Where are next facilities: Emergency planning in non-Euclidean space using graph-based reinforcement learning

Xilong Zhao[1,2], Wenxuan Guo[1,2], Yaohui Jin[1,2] and Yanyan Xu[1,2†]
[1] *MoE Key Lab of Artificial Intelligence, AI Institute, School of Computer Science, Shanghai Jiao Tong University*
[2] *Data-Driven Management Decision Making Lab, Shanghai Jiao Tong University*
Shanghai 200240, China
[†] Email: yanyanxu@sjtu.edu.cn

***Abstract*—Facility siting is a common class of combinatorial optimization problems with widespread applications in real-world scenarios. In applications such as logistics and delivery, the platform needs to perform quasi-real-time planning to meet the dynamic needs of users. Moreover, the establishments of shelters and temporary medical centers in response to unexpected events also require quasi-real-time planning. However, when it comes to large-scale scenarios, traditional combinatorial optimization solvers are unable to provide a satisfactory solution within the required time frame. Unlike the typical p-median problem, real-world facility planning mostly builds one or more additional facilities upon existing infrastructure rather than originating from scratch. Meanwhile, the distances between locations are often computed within non-Euclidean spatial contexts due to the constraints of road networks. For this scenario, we propose an innovative reinforcement learning method to incrementally predict optimal sites for new facilities by leveraging graph neural networks in non-Euclidean space. Our approach exhibits adaptability to accommodate fluctuations in population dynamics. Experiments results on real-world cities show that our method achieves over 1000 times speedup compared to Gurobi on the Los Angeles dataset with 2656 nodes, with the gap of solution under 5% on all datasets.**

I. Introduction

Facility siting problems are a common class of problems in urban planning [1]–[3]. Given the population distribution and the locations of facility candidates, the objective is to minimize the collective travel distance between people's places of residence and their nearest facilities.

Facility planning problems can be delineated into two distinct categories. The first category pertains to the comprehensive initiation of planning numerous facilities, where all facility placements are determined simultaneously [4]–[6]. This procedure possesses a greater realm for solution exploration and is particularly suitable for the systematic planning of novel types of facilities. In contrast, the latter category refers to augmenting an existing set of facilities with additional ones [7]–[9]. This typically chooses a smaller number of new facilities and is more prevalently implemented in real-world circumstances. Concerning these two distinct categories, a naive way to obtain an optimal solution to the siting problem is brute-force searching, but the computation time increases exponentially and soon becomes unacceptable as the problem scales up.

Moreover, in many real-world scenarios, there is a need for quasi-real-time planning. As shown in Figure 1, in logistics and supply chain management, logistics centers need to be selected in quasi-real-time to reduce transportation costs and meet the dynamic demand of users. In practical applications, there is a need to open new logistics centers based on those already in operation. In this case, when planning the locations of new facilities, the distributions of user demand and existing logistics centers must be particularly considered.

In some other emergency cases, for example, natural disasters and sudden public health events like the COVID-19 pandemic, a rapid response is crucial to minimizing loss and protecting the safety and health of people. These scenarios require expedient facility selection for shelters and medical centers to meet temporary demands. Although there are already some commercial and open-source solvers for such combinatorial optimization problems, when organizing large-scale events, efficient and expedient venue selection is also imperative. In network applications, the selection of nodes often exhibits high requirements for real-time performance. In these application scenarios, when the problem scale is large, the runtime of commonly used combinatorial optimization solvers such as Gurobi is unacceptable, posing time performance challenges.

As is the case for other classical combinatorial optimization problems, machine learning methods have been a powerful tool for their fast inference and generalization ability [10]–[12]. However, leveraging this tool in the facility planning problem in real urban cases can be challenging. First, the distances between different locations in real-world cities are based on road networks, thus in a non-Euclidean space. Many existing methods assume Euclidean distance between locations, and can hardly be applied to real-world problems. Second, the population distribution in the city is dynamic, so a

This work was jointly supported by the National Natural Science Foundation of China (72432007), the Shanghai Key Laboratory of Urban Design and Urban Science, NYU Shanghai Research Grants (No.2025YYXu_LOUD), and the Fundamental Research Funds for the Central Universities. The computations in this paper were run on the AI for Science Platform, supported by the Artificial Intelligence Institute at Shanghai Jiao Tong University.

robust model should adapt to this change and give reasonable predictions even after the population changes, i.e., the trained model generalizes well on a changed population. Third, in the real-world facility selection problem, the objective is often formulated as choosing new facilities given existing ones, rather than selecting from scratch. A practical model needs to take into account the current configuration for better prediction.

