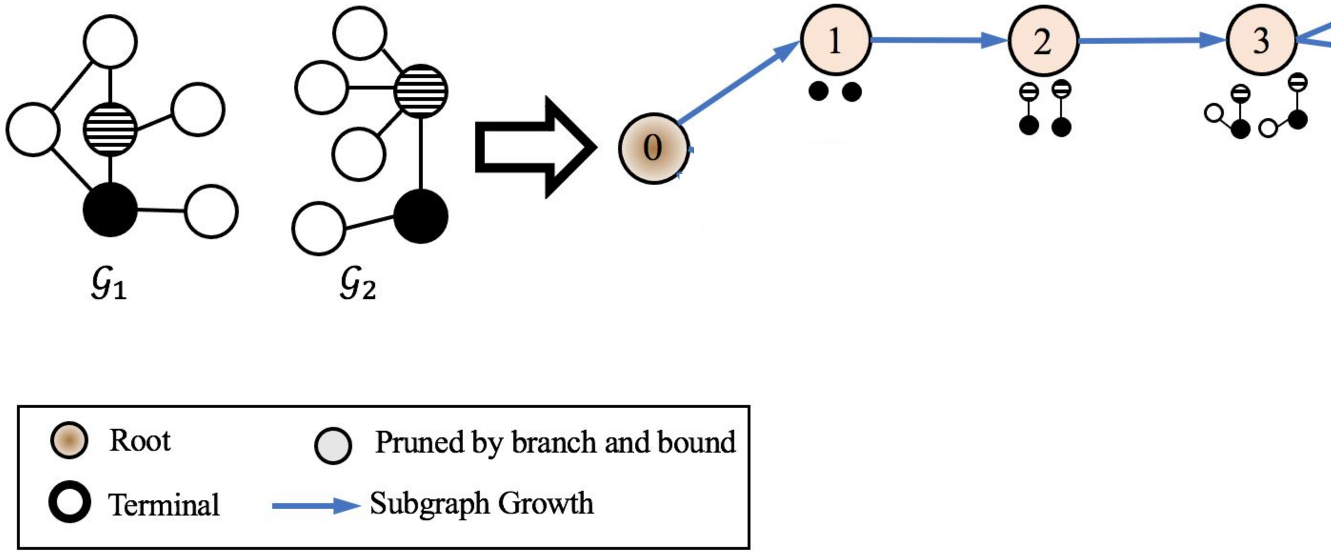
Existing Work on MCS Detection

State-of-the-art branch and bound search algorithm for MCS



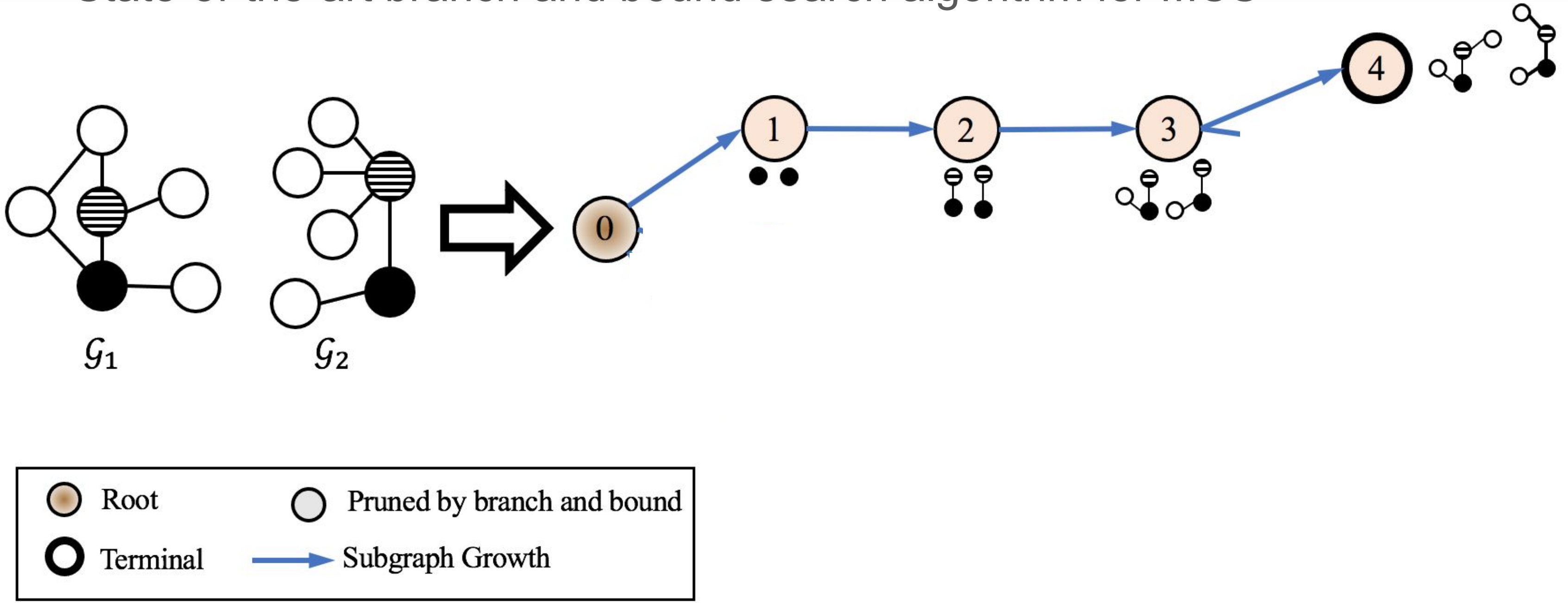
Existing Work on MCS Detection

State-of-the-art branch and bound search algorithm for MCS



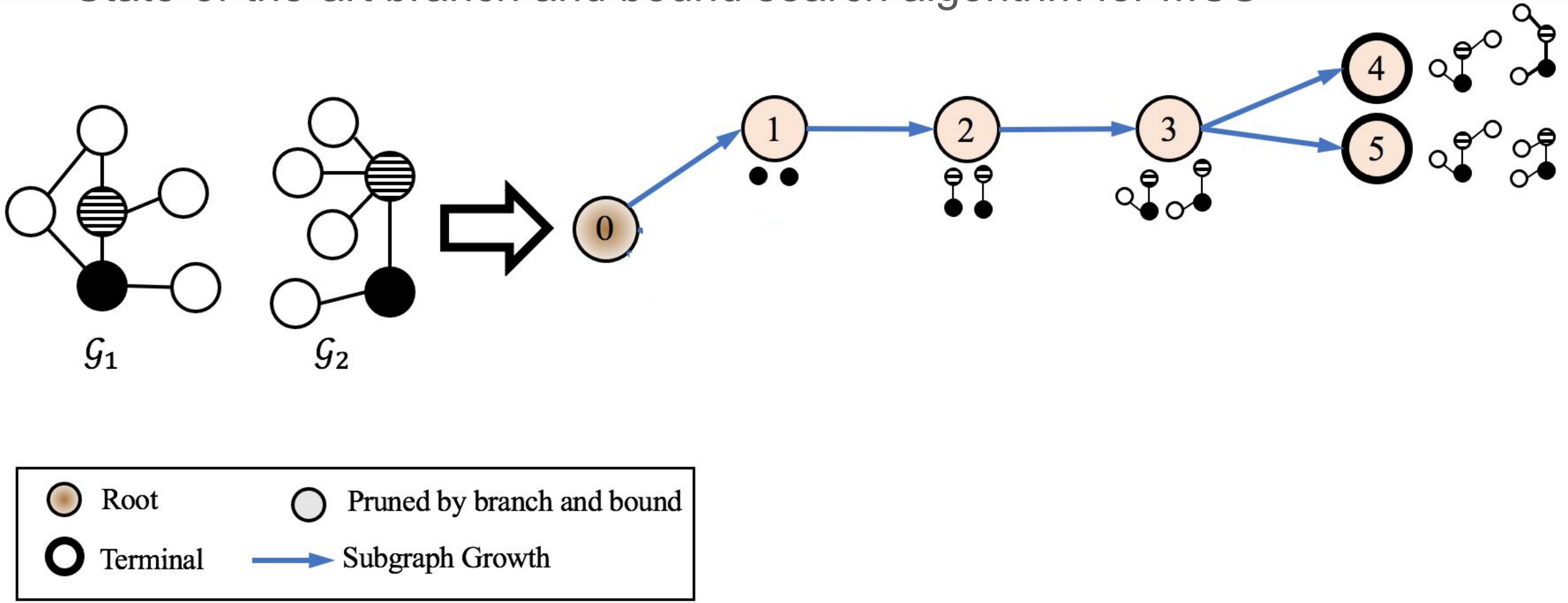
