Reinforcement Learning Assisted Dynamic Large Scale Graph Learning

Anonymous Author(s)
Affiliation
Address
`email`

Abstract

Graph Neural Networks (GNNs) have proven to be highly effective for link and edge prediction across domains ranging from social networks to drug discovery. However, processing extremely large graphs with millions of densely connected nodes poses significant challenges in terms of computational efficiency, learning speed, and memory management. Thus making Graph Foundational Model very computationally expensive. In this work, we present a reinforcement learning (RL) assisted dynamic graph learning algorithm that addresses these scalability issues, making Graph Foundational Model computationally feasible for many use cases. Our approach provides new perspectives in Advanced Graph Machine Learning by employing an RL agent to strategically sparsify large graphs by preserving only the most salient edges for downstream applications like node classification. We demonstrate the effectiveness of our framework on an academic network containing papers, authors, and their affiliations. Our method first partitions the network into two components: a core graph of papers and a satellite graph of authors and affiliations. The RL agent then selectively merges these components by identifying and maintaining only the most informative connections between papers and authors for the node classification task. Experimental results show that our approach achieves comparable performance to baseline methods while reducing memory requirements and accelerating the learning process.

1 Introduction

Graph Neural Networks (GNNs) have demonstrated remarkable capabilities in learning meaningful representations from graph-structured data and have been successfully adapted for various heterogeneous networks, including academic collaborations, knowledge graph, and drug discovery [4, 2, 14, 15, 10]. However, their practical application is often hindered by the scale of real-world graphs, which can contain millions of nodes and billions of edges. Training GNNs on the entirety of such graphs is often infeasible due to prohibitive computational costs and memory demands. The naive inclusion of all connections can introduce further noise, degrading the model's learning quality.

A pragmatic approach to manage this complexity is to partition the graph based on semantic relevance [16, 8, 13]. We formalize this by defining a **core** graph, which contains the primary entities and their most critical relationships, and one or more **satellite** graphs, which house auxiliary nodes and their relations. In an academic network, for example, paper-citation links form the core graph, while author collaborations, institutional affiliations, and fields of study constitute satellite graphs. While this partitioning is efficient, it introduces a new challenge: how to strategically merge satellite information to enrich the core graph without reintroducing computational bottlenecks.

Submitted to the NeurIPS 2025 Workshop on Differentiable Learning of Combinatorial Algorithms (DiffCoAlg). Do not distribute.

Reinforcement learning (RL) has proven effective for various GNN optimizations, in learning sampling or topology adaptation policies [5, 9, 12]. Existing RL-based methods primarily focus on single unified graphs, rarely addressing partitioned architectures. Similarly, while graph pruning methods – ranging from random edge dropping to structural sparsification – can reduce memory costs, they lack adaptive policies for selecting meaningful connections.

In this work, we propose an RL-based framework to dynamically construct an optimal graph for GNN training,where RL's objective is to solve the combinatorial optimization problem of merging two graphs (core and satellite) for downstream task. Our method learns a selective merging policy that identifies and integrates only the most salient connections from satellite graphs, thereby discovering valuable high-order relationships while adhering to a computational budget. This advances beyond conventional pruning techniques to address the challenges of large-scale graph learning through core and satellite graph framework.

2 Related Work

2.1 Graph Pruning and Sparsification

Simple heuristics has been widely adopted such as Forest Fire [7] and edge dropping to control graph size. Network theories later inspired more sophisticated methods including subgraph extraction, motif sampling, and spectral or structure-preserving pruning strategies to retain informative local context for GNN training while drastically reducing neighborhood size, such as layer-wise sampling through GraphSAGE [3], DropEdge [11], and topological sparsification [6]. Recent advances integrate reinforcement learning as a mechanism for learning graph sampling or augmentation policies [5], including node and edge selection for mini-batch construction [9], or active subgraph expansion [12]. These methods succeed at local scalability and dynamic adaptation, but are typically designed for complete (i.e., single-partition) graph that do not address scenarios where efficient graph analytics demand on-demand resource-aware merging across independently stored components.

3 Problem Statement

3.1 Core and Satellite Graphs

Let $G = (V, E)$ represent a heterogeneous graph. We decompose G into:

- Core graph $G_{core} = (V_{core}, E_{core})$: contains primary nodes and their direct relations
- Satellite graph $G_{sat} = (V_{sat}, E_{sat})$: contains auxiliary nodes and their remaining relations

For each primary core node $v_p \in V_{core}$, we define its target set $T(v_p) \subseteq V_{sat}$ as directly connected satellite nodes to the core node. These targets serve as entry points for satellite graph queries.

