Reinforcement Learning Discovers Efficient Decentralized Graph Path Search Strategies

Alexei Pisacane[*1], **Victor-Alexandru Darvariu**[2], **Mirco Musolesi**[1,3]
[1]University College London [2]University of Oxford [3]University of Bologna

Abstract

Graph path search is a classic computer science problem that has been recently approached with Reinforcement Learning (RL) due to its potential to outperform prior methods. Existing RL techniques typically assume a global view of the network, which is not suitable for large-scale, dynamic, and privacy-sensitive settings. An area of particular interest is search in social networks due to its numerous applications. Inspired by seminal work in experimental sociology, which showed that decentralized yet efficient search is possible in social networks, we frame the problem as a collaborative task between multiple agents equipped with a limited local view of the network. We propose a multi-agent approach for graph path search that successfully leverages both homophily and structural heterogeneity. Our experiments, carried out over synthetic and real-world social networks, demonstrate that our model significantly outperforms learned and heuristic baselines. Furthermore, our results show that meaningful embeddings for graph navigation can be constructed using reward-driven learning.

1 Introduction

Graph path search is a fundamental task in Computer Science, pivotal in various domains such as knowledge bases [1], robotics [2], and social networks [3]. Given a start node and end node, the goal is to find a path from a source to a destination in the graph that connects them and optimizes desiderata such as minimizing path length. We refer to search strategies that achieve this as *efficient*. The problem is generally framed from a centralized perspective with a global view of the network, which is impractical or infeasible for several applications. In peer-to-peer networks [4], where privacy is a primary concern, a centralized agent poses significant risks [5–7]. Large graphs may also induce scalability bottlenecks as the storage requirements of a centralized directory strain memory limitations [8]. Moreover, in dynamic networks, maintaining a consistent global view of the topology may be impossible [9]. Graph path search is of particular interest in social networks given the inherent commercial applications and potential for new insights from a social sciences perspective [10, 11].

In this paper, we will study the problem of decentralized path graph search using local information. We will consider social networks and we will discuss how the proposed method can be directly applied to any networks for which topological and node attribute information is available. Indeed, prior experiments in human social networks, such as Stanley Milgram's renowned "small world" experiment [12][2] reveals the existence of short paths in social networks that are discoverable solely through local

[*]Correspondence to `pisacane.alexei@gmail.com`.

[2]Milgram's experiment investigated the degree of connectedness among people in the United States, leading to the concept of "six degrees of separation", the idea that any two people on Earth are connected by a chain of no more than six acquaintances, which has also entered popular culture [13]. Milgram selected participants from Nebraska and Kansas. Each participant was given a letter and instructed to send it to a target person, a stockbroker living in Boston. However, they could only send the letter to someone they knew personally, characterized by certain (node) *attributes* who they thought might be closer to the target. Each recipient of the letter would then forward it to someone they knew personally, continuing this process until the letter reached the target. On average, it took about six steps for the letters to reach the stockbroker in Boston [14].

A. Pisacane et al., Reinforcement Learning Discovers Efficient Decentralized Graph Path Search Strategies. *Proceedings of the Third Learning on Graphs Conference (LoG 2024)*, PMLR 269, Virtual Event, November 26–29, 2024.

graph topology and high-level node attributes, e.g., characteristics of the individuals, such as their occupation, their high-school, and so on. Many social networks exhibit two key properties that make decentralized search with partial information possible and efficient. The first, *homophily* [15], reflects the tendency of individuals to connect with others who share similar attributes. The second is the *heterogeneity* of local structure in many networks, in which nodes are often organized into highly connected communities [16], with a smaller number of *weak ties* [17, 18] or central connectors [19] bridging these node clusters and acting as shortcuts.

Reinforcement Learning (RL) has recently been employed with a centralized perspective for discovering learned heuristics for graph search [20, 21] and reasoning over paths in knowledge graphs [22, 23], in a way that complements or outperforms classic algorithms. Motivated by the promise of RL and the goal to attain decentralized graph path search, in this paper, we propose a multi-agent RL formulation in the Decentralized Partially Observable Markov Decision Process (Dec-POMDP) framework. Agents have *only local visibility of graph topology and neighbor attributes*, and cooperate towards finding paths to the target node. We propose a method for learning in this Dec-POMDP that, in accordance with the Centralized Training and Decentralized Execution (CTDE) paradigm [24], trains an actor-critic model with node representations learned by a Graph Attention Network [25] with shared parameters. These embeddings are computed via a message-passing procedure starting from the raw node attributes and the graph topology. For this reason we name the resulting method **GARDEN**: **G**raph **A**ttention-guided **R**outing for **DE**centralised **N**etworks. At execution time, the policy is used in a decentralized fashion by all agents.

We conduct experiments on synthetic and real-world social network graphs with up to 600 nodes to evaluate our model design. Our findings highlight the superior ability of our method to utilize both homophily and local connectivity better when compared to learned and handcrafted baseline models. Moreover, we find that the learned embeddings are meaningful representations for navigation in high-dimensional feature space. Our results show that the dynamics observed in Milgram's experiment can emerge using reward-driven learning. RL is able to construct a latent feature space that effectively incorporates both node attribute information and network structure for use in graph path search. Therefore, this work supports the notion that decentralized graph path search can succeed given appropriate representations, and shows a possible mechanism for how representations similar to those inherently used by individuals may be constructed.

2 Related Work

2.1 Network Science

Search is a common operation in network applications. Various classic algorithms [26] ensure path discovery between two nodes under specific conditions. They require maintaining global knowledge of the graph structure, which, as we have argued, is impractical in certain cases due to considerations of privacy, scalability, and dynamicity. We therefore focus our attention on graph path search using only local information.

As previously discussed, our inspiration for studying this problem is Milgram's "small world" experiment [12]. The findings, later validated on a larger scale [27], support this hypothesis in social networks, which are characterized by short mean path lengths. Subsequent research [28] highlighted the discovery of effective routing strategies, emphasising the concept of homophily [15], which states that individuals seek connections to others that are similar to themselves.

In addition to homophily, many networks are characterized by a power-law degree distribution [29] and exhibit heterogeneity in node degree. In such networks, a few "hub" nodes with numerous connections coexist with many nodes having a relatively small degree. Highly connected nodes therefore offer potential shortcuts in search trajectories by bridging sparsely connected communities.

For effective search, finding the bridging node between two communities is often required. Relying solely on homophily or node degree may be ineffective, as the bridging node might lack a large degree or significant attribute similarity with the target node. In networks with large clusters, an agent may spend considerable time navigating the current cluster before reaching the desired community. Identifying *weak ties* [17] between communities is challenging using only node attributes or degrees; therefore, an effective search for weak ties requires awareness of candidate nodes' neighborhoods.