Existing Work on MCS Detection

State-of-the-art branch and bound search algorithm for MCS



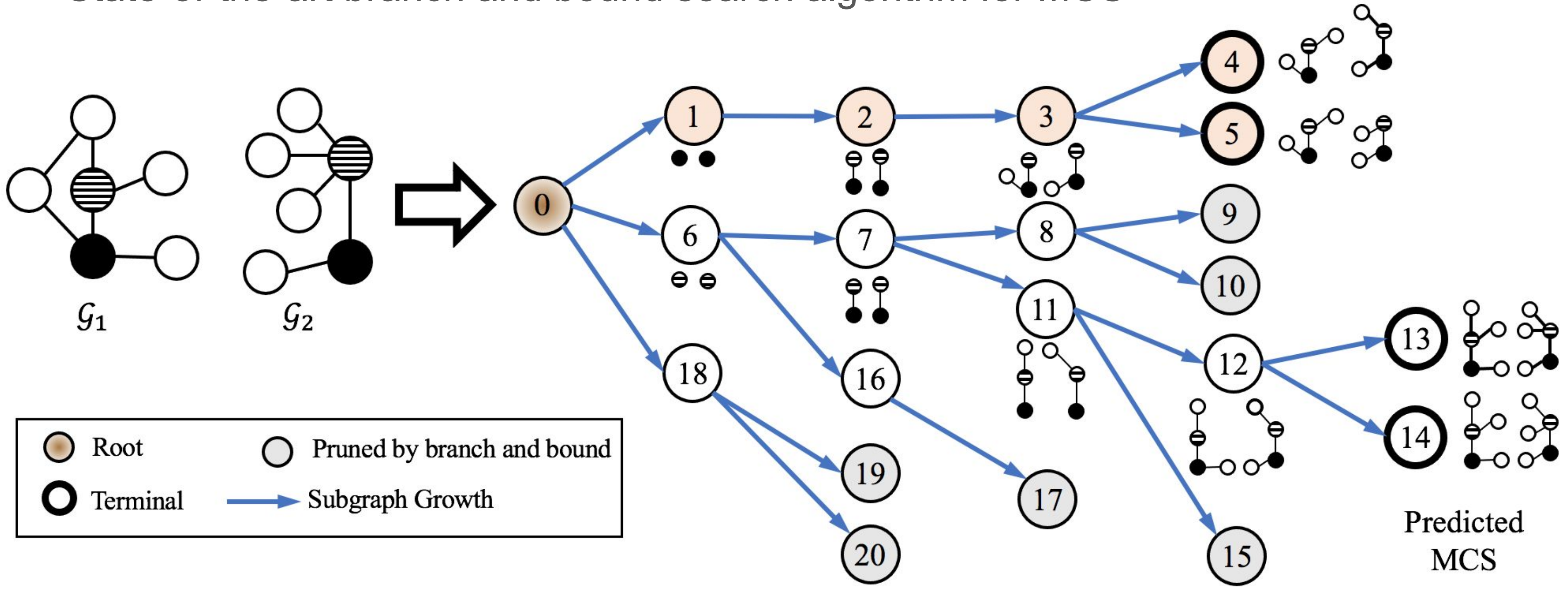
Existing Work on MCS Detection

State-of-the-art branch and bound search algorithm for MCS



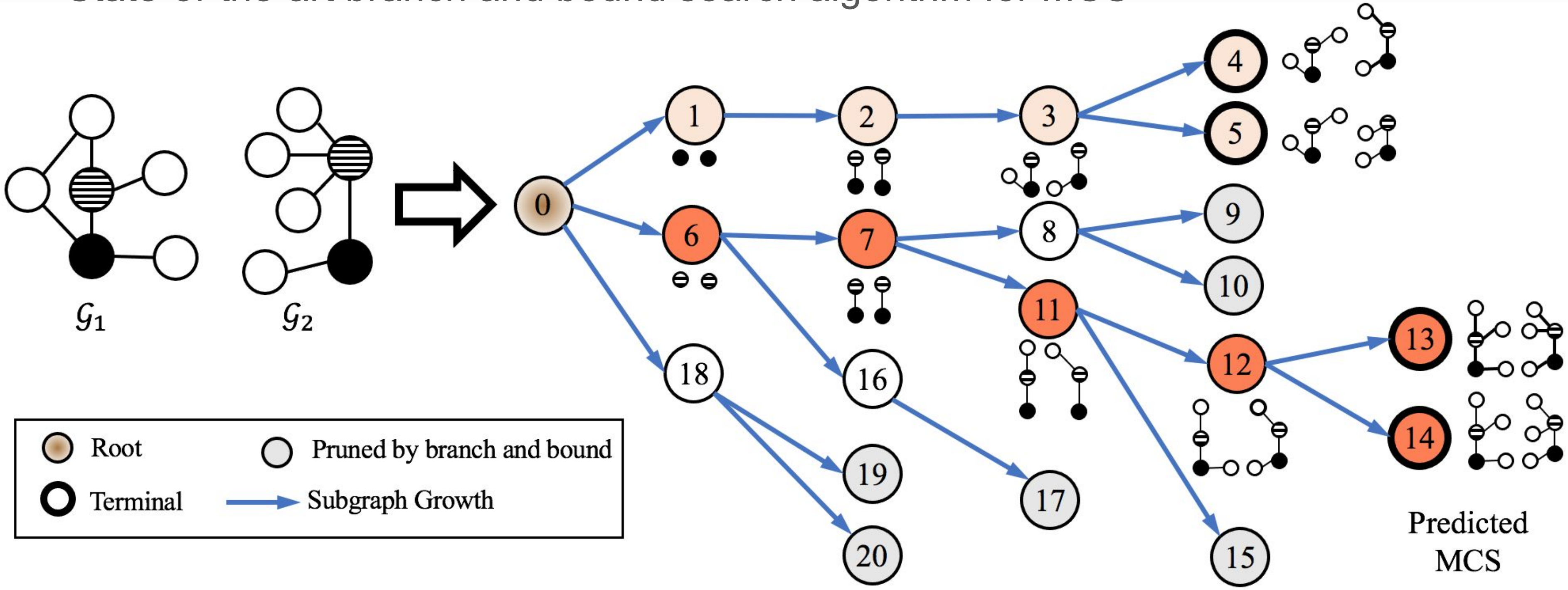
Existing Work on MCS Detection