3.2 Merged Graph

Given a target $\tau \in T(v_p)$, a K-hop query retrieves the edges in satellite graph within the distance of K from the targeted node τ as follows:

$$Query(\tau, K) = \{(v_1, v_2) \mid \text{dist}(v_1, \tau) \leq K, \text{dist}(v_2, \tau) \leq K, v_1, v_2 \in V_{sat}\} \tag{1}$$

The merged graph G_{merge} combines core graph with queried satellite graph expansions:

$$G_{merge} = (V_{core} \cup V_{exp}, E_{core} \cup E_{exp}) \tag{2}$$

where expanded edges are aggregated through:

$$E_{exp} = \bigcup_{v_p \in V_{core}} \bigcup_{\tau \in T(v_p)} Query(\tau, K) \tag{3}$$

This query-based framework allows controlled integration of relevant satellite information into the core graph by expanding only within specified K-hop neighborhoods of target nodes.

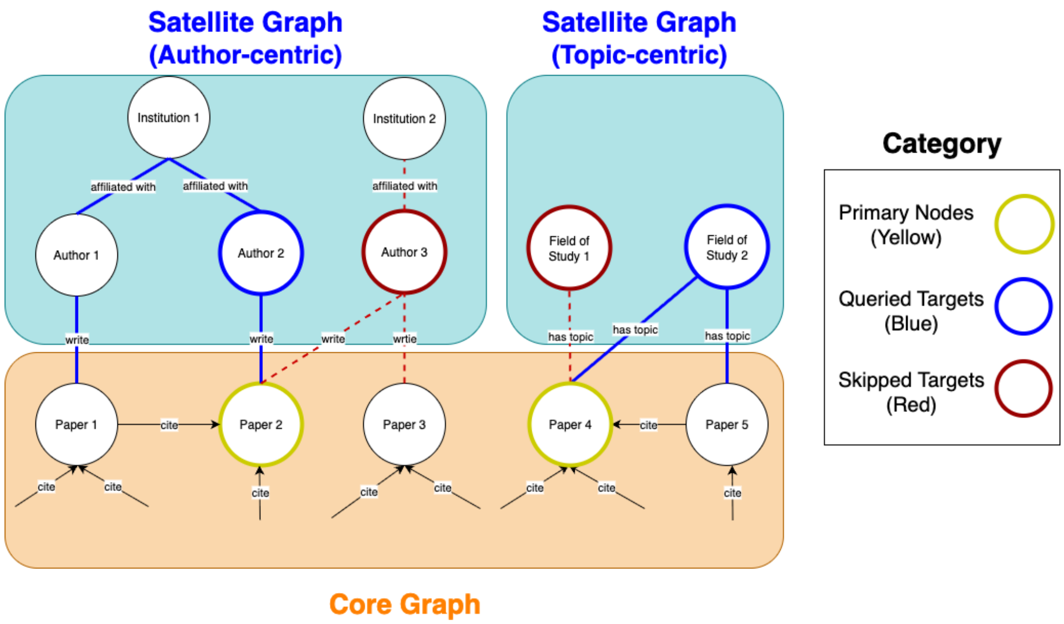

Figure 1: Overview of graph merging in the `ogbn-mag` dataset. In author-centric satellites, for paper 2 (yellow), we query target node author 2 (blue), which adds valuable institutional connections. Author 3 (red) is not queried due to lack of connectivity gain. Similarly, in topic-centric satellites, paper 4 has a target that queries field of study 2 but skips field of study 1 based on connectivity value. Blue edges show connections added to the core graph, while red edges are excluded from the merge.

4 Methodology

We propose different ways to merge core and satellite graphs through static and dynamic approaches. Figure 2 illustrates our three proposed merging techniques.

4.1 Static Graph Merging (Heuristics-based Approach)

For each core node v_p, we define its set of potential targets $T(v_p)$ that can be queried to expand the graph through satellite connections. We compare two baseline merging strategies:

- **Complete Merging:** Queries all targets $\tau \in T(v_p)$ for each core node v_p
- **Partial Merging:** Queries only the first target $\tau_1 \in T(v_p)$ for each core node v_p

Complete merging provides maximum information but is computationally expensive and may include redundant connections. Partial merging is efficient but may miss important relationships. These heuristic approaches serve as baselines for evaluating more sophisticated merging strategies.