A useful lens for viewing this problem is through a “hidden metric space" [30] of node features. Assuming node features are representative of their position within this space, the probability of edge-sharing increases with decreasing pairwise attribute distance. Empirical evidence supports the efficiency of navigation using this underlying metric. If an approximation to the hidden metric using only local graph structure is feasible, a decentralized strategy could involve moving toward nodes minimizing the approximate metric [30, 31].

The approach of Simsek and Jensen [32] is most closely related to ours as it also treats the problem of search by leveraging both homophily and the node degree disparity. The algorithm uses an estimate of the statistical relationship between the attribute similarity and connection probability whose computation requires knowledge of the attributes of all the nodes in the network. In contrast, our method does not require the availability of this global information.

2.2 Reinforcement Learning for Graph Routing and Search

Reinforcement Learning methods have been applied for a variety of graph optimization problems in recent years as a mechanism for discovering, by trial-and-error, algorithms that can outperform classic heuristics and metaheuristics [33]. Their appeal stems from the flexibility of the RL framework to address a wide variety of problems, requiring merely that they can be formulated as MDPs. The most relevant works in this area treat routing and search problems over graphs.

Early work on RL for routing demonstrated the potential of the MDP formalism [34–36], but suffered from the main pitfall of tabular RL methods: poor scalability. Interest has been reignited recently by several works that employ function approximation for scaling to larger problems. In this line of work, Valadarsky et al. [37] considered learning routing policies on a dataset of historical traces of pairwise demands and applying them in new traffic conditions. The MDP is framed as learning a set of edge weights from which the routing strategy is determined. Hope and Yoneki [38] expanded on this work by introducing a Graph Neural Network (GNN) [39] technique for function approximation, showing its advantages over using simple feedforward neural networks. More recent work by Almasan et al. [40] leveraged a GNN representation trained using a policy gradient algorithm. They frame actions as the choice of a middle-point for a flow given start and target nodes, with previous action choices becoming part of the state. Other recent works on RL for routing considered optimizing a weighted combination of delay and throughput [41] and deciding how to re-route the most important flows (i.e., those with the most traffic) given an initial routing scheme [42].

Another important line of work studies how to perform search on graphs. In contrast to routing, for search tasks there is no notion of a link load associated with traversing a particular node or edge in the graph. A notable contribution in this direction is work by Pándy et al. [21], where RL agents are tasked with learning a heuristic function for augmentation of A* search. Patankar et al. [43] considered the task of validating the way in which humans perform graph navigation, adopting two theories relating topological graph properties to minimizing gaps in knowledge or compressing existing knowledge. Their DQN-based agent parameterized by a GNN was validated successfully using human graph navigation trajectories.

The problem of *knowledge graph completion* may also be viewed as graph traversal in instances with heterogeneous edge types [44] and with a target node that is not specified a priori. Das et al. [22] proposed an MDP formulation of this task, in which an agent chooses the next relationship to traverse given the current node. A proportion of the true relationships in the knowledge graph is masked and used to provide the reward signal for training the agent via REINFORCE. The M-Walk method [23] builds further in this direction by leveraging the determinism of the transition dynamics. Therefore, training with trajectories from a Monte Carlo Tree Search (MCTS) policy [45] can overcome the reward sparsity associated with the random exploration of model-free methods. Zhang et al. [46] proposed a hierarchical method that features a high-level agent for choosing a cluster in which the target may be located, and a lower-level agent that navigates within the cluster.

Lastly, we note that, while the works reviewed in this section share features of our MDP and model design, none are directly applicable to the problem formulation. Chiefly, we consider a decentralized graph path search scenario in which each agent has only partial visibility of the network.

3 Methods

In this section, we first introduce our decentralized mathematical formulation of the graph search problem. Next, we describe the proposed multi-agent reinforcement learning algorithm, which leverages learnable graph embeddings.

3.1 MDP Formulation

We frame the search problem as a Decentralized Partially Observable Markov Decision Process (Dec-POMDP) [47] taking place over an attributed, undirected, and unweighted graph structure $G = (V, E)$ with n nodes and m edges. An agent is placed on each node $u_i \in V$ in the graph, while the edges E indicate direct bidirectional communication links between agents. An attribute vector $\mathbf{x}_{u_i} \in \mathbb{R}^d$ is associated with each agent. The aim is to find a path starting from an initial node u_{src} to a designated target node u_{tgt} by passing a single global *message*.

States. The global state S_t at time t is a tuple $\langle S_t^{(1)}, S_t^{(2)}, \ldots, S_t^{(n)} \rangle$ composed of the states $S_t^{(i)} = (M_t^{(i)}, u_{\text{tgt}})$ of the individual agents. Here, $M_t^{(i)}$ is an indicator variable that denotes the presence or absence of the *message* at a given node u_i at time t. While specifying the target node u_{tgt} is required for MDP stationarity, its identity is not provided to the agent in the observation.

Actions. The joint action space $\mathcal{A} = \bigtimes_i \mathcal{A}^{(i)}$ is the product of the agent-wise action spaces. At each timestep t of an MDP episode, a node u_i receives the *message* $\mathbf{m} \in \mathbb{R}^d$ specifying the attributes $\mathbf{x}_{u_{\text{tgt}}}$ of the target node (but not its identity). It chooses as its action $A_t^{(i)}$ one of its neighbors, denoted $\mathcal{N}(u_i)$, to pass the message on to. We denote this action of node u_i passing the message to node u_j by $a_{u_i \to u_j}$. All other agents take a no-op action at this step, which has no effect. Hence, $\mathcal{A}_t^{(i)} = \{a_{u_i \to u_j} | u_j \in \mathcal{N}(u_i)\}$ if $M_t^{(i)} = 1$, and $\{\texttt{no-op}\}$ otherwise.

Observations. The environment emits a global observation $O_t = \langle O_t^{(1)}, O_t^{(2)}, \ldots, O_t^{(n)} \rangle$ at each time step t, from which each node u_i only observes its own component $O_t^{(i)}$. In accordance with our motivations, we provide agents with only local observations of the graph topology. Concretely, we equip each agent u_i with observations of 1-hop ego subgraphs G_{u_j} centered on its neighboring nodes, including visibility of pairwise edges between 1-hop neighbors. The observation will also contain information on the target node: if the agent possesses the message, it symmetrically can observe an ego graph centered on u_{tgt}. Formally, the observation $O_t^{(i)}$ is defined as $(\mathbf{m}, \{G_{u_j} | u_j \in \mathcal{N}(u_i), G_{u_{\text{tgt}}}\})$ if $M_t^{(i)} = 1$, and $\{G_{u_j} | u_j \in \mathcal{N}(u_i)\}$ otherwise. The ego graph $G_{u_j} = (V_{u_j}, E_{u_j})$ is defined such that $V_{u_j} = \mathcal{N}(u_j) \cup \{u_j\}$ and $E_{u_j} = \{(u_k, u_l) \in E | u_k \in V_{u_j} \wedge u_l \in V_{u_j}\}$.