State-of-the-art branch and bound search algorithm for MCS



Existing Work on MCS Detection

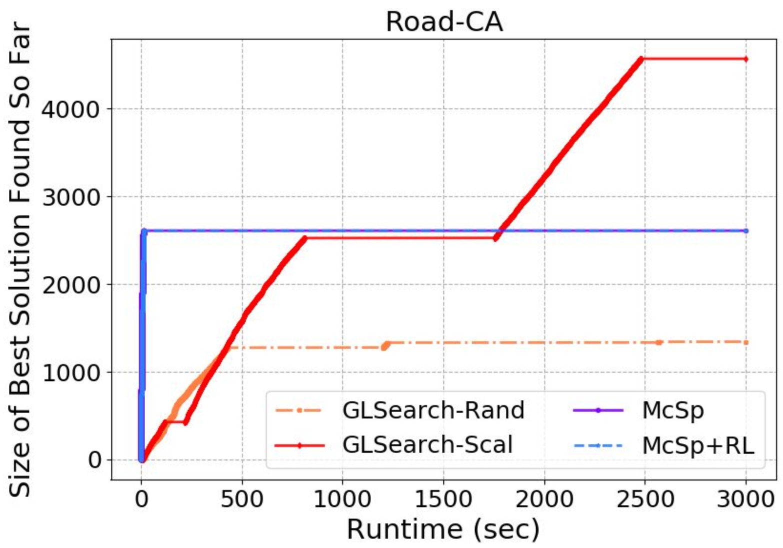
State-of-the-art branch and bound search algorithm for MCS



Existing Work on MCS Detection

State-of-the-art branch and bound search algorithm for MCS:

- Use **heuristics** or **shallow learning** to choose an action which,
  - cannot adapt to various real-world graphs
  - may lead to many wasted search iterations without finding a larger common subgraph
    - resulting in a **suboptimal solution** under a limited time budget for large graph pairs