4.2 Dynamic Graph Merging (RL-based approach)

The core graph alone may miss important connections that exist through satellite relationships. For example, in academic networks, papers that should be related may not have direct citations but share strong connections through author collaborations or institutional ties. We define these indirect relationships as high-order connections – paths between core nodes that are formed through satellite nodes (e.g., paper$_1$ connecting to paper$_2$ via paper$_1$ $\rightarrow$ author$_1$ $\rightarrow$ institution$_1$ $\rightarrow$ author$_2$ $\rightarrow$ paper$_2$).

We formalize the process of learning such high-order connections as a RL-based search problem. As shown in Figure 1, the agent evaluates target nodes (e.g., authors) based on their potential to form valuable high-order connections. Some targets (green) may enable important paths through institutional affiliations or collaborations, while others (red) may not contribute meaningful connections and are thus skipped during the merging process.

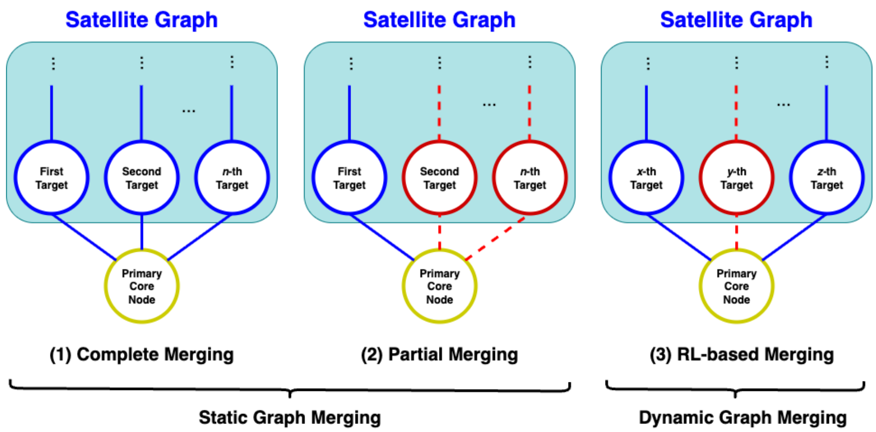

Figure 2: Merging approaches illustrated: primary node (yellow) with queried target (blue) and skipped target (red). Blue edges indicate satellite connections. Three approaches compared: complete (querying all targets), partial (querying first target only), and RL-based (selective target querying).

The RL agent follows the trajectory for graph merging with the following components:

State: At time step t, the state comprises the current merged graph structure, the core node being processed, the index of the candidate target node for querying, and the remaining query budget.

Action: For each target node (e.g., an author of a paper), the agent decides to either: 1) `query`: merge K-hop satellite edges into the current merged graph, or 2) `skip`: maintain the current merged graph.

Reward captures the number of newly discovered high-order connections when the agent queries a target and combines the satellite connections to the merged graph. The reward function is defined as:

$$R(s_t, a_t) = \begin{cases} \alpha \cdot (\text{number of high-order connections}) & \text{if high-order connection found} \\ -\beta & \text{otherwise} \end{cases}$$

where α weights the influence of high-order connections and β penalizes queries that yield no new connections.

The agent learns a policy π_θ to maximize expected cumulative reward under budget constraint B:

$$\max_{\pi_\theta} \mathbb{E}_{\pi_\theta} \left[\sum_{t=1}^{B} R_t \right]$$

This framework enables the agent to strategically identify and query targets that establish valuable high-order connections while maintaining the specified budget constraints for the number of queries. Note, here the action of the RL agent solves the problem of merging two graphs (core and satellite) in an efficient way such that only meaningful connections are added. Since, input state to the RL agent is representation of local sub-graph, and thus it helps the RL agent's policy to implicitly learn to map these local sub-graphs representation to actions such as whether to query additional neighbors for a specific target node in satellite graph. Without RL, a naive approach would generate an exponentially large search space, as each target node from the satellite graph becomes a potential candidate for addition or removal. Alternatively, heuristic-based methods tend to produce sub-optimal results, often leading to only partial merging. While a complete and accurate merge is theoretically possible, it would be prohibitively expensive in terms of memory and computational resources.

5 Experiment

5.1 Heterogeneous Graph Dataset

We use the `ogbn-mag` dataset [4], a large-scale Microsoft academic network. As shown in Table 1, the dataset comprises four types of nodes (paper, author, field of study, and institution) connected through four types of edges (citations, authorship, affiliation, and topic relationships).