Transitions. The message moves deterministically to the selected node, updating the indicator $M_{t+1}^{(i)}$ accordingly. Concretely, $M_{t+1}^{(i)} = 1$ if $M_t^{(j)} = 1 \wedge A_t^{(j)} = a_{u_j \to u_i}$, and 0 otherwise.

Rewards. The episode ends when the message reaches the target node, yielding a collective reward of +1 for the agents and terminating the episode. Formally, $R_{t+1}^{(k)} = 1 \; \forall k$ if $\exists i \,.\, M_t^{(i)} = 1 \wedge A_t^{(i)} = a_{u_i \to u_{\text{tgt}}}$, and 0 otherwise. To prevent agents from entering action cycles, we introduce a truncation criterion: the episode can also end after T_{max} interaction steps with the environment.

3.2 Learning Architecture

In our design, we employ the common multi-agent Reinforcement Learning paradigm of Centralized Training with Decentralized Execution (CTDE) [48]. We consider a fully collaborative setting in which the agents are all rewarded if messages are successfully delivered to the target node. The collaborative objective is formulated such that each agent selecting the optimal next action results in an optimal trajectory through the graph. Therefore, the optimal trajectory can be constructed in a decentralized manner.

As it is common in the CTDE paradigm, we utilize parameter-sharing across agent networks. In the training scheme, a centralized agent receives localized observations from individual agents at each step, and is tasked with selecting optimal actions in the search path. The optimal decision is first learnt, and then replicated and distributed to individual nodes at execution time.

At each training step, a central agent is given incomplete observations and receives sparse and delayed rewards from the environment. Given these specifications, we propose the use of a variant of the Advantage Actor-Critic (A2C) algorithm [49] to promote adequate exploration with acceptable sample efficiency. The A2C value network is also learned in a centralized fashion to guide the training of the policy network. Learning a stochastic policy (rather than a deterministic one) is important for the problem under consideration given that a short path to the target may not be available via a particular neighbor despite a high level of attribute similarity. Lastly, we incorporate entropy regularization to ensure the policy maintains a high degree of randomness while still aiming to maximize the expected discounted return.

The goal is to learn, for each agent, a policy π_{u_i} that maps observations to actions. We formulate the choice of neighbor to which the message should be transmitted based on values output by an MLP-parameterized policy network $f_\pi^{\Theta_1}$. The policy network is applied for each neighbor $u_j \in \mathcal{N}(u_i)$ of the node u_i that is currently in possession of the message at time t, and the SoftMax function is used to derive a probability distribution. Concretely, for node u_i in posession of the message $\mathbf{m}$, the policy is defined as:

$$\pi_{u_i}(a_{u_i \to u_j}|O^{(i)}) = \frac{\exp(f_\pi^{\Theta_1}([\mathbf{x}_{u_j}||\mathbf{x}_{u_{\text{tgt}}}]))}{\sum_{u_k \in \mathcal{N}(u_i)} \exp(f_\pi^{\Theta_1}([\mathbf{x}_{u_k}||\mathbf{x}_{u_{\text{tgt}}}]))}, \tag{1}$$

where $[\cdot||\cdot]$ denotes concatenation. Similarly, to estimate the value function, we pass the current node u_i and the target node u_{tgt} attributes through an MLP-parameterized value network $f_v^{\Theta_2}$:

$$v(O^{(i)}) = f_v^{\Theta_2}([\mathbf{x}_{u_i}||\mathbf{x}_{u_{\text{tgt}}}]), \tag{2}$$

Where u_i is the node in possession of the message $\mathbf{m}$ at time t. It is interesting to note that the node features that are used as input to the policy and value networks will impact the effectiveness of the learned policies. The simplest choice is to use the raw node features $\mathbf{x}_{u_i}$, and we denote the resulting algorithm as MLPA2C. We also consider the simplest extension to this model that minimally incorporates local graph topology by augmenting node attributes with node degrees, i.e., $\mathbf{x}_{u_i}^{\text{WD}} = [\mathbf{x}_{u_i}||\deg(u_i)]$. We refer to this as MLPA2CWD.

3.3 GARDEN

Recall the "hidden metric" hypothesis discussed in Section 2.1, which posits that a viable policy can be motivated by moving through the graph to reduce node distance, provided a good approximation of the underlying metric is obtained. Instead of prescribing that the raw node attributes should be used to approximate this metric, we propose that relevant node features, which capture the potentially complex interplay between attributes and topologies, *can be learned*. To do so, we suggest replacing raw node attributes with learned embeddings $\mathbf{x}_{u_i}^{\text{GAT}}$ obtained from a Graph Attention Network (GAT) [25], denoted $f_{\text{rep}}^{\Theta_3}$. These embeddings are computed via a message-passing procedure starting from the raw node attributes and the graph topology.

The method, which we refer to as Graph Attention-guided Routing for Decentralized Networks (GARDEN), is shown using pseudocode in Algorithm 1. The full set of model parameters $\Theta = \{\Theta_i\}_{i=1}^3$ is trained implicitly as we take gradient descent steps over the combined episodic loss $\sum_t L_t^{(\pi)} + L_t^{(v)}$. The node embeddings are recalculated at the start of each episode. The notation $[[\cdot]]$ denotes the partial stopping of gradients, and $H(p)$ denotes the entropy of a discrete distribution, given by $\sum_i p_i \log(p_i)$.

4 Experimental Setup

4.1 Datasets

Real-world Graphs. To assess the performance of decentralized graph strategies on real-world data, we consider several ego graphs from the Facebook social network [50] present in the SNAP [51] repository. These graphs depict individuals and their Facebook friendships. Each node is equipped

Algorithm 1 Graph Attention-guided Routing for DEcentralized Networks (GARDEN).