In this paper, we propose a reinforcement learning powered graph neural model to solve the problem of site selection in non-Euclidean space with real city data. We incorporate the population distribution, road network structure, and distribution of facilities in the city for graph construction, training and prediction. A graph neural network-based model is trained with reinforcement learning and it can adapt to population changes.

II. Related Work

We first review the traditional heuristics and meta-heuristics for the p-median problem, and then introduce previous work on combinatorial optimization problems with machine learning techniques, from supervised learning to the prevailing reinforcement learning.

A. Approximation Algorithms for the P-median Problem

The problem addressed in this paper is a variant of the P-Median Problem (PMP). As a long-standing NP-hard problem, there are a number of algorithms for solving it. Since finding the exact solutions to large-scale instances remains challenging, we focus on approximation algorithms in the following discussion. Many meta-heuristics have been applied to this problem, including variable neighborhood search [13], genetic algorithms [14], tabu search [15], etc. Since the p-median problem can also be formulated as an integer program, it is therefore compatible with Lagrangian relaxation methods [16] and general solvers such as Gurobi [17]. Other manually designed heuristics include greedy addition [18], alternate selection and allocation [19], and exchange algorithm [20].

Recently, there have been some attempts to approach this classical problem with machine learning techniques. [5] leverages reinforcement learning to and constructs solutions step-by-step. [6] uses convolutional neural networks and reinforcement learning for the weighted p-median problem. [21] focuses on relocation with a twofold objective of facility exposure and user convenience.

B. Machine Learning for Combinatorial Optimization on Graphs

Researchers have attended to a variety of combinatorial optimization problems on graphs for a long time. [22] proposes a sequence-to-sequence Ptr-Net and uses node-predictive graph neural networks for supervised learning to solve the traveling salesman problem (TSP). [10] solves the maximum cut, minimum vertex cover, and maximum independent set on large-scale graphs, outperforming traditional solvers. However, supervised approaches have several limitations. They require pre-calculated optimal solutions as labels and require significant time overhead for training. The supervised style of training therefore fails to apply to complex real datasets.

To overcome the above drawbacks, [11] proposes an unsupervised approach using a loss function that measures the current solution's compliance with constraints for training. [12] solves the circuit-SAT problem along this line. [23] uses a GNN to generate distributions over a subset of nodes, minimizing a probabilistic penalty loss function, and proposes an unsupervised learning method with theoretical guarantees.

Since common combinatorial optimization problems on graphs can be formulated as strategy selection, reinforcement learning is naturally introduced. [24] proposes a framework that employs Q-learning [25], which is subsequently applied to tasks such as DAG scheduling [26] and graph matching [27].

There are other variants in the framework of GNN combined with reinforcement learning. [28] make predictions of better strategies from existing solutions. [29] introduces the attention mechanism based on Graph Attention Network [30] and reinforcement learning to solve the problems of TSP, the capacitated VRP (CVRP), the Orienteering Problem (OP) and the Prize-Collecting TSP (PCTSP).