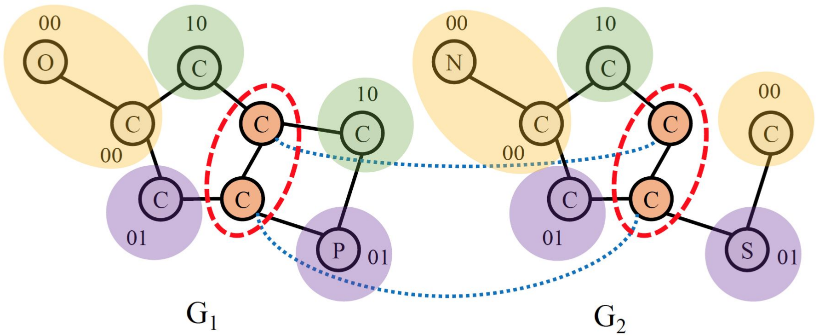


Learning to Search for MCS Detection

- Overall: Train to find the largest common subgraph with Deep Q-Learning
- Our DQN function is guided by "graph partitionings" from search:

$$Q(s_t, a_t) = 1 + \gamma \text{MLP}\Big(\text{CONCAT}\big(\text{INTERACT}(\boldsymbol{h}_{\mathcal{G}_1}, \boldsymbol{h}_{\mathcal{G}_2}), \text{INTERACT}(\boldsymbol{h}_{s1}, \boldsymbol{h}_{s2}), \boxed{\boldsymbol{h}_{\mathcal{D}c}, \boldsymbol{h}_{D_0}}\big)\Big).$$

  - graph-level embeddings ($\boldsymbol{h}_{\mathcal{G}_1}, \boldsymbol{h}_{\mathcal{G}_2}$)
  - subgraph-level embeddings ($\boldsymbol{h}_{s1}, \boldsymbol{h}_{s2}$)
  - bidomain embeddings $\boxed{\boldsymbol{h}_{\mathcal{D}c}, \boldsymbol{h}_{D_0}}$



Learning to Search for MCS Detection

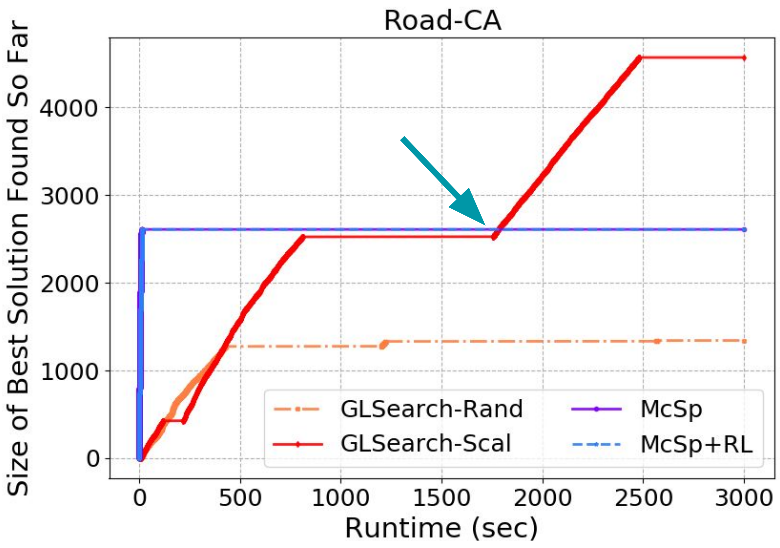
- Our DQN design factors out the action by “looking ahead”

$$Q(s_t, a_t) = \boxed{1+\gamma}\,\mathrm{MLP}\Big(\mathrm{CONCAT}\big(\mathrm{INTERACT}(\boldsymbol{h}_{\mathcal{G}_1}, \boldsymbol{h}_{\mathcal{G}_2}),$$
$$\mathrm{INTERACT}(\boldsymbol{h}_{s1}, \boldsymbol{h}_{s2}), \boldsymbol{h}_{\mathcal{D}c}, \boldsymbol{h}_{D_0}\big)\Big).$$

    - Since the immediate reward $r_t = +1$ regardless of which action to choose for the MCS detection task
        - → We can factor out the effect and look at the next state’s graph-level, subgraph-level, and bidomain embeddings
        - Add extra computation but gives the model more knowledge of the effect of choosing an action

Learning to Search for MCS Detection

- GLSearch further leverages the search algorithm by
  - Promise-based search: During inference, can **“jump” out of local minima** to an earlier search state after no progress has been made for a certain number of iterations
  - **Pre-training**: At the beginning of training, use a supervised loss function to train the DQN to predict MCS size for small graph pairs
  - **Imitation learning**: After pre-training, follow the expert trajectories provided by a heuristic-based search algorithm for MCS detection