Input: Policy Network $f_\pi^{\Theta_1}$, Value Network $f_v^{\Theta_2}$, Graph Representation Network $f_{\text{rep}}^{\Theta_3}$, Ego Graphs $\{G_u | u \in V\}$, entropy regularization coefficient λ, discount factor γ.
Output: Learned policy, value and representation networks $f_\pi^{\Theta_1}, f_v^{\Theta_2}, f_{\text{rep}}^{\Theta_3}$.
Randomly initialize model parameters $\Theta = \{\Theta_i\}_{i=1}^3$.
for $i = 1$ **to** N_{episodes} **do**
  Initialize episode buffer $\mathcal{B}$
  Sample starting node u, target node u_{tgt}
  for $w \in V$ **do**
    Compute $\mathbf{x}_w^{\text{GAT}}$ using $f_{\text{rep}}^{\Theta_3}$
  end for
  $\mathbf{m} = \mathbf{x}_{u_{\text{tgt}}}^{\text{GAT}}$
  $t = 0$
  while $u \neq u_{\text{tgt}}$ and $t < T_{\text{max}}$ **do**
    Sample action $a_{u \to u'} \sim \pi_u(\cdot \,|\mathbf{m})$
    Move message to node u', observe reward r
    Store transition (u, u', r) in $\mathcal{B}$
    $u \leftarrow u'$
    $t = t + 1$
  end while
  Initialize episode loss $L = 0$
  for (u, u', r) in $\mathcal{B}$ **do**
    $\hat{A} = r + [[f_v^{\Theta_2}([\mathbf{x}_{u'}^{\text{GAT}} || \mathbf{m}])]] - f_v^{\Theta_2}([\mathbf{x}_u^{\text{GAT}} || \mathbf{m}])$
    $L^{(\pi)} = -\hat{A} \log f_\pi^{\Theta_1}([\mathbf{x}_{u'}^{\text{GAT}} || \mathbf{m}]) - \lambda H(\pi(u|\mathbf{m}))$
    $L^{(v)} = \hat{A}^2$
    $L = L + L^{(\pi)} + L^{(v)}$
  end for
  Take gradient descent step on L w.r.t. Θ
end for

with binary, anonymized attributes collected through surveys. Due to computational budget constraints, we select the largest connected components of 5 graphs such that they have between 100 and 600 nodes. High-level descriptive statistics for these graphs are presented in Table 4 in the Appendix.

Synthetic Graphs. We additionally consider synthetically generated graphs that are both attributed and display homophily. This allows for the creation of a diverse range of graphs with varying degrees of sparsity, enabling evaluations under different synthetic conditions. We follow the generative graph construction procedure proposed by Kaiser and Hilgetag [52], which samples node attributes uniformly from a unit box $[0, 1]^d$ and creates edges stochastically according to the rule $p((u, u') \in E) = \max(1, \beta e^{-\alpha \|\mathbf{x}_u - \mathbf{x}_{u'}\|_2})$, where α and β are scaling coefficients.

4.2 Baselines

The learned baselines we use are the MLPA2C and MLPA2CWD techniques as introduced in Section 3.2. Furthermore, we consider a suite of heuristic baselines that utilize homophily and graph structure for graph path search. The simplest baseline, *GreedyWalker*, selects the next node greedily based on the smallest Euclidean attribute distance: $\pi(u) = \text{argmin}_{u' \in \mathcal{N}(u)} \|\mathbf{x}_{u'} - \mathbf{x}_{u_{\text{tgt}}}\|_2$.

Given that deterministic policies may result in action loops, we generalize this to a stochastic agent (*DistanceWalker*) that acts via a SoftMax policy over attribute distances with a tuned temperature parameter: $\pi^{(\tau)}(u'|u) = \frac{\exp(-\|\mathbf{x}_{u'} - \mathbf{x}_u\|_2/\tau)}{\sum_{u'' \in \mathcal{N}(u)} \exp(-\|\mathbf{x}_{u''} - \mathbf{x}_u\|_2/\tau)}$. Similarly, we consider the stochastic *ConnectionWalker* agent, which uses a SoftMax policy over node degree: $\pi^{(\tau)}(u'|u) = \frac{\exp(\deg(u')/\tau)}{\sum_{u'' \in \mathcal{N}(u)} \exp(\deg(u'')/\tau)}$. Lastly, the *RandomWalker* baseline selects uniformly at random between nodes from the current neighbourhood: $\pi(u'|u) = \frac{1}{\deg(u)}$.

The stochastic agents use a temperature parameter τ to control greediness. To perform a fair comparison with our learned models, we individually tune the temperature for the DistanceWalker and ConnectionWalker models using a validation set for each graph.

4.3 Model Evaluation & Selection

For evaluating models, we consider the following metrics:

1. Mean Oracle Ratio $\bar{R}_{\text{oracle}}$: the ratio between episode length and the shortest path length averaged over all source-destination pairs;
2. Truncation Rate R_{trunc}: % of episodes exceeding the truncation length T_{max};
3. Win Rate R_{win}: % of episodes where a given agent obtains the relative shortest path length, with ties broken randomly.

To mitigate potential memorization of routes during training, especially when nodes are uniquely identifiable based on attributes, we partition the node set V into three disjoint groups: V^{train}, V^{val}, and V^{test} at ratios of $80\%/10\%/10\%$. The source node u_{src} is sampled uniformly at random from V, while the target node u_{tgt} depends on whether training, validation, or testing is performed. This ensures that the agent cannot memorize the path to a target node since, by construction, it is not encountered during training. We always sample a "fresh" source-target pair during training, while for validation and evaluation the source-target pairs are serialized and stored (such that the performance evaluated over them is consistent). The Mean Oracle Ratio $\bar{R}_{\text{oracle}}$ is used as the primary metric for model validation and evaluation.

4.4 Sensitivity Analysis of Graph Density Parameter

Given a constant graph size, reduced graph density diminishes available paths to a target [53]. This intensifies exploration challenges and heightens the risk of truncated episodes, yielding sparser reward signals in training. Motivated by this rationale, we assess GARDEN across a set of generated graphs with diverse sparsity levels. We randomly generate 10 graph topologies for $\beta \in \{0.01, 0.05, 0.1, 0.2, 0.3, 0.4, 0.5, 0.75, 1.0\}$ with number of nodes $n = 200$ and $\alpha = 30$. We train GARDEN separately for each value of β and gauge its performance against the baselines.

4.5 Ablation of Node Representation

We assess our GNN-based model against alternative designs through an ablation study on synthetic graphs. Using five random seeds and fixed graph parameters ($n = 200$, $\alpha = 30$, $\beta = 5$), we conduct experiments on our three model designs: the MLPA2C model using only the raw node attributes $\mathbf{x}$, the MLPA2C variant incorporating both node attributes and degrees $\mathbf{x}^{\text{WD}}$, and the proposed GNN-based GARDEN method, which employs learned graph embeddings $\mathbf{x}^{\text{GAT}}$.

5 Experimental Results

5.1 Facebook Graphs

As shown in Table 1, we find that GARDEN significantly outperforms baselines across all the real-world datasets and metrics we have tested on. Given the variety of attribute dimensions and densities, as displayed in Table 4 in the Appendix, we may argue that in graphs with high amounts of latent structure, our model is robust to these factors.

In Figure 3 in the Appendix, we visualize the value function learned by GARDEN on these social network ego graphs. This highlights that the values obtained by GARDEN serve as a reliable proxy for graph distance, assigning highest values to nodes in the target's cluster or clusters with strong connectivity to the target's community. Furthermore, it demonstrates the interpretability of the proposed technique for graph path search.