Table 1: Statistics of the `ogbn-mag` dataset. Edges are marked as directed ($\rightarrow$) or undirected ($\leftrightarrow$).

Node Type	Count	Edge Type (Relationship)	Count
Paper	736,389	Paper $\rightarrow$ Paper (Citation)	5,416,271
Field of Study	59,965	Paper $\leftrightarrow$ Field of Study (Topic)	7,505,078
Author	1,134,649	Paper $\leftrightarrow$ Author (Authorship)	7,145,660
Institution	8,740	Author $\leftrightarrow$ Institution (Affiliation)	1,043,998

5.2 Core and Satellite Graph Setup

We define papers and their citation edges serve as our core graph, and we construct two satellite graphs based on two available targets (1) author and (2) field of study (i.e., topic). These satellite graphs are used to show query strategies to explore alternative connection patterns between papers.

- **Satellite Graph 1 (Author-Centric)**: This graph is composed of author, institution, and paper nodes, linked by 'Affiliation' and 'Authorship' relationships. A query on a target author retrieves their 3-hops, including institutions, authors and their papers.
- **Satellite Graph 2 (Topic-Centric)**: This graph contains paper and field of study nodes, connected by the 'Topic' and 'Authorship' relationship. A query on a target field of study expands 1-hop to retrieve all papers associated with that field of study.

5.3 Downstream Task Setup

The `ogbn-mag` dataset presents a transductive multi-class node classification task for academic papers, split temporally into pre-2018 training ($\sim$630K nodes), 2018 validation ($\sim$65K nodes), and 2019 testing ($\sim$42K nodes) to predict which of the 349 possible venues published each paper. In our experiments, we maintain this classification task but operate on our merged graph rather than the original heterogeneous structure, with those labels hidden for validation and test sets during training.

5.4 Graph Merging Performance

Table 2 demonstrates the results of merging with the topic-centric satellite graph. We implemented a Soft Actor-Critic (SAC) algorithm with discrete actions [1] to navigate the space of possible queries, utilizing its entropy maximization for exploration and its effectiveness in continuous action spaces. The RL agent uses the primary core nodes in the training set to find the high-order connections during the training time and applied to every core node during the inference time.

The *RL-based merging* shows distinct characteristics in how it compresses the graph. It reduces nodes by 62% while maintaining most edge connections, with only a 12% edge reduction. In contrast, *Partial Merging* reduces nodes by 87% and edges by 76%. These numbers suggest that the *RL-based Merging* tends to preserve targets that connect to multiple papers, resulting in a more connections compared to *Partial Merging*, while still achieving meaningful node reduction.

Table 2: Statistical comparison of merged graphs using `ogbn-mag` dataset, showing only the number of nodes and edges from the topic-centric satellite graph. Percentages marked with $\downarrow$ indicate reduction rate in nodes and edges compared to the complete graph merging.

Merged Graph	#Satellite Graph Nodes	#Satellite Graph Edges
Complete Merging	59,965	7,505,078
RL-based Merging	22,814 ($\downarrow$ 62%)	6,579,314 ($\downarrow$ 12%)
Partial Merging	7,706 ($\downarrow$ 87%)	1,767,273 ($\downarrow$ 76%)

5.5 Node Classification Performance

We selected GraphSAGE [3] as our graph learning framework's foundation, leveraging its computational efficiency with large-scale graphs and its inductive capabilities for generating embeddings of unseen nodes through neighborhood aggregation. While the *Complete Merging* approach achieves optimal test accuracy at 34.07, this comes with substantial computational overhead. Our proposed *RL-based Merging* strategy attains a comparable test accuracy of 32.96, while operating on a significantly reduced graph structure compared to the fully merged graph. In contrast, the *Partial Merging* approach yields notably inferior results with a test accuracy of 24.02. These findings substantiate our hypothesis that an intelligent, selective merging policy can effectively capture the essential information required for downstream tasks while maintaining computational efficiency. The results demonstrate that intelligently incorporating satellite information improves classification performance.

Table 3: Node classification accuracy on `ogbn-mag` dataset. The result is based on the merged graph that only contains and edges from the core and topic-centric satellite graph. Percentages marked with $\downarrow$ indicate reduction rate in accuracy compared to the complete graph merging.