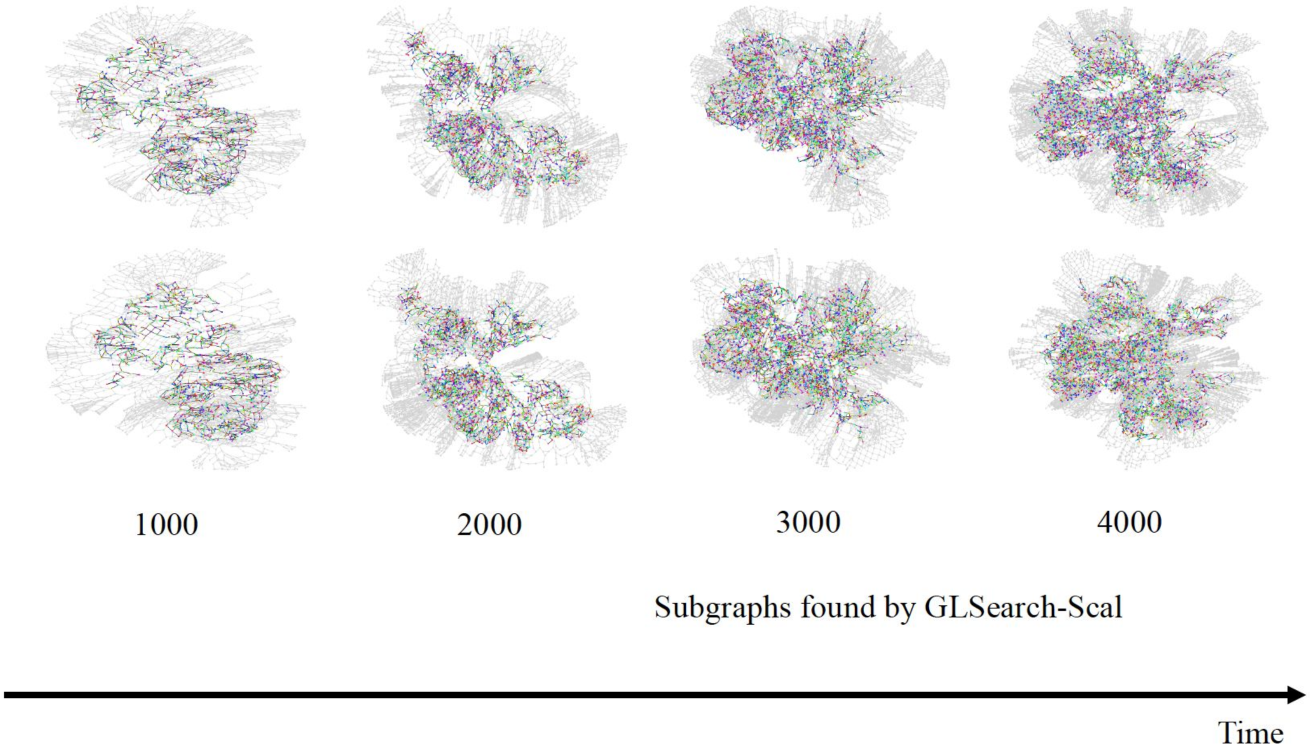
Evaluation: Effectiveness

Method	ROAD 652	DBEN 1945	DBZH 1907	DBPD 1907	ENRO 3369	COPR 3518	CIRC 4275	HPPI 2152
MCSP	0.374	0.815	0.797	0.722	0.694	0.684	0.498	0.864
MCSP+RL	0.771	0.699	0.589	0.434	0.742	0.674	0.583	0.787
GW-QAP	0.305	0.929	0.855	0.808	0.711	0.860	0.354	0.834
I-PCA	0.267	0.551	0.589	0.607	0.650	0.707	0.203	0.762
NEURALMCS	0.977	0.785	0.616	0.620	0.737	0.742	0.561	0.785
GLSEARCH-RAND	0.641	0.762	0.658	0.639	0.814	0.755	0.603	0.814
GLSEARCH	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**
BEST SOLUTION SIZE	131	508	482	521	543	791	3515	404