In Table 5 in the Appendix, we also include a runtime analysis to quantify the scalability of the proposed method. These results show that GARDEN maintains sub-millisecond per-action inference times even with CPU-only execution as the graph size increases, thanks to its decentralized nature. We therefore expect the method to maintain fast inference times even in substantially larger topologies.

Table 1: Metrics obtained by the methods on the 5 social network ego graphs. GARDEN consistently yields the best performance, followed by the DistanceWalker method.

Metric	GreedyWalker	DistanceWalker	ConnectionWalker	RandomWalker	GARDEN (Ours)
$\bar{R}_{\text{oracle}}(\downarrow)$	27.17 ± 1.09	15.98 ± 0.79	33.39 ± 1.15	33.17 ± 1.16	**10.95 ± 0.76**
	31.78 ± 1.22	13.01 ± 0.90	34.62 ± 1.17	34.27 ± 1.18	**9.89 ± 0.78**
	27.35 ± 1.19	**9.99 ± 0.78**	38.84 ± 1.42	38.88 ± 1.43	**8.74 ± 0.76**
	25.69 ± 0.88	14.33 ± 0.65	28.25 ± 1.00	28.01 ± 1.01	**12.08 ± 0.71**
	28.29 ± 0.66	22.71 ± 0.70	28.11 ± 0.69	28.22 ± 0.67	**16.97 ± 0.74**
$R_{\text{trunc}}(\downarrow)$	79.00	39.70	76.90	76.70	**14.90**
	78.00	19.50	68.50	64.10	**12.60**
	70.00	13.90	69.80	67.20	**13.50**
	88.00	44.00	81.50	79.40	**40.10**
	96.00	72.70	89.50	90.00	**48.20**
$R_{\text{win}}(\uparrow)$	7.90	26.20	2.70	2.50	**60.70**
	10.90	29.90	4.30	3.90	**51.00**
	11.90	33.80	1.50	2.00	**50.80**
	6.10	32.10	5.30	8.20	**48.30**
	2.20	24.90	7.60	10.50	**54.80**

5.2 Sensitivity Analysis of Graph Density and Temperature Parameters

In Figure 1, we show the validation performance as a function of the SoftMax temperature τ of the stochastic DistanceWalker and ConnectionWalker baselines. For both methods, a middle-ground temperature value yields shorter path lengths. Furthermore, performance is more sensitive to τ for the DistanceWalker method.

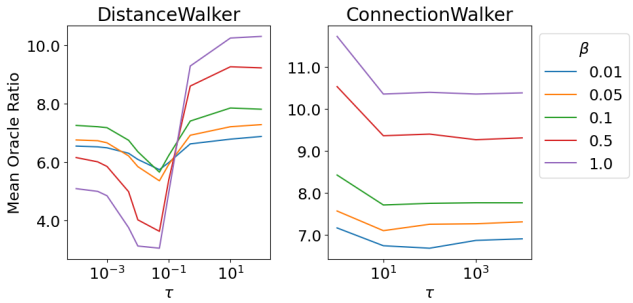

Figure 1: Mean Oracle Ratio obtained by the stochastic baselines on the validation dataset as a function of the temperature τ for varying values of β.

The sensitivity analysis for the synthetic graph density parameter β is shown in Figure 2. GARDEN consistently matches or surpasses baseline performance for all β values for both Mean Oracle Ratio and Win Rate metrics. However, DistanceWalker outperforms our model for higher β in Truncation Rate. In this setting, DistanceWalker benefits from knowledge of the "true" node attributes determining link generation and high β values leading to most connections being realized. This is in contrast with the gap on real-world datasets, for which this metric is not available: indeed, GARDEN may be seen as recovering an underlying "hidden metric".

5.3 Ablation of Node Representation

As shown in Table 2, GARDEN significantly outperforms the MLPA2C and MLPA2CWD designs that only use raw node attributes and MLP-parameterized policies in both R_{trunc} and $\bar{R}_{\text{oracle}}$. The standard MLP-parameterized model achieves the best Win Rate R_{win}, and the differences with respect to this metric are less conclusive. This can be explained by the arbitrary tie-breaking performed when path lengths match for all methods, coupled with the high density parameter $\beta = 5$ leading to shorter path lengths.

Table 2: Ablation results obtained by pairing different node representations with the proposed problem formulation and Reinforcement Learning algorithm.

Agent	$\bar{R}_{\text{oracle}}(\downarrow)$	$R_{\text{trunc}}(\downarrow)$	$R_{\text{win}}(\uparrow)$
GARDEN	**1.95±0.09**	**1.68**	34.44
MLPA2CWD	2.23±0.13	2.94	30.78
MLPA2C	2.31±0.12	4.32	**34.78**

6 Conclusions and Future Work

In this paper, we have considered the problem of decentralized search in graphs, which is motivated by privacy, scalability, and dynamicity requirements of many network modeling scenarios. Despite the lack of a central view of the network, the homophily and community structure observed in

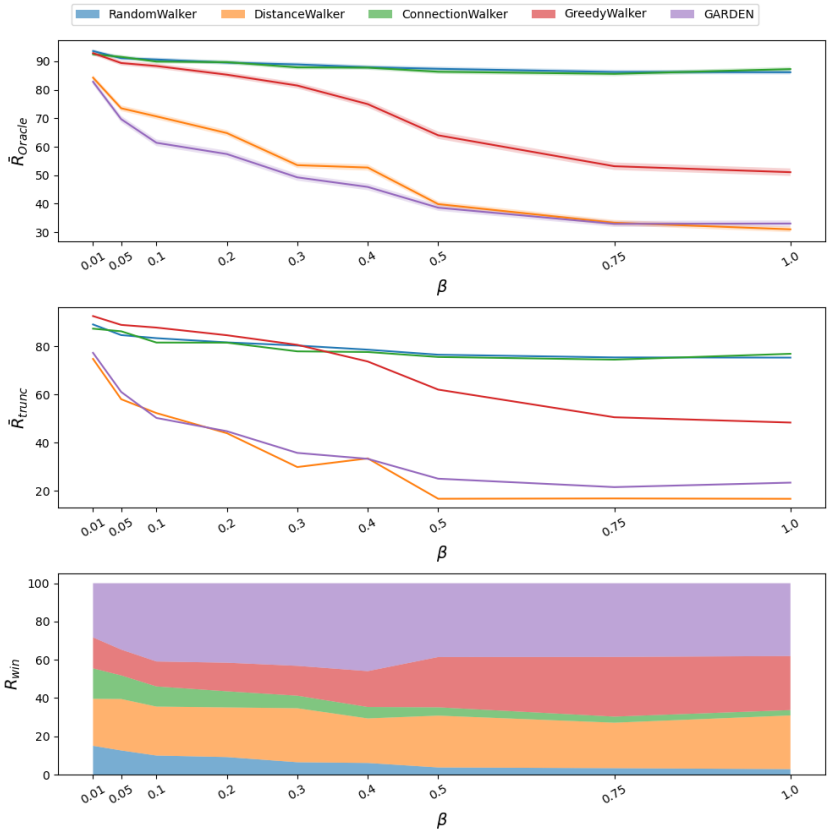

Figure 2: Metric values obtained by the methods as a function of the synthetic graph density parameter β. GARDEN generally performs best, but it is notably surpassed by DistanceWalker in the truncation rate for high values of β.

many networks can allow for decentralized agents to find short paths to a given target, as famously demonstrated by the Milgram experiment [12].