Merged Graph	**Validation**	**Test**
Complete Merging	38.18	34.07
RL-based Merging	37.78 ($\downarrow$ 1%)	32.96 ($\downarrow$ 3%)
Partial Merging	28.35 ($\downarrow$ 26%)	24.02 ($\downarrow$ 29%)

6 Conclusion

We proposed a novel reinforcement learning framework for dynamic graph construction that effectively addresses the scalability challenges inherent in large-scale graph learning. Our approach introduces a principled method for selectively incorporating auxiliary information from satellite graphs into a core graph structure, resulting in an optimized and computationally efficient representation. Through our experiments on the `ogbn-mag dataset`, we demonstrated that this selective merging strategy can achieve near-competitive performance with complete graph processing while significantly reducing computational overhead.

Specifically, our RL-based graph merging achieved a test accuracy of 32.96, approaching the 34.07 benchmark of complete merging while operating on a substantially reduced graph structure. This performance validates our core hypothesis that intelligent, policy-driven graph construction can effectively capture essential relationships while maintaining computational efficiency. The framework's ability to construct crucial high-order connections while eliminating redundant information presents a promising direction for scaling GNN applications to massive real-world networks.

Looking ahead, this work establishes a foundation for future research in resource-aware graph learning, particularly in scenarios where processing complete graph structures becomes prohibitively expensive. The core-satellite framework we have introduced offers a flexible architecture that could be adapted to various domains and tasks where selective information integration is crucial for balancing performance and computational constraints.

References

[1] Petros Christodoulou. Soft actor-critic for discrete action settings. *arXiv preprint arXiv:1910.07207*, 2019.

[2] Qingyu Guo, Fuzhen Zhuang, Chuan Qin, Hengshu Zhu, Xing Xie, Hui Xiong, and Qing He. A survey on knowledge graph-based recommender systems. *IEEE Transactions on Knowledge and Data Engineering*, 34(8):3549–3568, 2020.

[3] Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. *Advances in neural information processing systems*, 30, 2017.

[4] Weihua Hu, Matthias Fey, Marinka Zitnik, Yuxiao Dong, Hongyu Ren, Bowen Liu, Michele Catasta, and Jure Leskovec. Open graph benchmark: Datasets for machine learning on graphs. *Advances in neural information processing systems*, 33:22118–22133, 2020.

[5] Weihua Hu, Bowen Liu, Joseph Gomes, Marinka Zitnik, Percy Liang, Vijay Pande, and Jure Leskovec. Strategies for pre-training graph neural networks. *arXiv preprint arXiv:1905.12265*, 2019.

[6] Wenbing Huang, Tong Zhang, Yu Rong, and Junzhou Huang. Adaptive sampling towards fast graph representation learning. *Advances in neural information processing systems*, 31, 2018.

[7] Jure Leskovec, Jon Kleinberg, and Christos Faloutsos. Graphs over time: densification laws, shrinking diameters and possible explanations. In *Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*, pages 177–187, 2005.

[8] Chuang Liu, Xueqi Ma, Yibing Zhan, Liang Ding, Dapeng Tao, Bo Du, Wenbin Hu, and Danilo P Mandic. Comprehensive graph gradual pruning for sparse training in graph neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, 35(10):14903–14917, 2023.

[9] Ziqi Liu, Zhengwei Wu, Zhiqiang Zhang, Jun Zhou, Shuang Yang, Le Song, and Yuan Qi. Bandit samplers for training graph neural networks. *Advances in Neural Information Processing Systems*, 33:6878–6888, 2020.

[10] Tengfei Lyu, Jianliang Gao, Ling Tian, Zhao Li, Peng Zhang, and Ji Zhang. Mdnn: A multimodal deep neural network for predicting drug-drug interaction events. In *Ijcai*, volume 3536, 2021.

[11] Yu Rong, Wenbing Huang, Tingyang Xu, and Junzhou Huang. Dropedge: Towards deep graph convolutional networks on node classification. *arXiv preprint arXiv:1907.10903*, 2019.

[12] Yuexin Wu, Yichong Xu, Aarti Singh, Yiming Yang, and Artur Dubrawski. Active learning for graph neural networks via node feature propagation. *arXiv preprint arXiv:1910.07567*, 2019.

[13] Carl Yang, Yuxin Xiao, Yu Zhang, Yizhou Sun, and Jiawei Han. Heterogeneous network representation learning: Survey, benchmark, evaluation, and beyond. *arXiv preprint arXiv:2004.00216*, 2020.

[14] Zheng Yu. Predicting drug-drug interactions using heterogeneous graph neural networks: Hgnn-ddi. 2024.

[15] Chengcheng Zhang, Yao Lu, and Tianyi Zang. Cnn-ddi: a learning-based method for predicting drug–drug interactions using convolution neural networks. *BMC bioinformatics*, 23(Suppl 1):88, 2022.