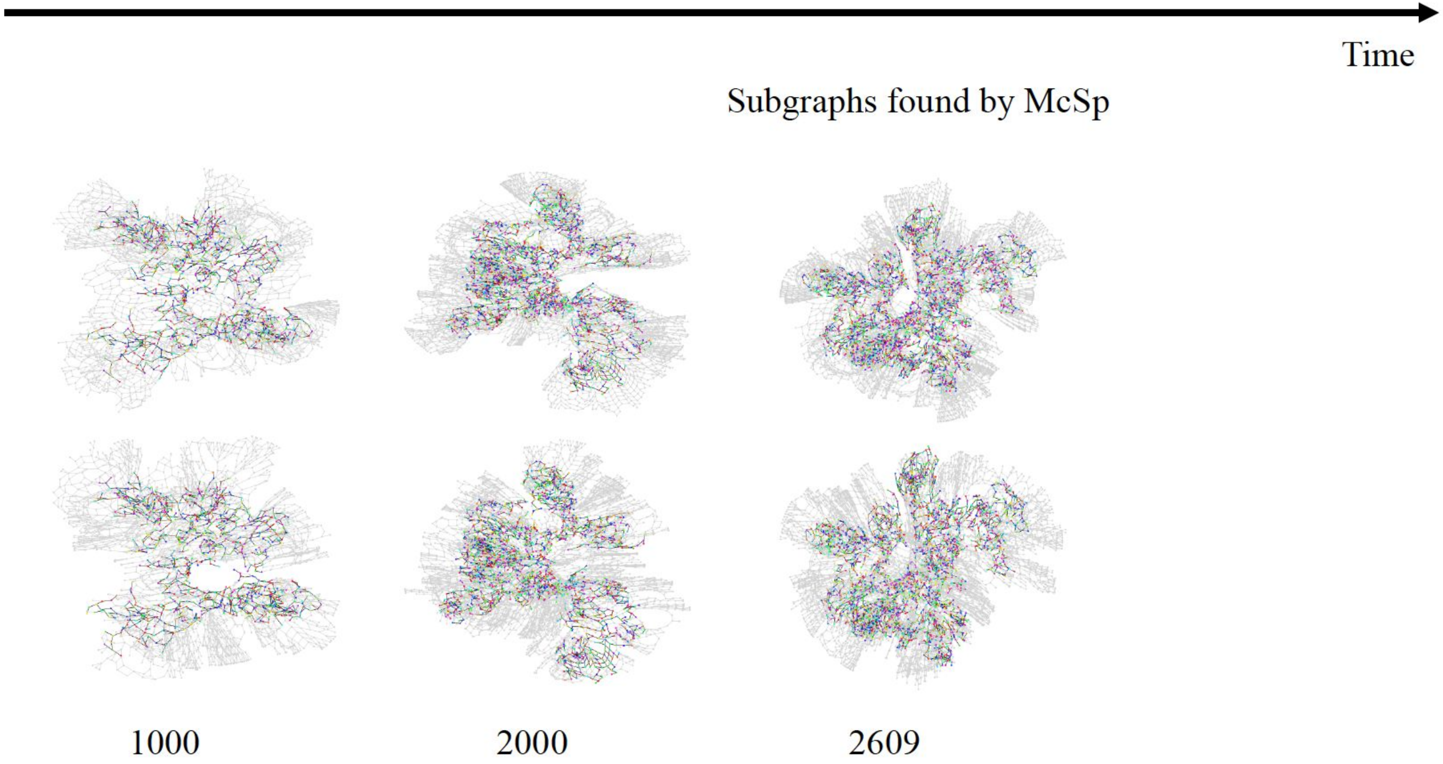
Consistently detect subgraphs that are ***larger* than all baselines**.

We make the code and datasets used in this paper publicly available.

GLSearch on California Road Network



Baseline on California Road Network



Insights and Conclusion

1. Search for MCS detection
    a. We design a learning based agent to choose smarter action at each search iteration
2. Learning in general
    a. Learning components can be further enriched by incorporating knowledge on tackling hard constraints of an NP-hard task, e.g. bidomain in our case into their model
3. Graph deep learning
    a. We enhance the existing Graph Neural Networks by leveraging non-local information
4. Reinforcement learning
    a. We show how to encode states and actions for an NP-hard task on a graph pair instead of a single graph and leverage an existing search algorithm for training the DQN

Thank you!