We have proposed the GARDEN method to address this problem, which trains agents in a centralized fashion and allows for decentralized policy execution. Our approach is based on message routing policies that are learned using Reinforcement Learning, paired with node features learned by a Graph Neural Network specifically for the task. Our results show that our method can outperform stochastic routing policies based on attribute or structural information alone. It is possible to observe significant improvements when searching on real-world social network graphs with non-trivial latent structures and high-dimensional node attributes.

For simplicity, we have considered a *memory-less* search procedure that is akin to a biased random walk. This means that the agents cannot react to the unsuccessful exploration of a given region of the graph before arriving at a previously visited node, and the same distribution over actions will apply independently of the historical trajectory. The problem formulation can be extended by including the history of visited nodes in the message $\mathbf{m}$ and forbidding using already-visited nodes as actions. RNNs may be used to encode the history as input to the learned models, as performed in other learning-based graph search works [21, 23].

We believe that our results provide evidence for a sort of "hidden metric" hypothesis, showing how a latent feature space amenable for graph navigation can be recovered by reward-driven learning. An interesting aspect that can be considered by future work is to compare these emergent representations with the means in which individuals take decisions for routing messages in experiments conducted over real social networks.

References

[1] Fabian M. Suchanek, Gjergji Kasneci, and Gerhard Weikum. Yago: a core of semantic knowledge. In *WWW'07*, 2007. 1

[2] Steven M. LaValle. *Planning Algorithms*. Cambridge University Press, 2006. 1

[3] Duncan J. Watts, Peter Sheridan Dodds, and M. E. J. Newman. Identity and search in social networks. *Science*, 296(5571), 2002. 1

[4] Andrew Oram. *Peer-to-Peer: Harnessing the Benefits of a Disruptive Technology*. O'Reilly Media Inc, 2001. 1

[5] Réka Albert, Hawoong Jeong, and Albert-László Barabási. Error and attack tolerance of complex networks. *Nature*, 406(6794):378–382, 2000. 1

[6] James Aspnes, Zoë Diamadi, and Gauri Shah. Fault-tolerant routing in peer-to-peer systems. In *PODC'02*, pages 223–232, 2002.

[7] Duncan S Callaway, Mark EJ Newman, Steven H Strogatz, and Duncan J Watts. Network robustness and fragility: Percolation on random graphs. *Physical Review Letters*, 85(25):5468, 2000. 1

[8] Yatin Chawathe, Sylvia Ratnasamy, Lee Breslau, Nick Lanham, and Scott Shenker. Making Gnutella-like P2P systems scalable. In *SIGCOMM'03*, 2003. 1

[9] Silvia Giordano. Mobile ad hoc networks. *Handbook of Wireless Networks and Mobile Computing*, pages 325–346, 2002. 1

[10] David Easley and Jon Kleinberg. *Networks, Crowds, and Markets: Reasoning about a Highly Connected World*. Cambridge University Press, 2010. 1

[11] Stanley Wasserman and Katherine Faust. *Social Network Analysis: Methods and Applications*. Cambridge University Press, 1994. 1

[12] Jeffrey Travers and Stanley Milgram. An experimental study of the small world problem. *Sociometry*, 32(4):425–443, 1969. 1, 2, 9

[13] John Guare. Six degrees of separation. In *The Contemporary Monologue: Men*, pages 89–93. Routledge, 2016. 1

[14] Duncan J Watts. *Six Degrees: The Science of a Connected Age*. WW Norton & Company, 2004. 1

[15] Miller McPherson, Lynn Smith-Lovin, and James M Cook. Birds of a feather: Homophily in social networks. *Annual Review of Sociology*, 27(1):415–444, 2001. 2

[16] Santo Fortunato. Community detection in graphs. *Physics Reports*, 486(3-5):75–174, 2010. 2

[17] Mark S Granovetter. The strength of weak ties. *American Journal of Sociology*, 78(6):1360–1380, 1973. 2

[18] Sinan Aral. The future of weak ties. *American Journal of Sociology*, 121(6):1931–1939, 2016. 2

[19] Mark E. J. Newman. The structure and function of complex networks. *SIAM Review*, 45(2): 167–256, 2003. 2

[20] Mohak Bhardwaj, Sanjiban Choudhury, and Sebastian Scherer. Learning heuristic search via imitation. In *CoRL'17*, 2017. 2

[21] Michal Pándy, Weikang Qiu, Gabriele Corso, Petar Veličković, Zhitao Ying, Jure Leskovec, and Pietro Liò. Learning graph search heuristics. In *LoG'22*, 2022. 2, 3, 9

[22] Rajarshi Das, Shehzaad Dhuliawala, Manzil Zaheer, Luke Vilnis, Ishan Durugkar, Akshay Krishnamurthy, Alex Smola, and Andrew McCallum. Go for a walk and arrive at the answer: Reasoning over paths in knowledge bases using reinforcement learning. In *ICLR'18*, 2018. 2, 3

[23] Yelong Shen, Jianshu Chen, Po-Sen Huang, Yuqing Guo, and Jianfeng Gao. M-walk: Learning to walk over graphs using Monte Carlo Tree Search. *Advances in Neural Information Processing Systems*, 31, 2018. 2, 3, 9

[24] Jakob Foerster, Gregory Farquhar, Triantafyllos Afouras, Nantas Nardelli, and Shimon Whiteson. Counterfactual multi-agent policy gradients. In *AAAI'18*, 2018. 2

[25] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. In *ICLR'18*, 2018. 2, 5

[26] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms*. MIT Press, 2022. 2

[27] Peter Sheridan Dodds, Roby Muhamad, and Duncan J Watts. An experimental study of search in global social networks. *Science*, 301(5634):827–829, 2003. 2

[28] Peter D Killworth and H Russell Bernard. The reversal small-world experiment. *Social networks*, 1(2):159–192, 1978. 2

[29] Albert-László Barabási and Réka Albert. Emergence of scaling in random networks. *Science*, 286(5439):509–512, 1999. 2