[16] Yuchen Zhong, Junwei Su, Chuan Wu, and Minjie Wang. Heta: Distributed training of heterogeneous graph neural networks. *arXiv preprint arXiv:2408.09697*, 2024.

Appendix

RL Agent Setups

Our implementation of SAC focuses on citation graph exploration. The observation space is defined as a 66-dimensional vector that encodes 128 features representing the main core node, a binary indicator for available fields to query, and the remaining query budget which is initialized to 100.

The RL agent's trajectory follows a standard Markov Decision Process (MDP) cycle with the following components:

1. **STATE $\boldsymbol{s_t}$**: At each timestep t, the state s_t contains the core graph with its current set of nodes and connections.
2. **ACTION $\boldsymbol{a_t}$**: Given state s_t, the agent selects an action a_t from a continuous space $[-1, 1]$, which is discretized to make binary decisions about which nodes to query. These queries establish new satellite connections to the core graph.
3. **REWARD $\boldsymbol{R(s_t, a_t)}$**: The reward function evaluates action effectiveness by measuring the in-degree of new connections to the core graph. Successful connections yield positive rewards based on their in-degree, while unsuccessful queries incur a penalty of -3.
4. **NEXT STATE $\boldsymbol{s_{t+1}}$**: The environment transitions to state which contains the newly updated merged graph. When the episode ends, we utilize the merged graph of the last state.

The architecture employs a 64-dimensional MLP feature extractor, with both policy and Q-networks consisting of two hidden layers (64 units each). The training process runs for 100,000 episodes with key parameters including a learning rate of 5×10^{-4}, replay buffer size of 10,000, batch size of 256, discount factor of 0.99, and soft update coefficient of 0.005. We employ automatic entropy tuning where the temperature parameter is learned to maintain a target entropy. The model updates occur every 10 steps with 10 gradient steps per update, and learning begins after an initial 1,000 steps.

GraphSAGE Model Setups

Our GraphSAGE implementation uses a two-layer architecture with hidden dimension of 256, taking 128-dimensional paper features from `ogbn-mag` as input and outputting class probabilities. The model employs SAGEConv with mean aggregation, ReLU activation, and dropout rate of 0.5. Training is performed over 2000 epochs using Adam optimizer with a learning rate of 0.01 and cross-entropy loss.

Pseudocode for RL-based Dynamic Graph Merging

Algorithm 1 RL-based Dynamic Graph Merging for Large Scale Graph Learning

Require:
G_{core} : Core citation graph
G_{sat} : Satellite citation graph
K : K-hop expansion limit
B : Query budget
Ensure:
G_{merged} : Merged citation graph
Initialize $G_{\text{merged}} \leftarrow G_{\text{core}}$
for v_p in V_{core} **do** ▷ Training episodes
  $\mathcal{H} \leftarrow \emptyset$ ▷ History of queried targets
  $b \leftarrow B$ ▷ Initialize budget
  while $b > 0$ and HasSkippedTargets(v_p) **do**
    $\mathcal{T} \leftarrow$ GetAvailableTargets(v_p) $\setminus \mathcal{H}$ ▷ Get all available targets
    $s_t \leftarrow (v_p, t, \mathcal{H}, b)$ where $t \in \mathcal{T}$ ▷ State for each target
    $a_t \leftarrow Q_\theta(s_t)$
    if a_t is `query` **then**
      $P_{\text{discovered}} \leftarrow$ SatelliteQuery(G_{sat}, target, K)
      $C_{\text{new}} \leftarrow$ FilterCitingPapers($P_{\text{discovered}}$, v_p)
      if $|C_{\text{new}}|/|P_{\text{discovered}}| > 0$ **then**
        $G_{\text{merged}} \leftarrow$ MergeConnections(G_{merged}, C_{new})
        $r_t \leftarrow \alpha \cdot |C_{\text{new}}|/|P_{\text{discovered}}|$ ▷ Reward for high-order connection
      else
        $r_t \leftarrow -\beta$ ▷ Penalty for redundant query
      end if
      $b \leftarrow b - 1$ ▷ Decrease query budget
    else
      $r_t \leftarrow 0$ ▷ No reward for skip action
    end if
    $\mathcal{H} \leftarrow \mathcal{H} \cup \{\text{target}\}$
    $s_{t+1} \leftarrow (v_p, \mathcal{T}, \mathcal{H}, b)$
  end while
end for
return G_{merged}