[30] Marián Boguñá, Dmitri Krioukov, and Kimberly C. Claffy. Navigability of complex networks. *Nature Physics*, 5(1):74–80, 2009. 3

[31] Jon Kleinberg. The small-world phenomenon: An algorithmic perspective. In *STOC'00*, pages 163–170, 2000. 3

[32] Özgür Simsek and David Jensen. Decentralized search in networks using homophily and degree disparity. In *IJCAI'05*, 2005. 3

[33] Victor-Alexandru Darvariu, Stephen Hailes, and Mirco Musolesi. Graph reinforcement learning for combinatorial optimization: A survey and unifying perspective. *Transactions on Machine Learning Research (TMLR)*, 2024. 3

[34] Justin Boyan and Michael Littman. Packet routing in dynamically changing networks: A reinforcement learning approach. *Advances in Neural Information Processing Systems*, 6, 1993. 3

[35] Samuel Choi and Dit-Yan Yeung. Predictive q-routing: A memory-based reinforcement learning approach to adaptive traffic control. *Advances in Neural Information Processing Systems*, 8, 1995.

[36] Leonid Peshkin and Virginia Savova. Reinforcement learning for adaptive routing. In *IJCNN'02*, 2002. 3

[37] Asaf Valadarsky, Michael Schapira, Dafna Shahaf, and Aviv Tamar. Learning to route. In *HotNets'17*, pages 185–191, 2017. 3

[38] Oliver Hope and Eiko Yoneki. GDDR: GNN-based Data-Driven Routing. In *ICDCS'21*, 2021. 3

[39] William L. Hamilton. *Graph Representation Learning*. Morgan & Claypool Publishers, 2020. 3

[40] Paul Almasan, José Suárez-Varela, Krzysztof Rusek, Pere Barlet-Ros, and Albert Cabellos-Aparicio. Deep reinforcement learning meets graph neural networks: Exploring a routing optimization use case. *Computer Communications*, 196:184–194, 2022. 3

[41] Zhiyuan Xu, Jian Tang, Jingsong Meng, Weiyi Zhang, Yanzhi Wang, Chi Harold Liu, and Dejun Yang. Experience-driven networking: A deep reinforcement learning based approach. In *INFOCOM'18*, 2018. 3

[42] Junjie Zhang, Minghao Ye, Zehua Guo, Chen-Yu Yen, and H. Jonathan Chao. CFR-RL: Traffic engineering with reinforcement learning in SDN. *IEEE Journal on Selected Areas in Communications*, 38(10):2249–2259, 2020. 3

[43] Shubhankar P. Patankar, Mathieu Ouellet, Juan Cervino, Alejandro Ribeiro, Kieran A. Murphy, and Dani S. Bassett. Intrinsically motivated graph exploration using network theories of human curiosity. In *LoG'23*, 2023. 3

[44] Kelvin Guu, John Miller, and Percy Liang. Traversing knowledge graphs in vector space. In *EMNLP'15*, pages 318–327, 2015. 3

[45] Cameron B Browne, Edward Powley, Daniel Whitehouse, Simon M Lucas, Peter I Cowling, Philipp Rohlfshagen, Stephen Tavener, Diego Perez, Spyridon Samothrakis, and Simon Colton. A survey of monte carlo tree search methods. *IEEE Transactions on Computational Intelligence and AI in Games*, 4(1):1–43, 2012. 3

[46] Denghui Zhang, Zixuan Yuan, Hao Liu, Hui Xiong, et al. Learning to walk with dual agents for knowledge graph reasoning. In *AAAI'22*, 2022. 3

[47] Frans Oliehoek and Christopher Amato. *A Concise Introduction to Decentralized POMDPs*. Springer, 2016. 4

[48] Ryan Lowe, Yi I Wu, Aviv Tamar, Jean Harb, OpenAI Pieter Abbeel, and Igor Mordatch. Multi-agent actor-critic for mixed cooperative-competitive environments. *Advances in Neural Information Processing Systems*, 2017. 4

[49] Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *ICML'16*, 2016. 5

[50] Jure Leskovec and Julian Mcauley. Learning to discover social circles in ego networks. *Advances in Neural Information Processing Systems*, 2012. 5

[51] Jure Leskovec and Andrej Krevl. SNAP Datasets: Stanford large network dataset collection. http://snap.stanford.edu/data, June 2014. 5

[52] Marcus Kaiser and Claus C Hilgetag. Spatial growth of real-world networks. *Physical Review E*, 69(3):036103, 2004. 6

[53] László Lovász. Random walks on graphs. *Combinatorics, Paul Erdős is Eighty*, 2(1-46):4, 1993. 7

[54] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR'15*, 2015. 12

A Appendix

A.1 Implementation Details

Our implementation is publicly available at https://github.com/flxclxc/rl-graph-search. Please see the `README.md` file for instructions on how to set up the dependencies, download the publicly available data, and run the code.

We train our models using the Adam optimizer [54] for $200,000$ episodes, evaluating performance every 100 episodes on the serialized validation set. Early stopping is applied based on the validation loss $\bar{R}_{\text{oracle}}$. Unless otherwise stated, we train and evaluate models over 10 random seeds, reporting confidence intervals where appropriate. Table 3 presents the hyperparameter configuration shared across the three model designs. We fix $\gamma = 0.99$ and the maximum episode length $T_{\text{max}} = 100$. Lastly, when providing node input features to the GAT, we append a binary indicator variable that signals that a particular node u is the center of the ego graph to the raw attributes defined as $[\mathbf{x}_w || \mathbb{I}[w = u]]$. This is used to distinguish the node from which the message must be sent.

We note that, while the GAT-based model contains 3 layers, we only use the ego graphs centered around the nodes to compute the embeddings, hence ensuring that only 1-hop visibility is provided. The first layer is fed the raw node attributes, while the second and third layers use the "latent" embeddings constructed by the previous layer. The message-passing therefore occurs over the same 1-hop subgraph in all the layers.

A.2 Additional Tables and Figures

Summary statistics. Statistics about the considered real-world graphs are shown in Table 4.

Runtime analysis. We carry out a runtime analysis to examine the scalability and computational cost of GARDEN. In terms of methodology, we calculate the mean action time, i.e., the elapsed wall clock time measured in milliseconds from the arrival of a message at a node until an action is chosen. The measurements are averaged over 100 target nodes and are carried out using an Intel i7-11800H CPU. We note that the execution of GARDEN also involves a time overhead for creating the local ego graph embeddings from $f_{\text{rep}}^{\Theta_3}$, which is reported separately.

The results are shown in Table 5. They demonstrate that both the embedding overhead time and action time increase slightly as the number of neighbors grows, but stays reasonably bounded. As it is expected, a clear hierarchy is present in which the neural network-based models require more inference time, followed by the heuristic attribute-based baselines, and finally the simple random walk baseline. This analysis highlights the fact that GARDEN maintains sub-millisecond inference

Parameter	Value
Actor Network Hidden Dimensions	64
Actor Network Layers	3
Critic Network Hidden Dimensions	64
Critic Network Layers	3
Graph Attention Network Hidden Dimensions	64
Graph Attention Network Heads	1
Graph Attention Network Layers	3
Entropy Regularization Coefficient λ	1×10^{-3}

Table 3: Hyperparameter configuration used for all the learning-based models.

Graph	n	l_G	ρ	d
1	148	2.69	0.16	105
2	168	2.43	0.12	63
3	224	2.52	0.13	161
4	324	3.75	0.05	224
5	532	3.45	0.03	262

Table 4: Number of nodes n, average shortest path length l_G, edge density ρ and attribute dimension d for each real-world ego graph.

Graph	1	2	3	4	5
Number of Nodes	148	168	224	324	532
Avg # Neighbors	22.86	19.71	28.5	15.52	18.09
Overhead Time GARDEN (ms)	0.3024	0.2349	0.5024	0.2214	0.2143
Action Time: GARDEN (ms)	0.1592	0.1715	0.1800	0.1602	0.1656
Action Time: MLPA2CWD (ms)	0.1446	0.1548	0.1686	0.1532	0.1520
Action Time: MLPA2C (ms)	0.1499	0.1487	0.1518	0.1464	0.1469
Action Time: GreedyWalker (ms)	0.0259	0.0256	0.0323	0.0281	0.0305
Action Time: DistanceWalker (ms)	0.0657	0.0697	0.0735	0.0674	0.0685
Action Time: ConnectionWalker (ms)	0.0453	0.0522	0.0562	0.0456	0.0517
Action Time: RandomWalker (ms)	0.0006	0.0006	0.0007	0.0006	0.0006

Table 5: Runtime analysis comparing average time for calculating actions across all models, including overhead computation time of local graph embeddings for GARDEN.

times even with CPU-only execution as the graph size increases, thanks to its decentralized nature. Notably, the total inference time is lower on the largest 532-node graph compared to the smallest 148-node graph, owing to differences in density. We therefore expect the method to maintain fast inference times even in substantially larger topologies.

Interpretability of the learned value function. In Figure 3, we plot GARDEN's learned value function $f_v^{\Theta_2}$ across the social network ego graphs. Brighter colors indicate a higher estimated value function relative to the target node, which is indicated with a black arrow and chosen randomly from the respective test sets. For comparison, we also plot the implicit preferability score $-\|\mathbf{x}_u - \mathbf{x}_{u_{\text{tgt}}}\|_2/\tau$ generated by the best-performing baseline, DistanceWalker, for the same source-target pairs. DistanceWalker struggles with Euclidean pairwise attribute distance due to high dimensionality and sparsity of node attributes. Conversely, values obtained by GARDEN serve as a reliable proxy for graph distance, assigning highest values to nodes in the target's cluster or clusters with strong connectivity to the target's community.

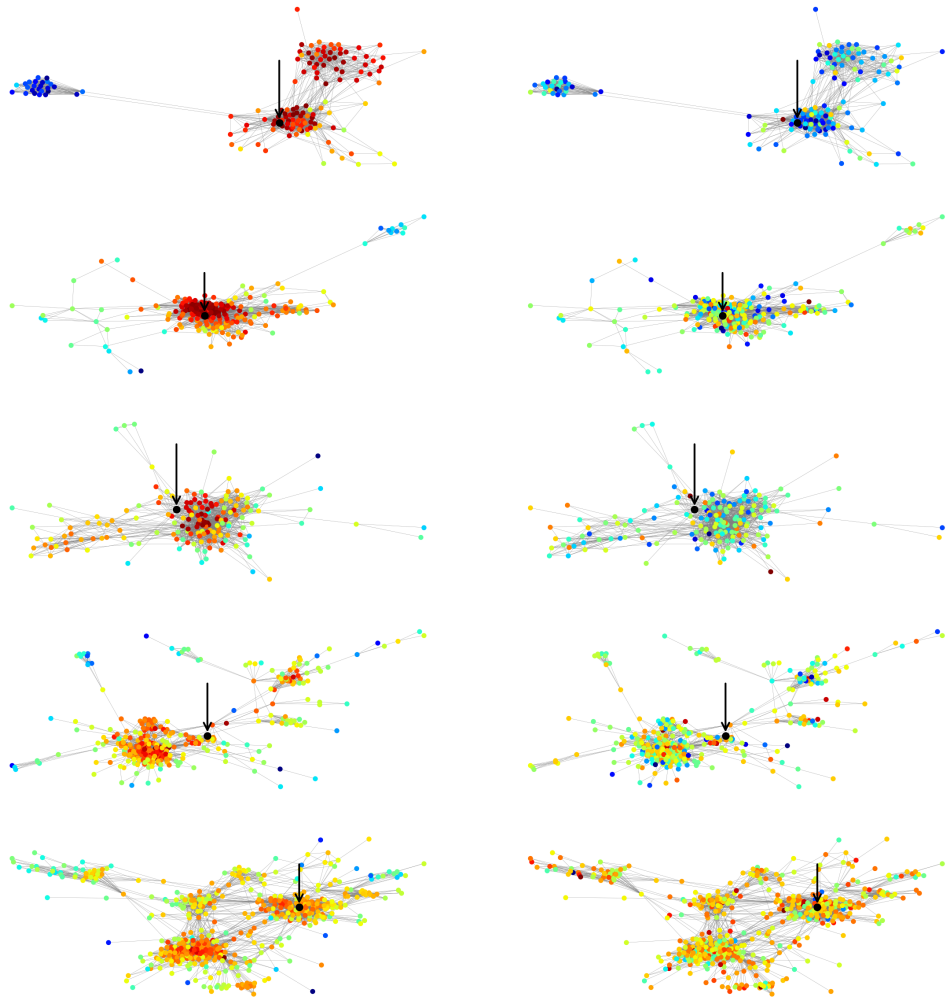

Figure 3: Visualization of the learned value function $v(u, u_{\text{tgt}})$ learned by GARDEN for each node u (left) and preferability score $-\|\mathbf{x}_u - \mathbf{x}_{u_{\text{tgt}}}\|_2/\tau$ of the DistanceWalker baseline (right) for the social network graphs. The black arrows indicate the target node, while the color intensities of the other nodes are proportional to the value function learned by GARDEN (left) or baseline score (right). Concretely, dark red nodes indicate high proximity to the target, while dark blue nodes reflect low proximity. GARDEN recovers meaningful values for performing graph navigation, effectively leveraging proximity in both node attributes and topological structure.