III. Facility siting problem

Given a weighted set of customers (demands), a set of facility candidates, and the number of facilities k, the objective of this problem is to choose k facilities from the candidates so that the total distance between customers to their nearest facilities is minimized. Specifically, denote the set of potential facility locations as L, with a cardinality of m, and the set of user locations as U, with a cardinality of n. We represent the spatial relationships between users and facilities using an $n \times m$ distance matrix D. Each element d_{ij} in this matrix corresponds to the distance between the user at location i and the facility at location j, where $i \in U$ and $j \in L$. The optimization objective can be formulated as follows:

$$\min \sum_{i \in U} \min_{j \in J} d_{ij}, \tag{1}$$

subject to the constraint that the set of facilities $J \subseteq L$ and $|J| = k$. In addition, we introduce binary variables and add constraint (Equation 7) to formulate the facility siting problem with existing facilities as follows:

$$\min \sum_{i} \sum_{j} d_{ij} x_{ij}, \tag{2}$$

subject to

$$\sum_{j} x_{ij} = 1, \forall i, \tag{3}$$

$$x_{ij} \leq y_j, \forall i, j, \tag{4}$$

$$\sum_{j} y_j = k, \tag{5}$$

$$x_{ij}, y_j \in \{0, 1\}, \tag{6}$$

$$y_j = 1, \forall j \in F, \tag{7}$$

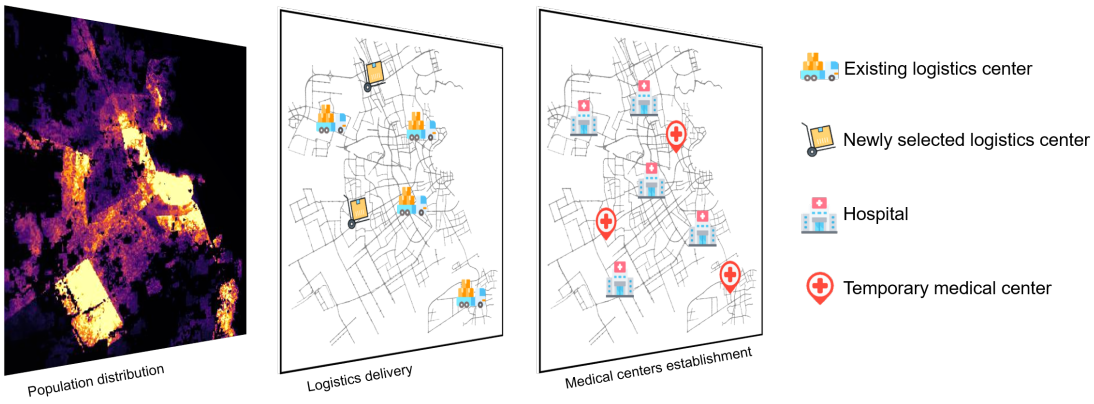


Fig. 1: Schematic diagram of facility planning. For instance, in Doha, in terms of logistics delivery, new logistic centers are opened based on existing ones and the dynamic needs of users. In response to natural disasters, new temporary medical centers are established upon existing hospitals.

where the binary variables x_{ij} indicate whether user $i \in U$ is assigned to facility location $j \in L$, while y_j represents the opening status of facility location j. F is the pre-determined set of facilities.

IV. Methodology

In this section, we introduce a graph reinforcement learning framework based on the attention mechanism. We propose the Masking Attention Selector (MAS) based on Graph Attention Network [30]. MAS takes the graph with node features and the Origin-Destination (OD) matrix-based mask as input, and outputs a specified number of nodes as a selection strategy. We use the REINFORCE algorithm [31] for reinforcement learning to train MAS and learn the selection strategy. Figure 2 demonstrates the overall working flow.

A. Feature Computation Module

For the facility siting problem, the primitive data are the population distribution and the road network. The population distribution records the number of people in each grid while the road network records the connectivity of different nodes and the length of edges. We apply Min-Max normalization to the population data, thereby scaling it to the range 0 to 1. We then aggregate the road network data according to the grid of population data, ensuring that at most one node exists in each grid (details in Section V-A) to obtain a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, with $\mathcal{V}$ and $\mathcal{E}$ denoting its nodes and edges respectively. $\mathcal{F} \subset \mathcal{V}$ denotes nodes with existing facilities. The goal of the problem is to find a set of new facilities $\mathcal{P} = \{p_1, p_2, \ldots, p_k\} \subset \mathcal{V} \backslash \mathcal{F}$ that minimizes the objective function (Equation 2). Let $v \in \{1, 2, \ldots, |\mathcal{V}|\}$ denote the index of a node. For each node, its feature vector $\boldsymbol{h}_v^0$ consists of the concatenation of four parts:

$$\boldsymbol{h}_v^0 = [\boldsymbol{h}_{v_1}^0 || \boldsymbol{h}_{v_2}^0 || \boldsymbol{h}_{v_3}^0 || \boldsymbol{h}_{v_4}^0], \tag{8}$$

where $\boldsymbol{h}_{v_1}^0$ is the population in the grid, $\boldsymbol{h}_{v_2}^0$ is the one-hot encoding of n odes and $\boldsymbol{h}_{v_3}^0$ is a binary variable indicating whether the node is in $\mathcal{F}$. Since the topological information in the road network is ignored during the subsequent attention operation, we include topological features of nodes as $\boldsymbol{h}_{v_4}^0$, i.e. degree, degree centrality, closeness centrality, and betweenness centrality. We represent the node features as a matrix form $\boldsymbol{H}^0$ where $\boldsymbol{H}_{:,i}^0 = h_i^0$.

Since the length information of the edges cannot be utilized when performing traditional attention operations among nodes, we compute a fixed OD mask and combine it into all subsequent attention operations. We first compute the OD matrix $\boldsymbol{D}$ by Dijkstra algorithm [32], where $D_{i,j}$ denotes the shortest distance between nodes i and j, and compute the mask $\boldsymbol{M}$:

$$M_{i,j} = 1 - \frac{D_{i,j}}{\max_{a,b} D_{a,b}}, \tag{9}$$

where each term of $\boldsymbol{M}$ is deflated to between 0 and 1. $\boldsymbol{M}$ is global and invariant throughout the algorithm.

B. Multi-Head Masking Attention Layer (MAT)

We now introduce multi-head masking attention layer (MAT), the core operation of our model. We use matrix $\boldsymbol{H}^l \in \mathbb{R}^{d_l \times |\mathcal{V}|}$ to denote the embedding of all nodes after the l-th GNN layer, where $\boldsymbol{H}_{:,i}^l = h_i^l$ and d_l denotes the dimension of node embeddings. We apply self-attention mechanism [33] by introducing three set of learnable matrices $\mathbb{W}^Q$, $\mathbb{W}^K$ and $\mathbb{W}^V$ with element matrices $\boldsymbol{W}^{Q_n} \in \mathbb{R}^{d_k \times d_l}$, $\boldsymbol{W}^{K_n} \in \mathbb{R}^{d_k \times d_l}$, $\boldsymbol{W}^{V_n} \in \mathbb{R}^{d_v \times d_l}$ shared by all nodes, where n is the index of the attention head. The query matrix $\boldsymbol{Q}_n \in \mathbb{R}^{d_k \times |\mathcal{V}|}$, key matrix $\boldsymbol{K}_n \in \mathbb{R}^{d_k \times |\mathcal{V}|}$ and value matrix $\boldsymbol{V}_n \in \mathbb{R}^{d_v \times |\mathcal{V}|}$ are calculated by:

$$\boldsymbol{Q}_n = \boldsymbol{W}^{Q_n}\boldsymbol{H}, \boldsymbol{K}_n = \boldsymbol{W}^{K_n}\boldsymbol{H}, \boldsymbol{V}_n = \boldsymbol{W}^{V_n}\boldsymbol{H}. \tag{10}$$

The masking attention matrix $\hat{\boldsymbol{A}}_n \in R^{|\mathcal{V}| \times |\mathcal{V}|}$ is calculated by:

$$\hat{\boldsymbol{A}}_n = \text{Softmax}(\boldsymbol{A}_n) = \text{Softmax}(\frac{\boldsymbol{M} \odot (\boldsymbol{K}_n^T \boldsymbol{Q}_n)}{\sqrt{d_k}}), \tag{11}$$

where $\odot$ denotes element-wise multiplication. Softmax( ) on a matrix is specifically computed by:

$$\hat{A}_{i,j} = \frac{\exp(A_{i,j})}{\sum_{k=1}^{|\mathcal{V}|} \exp(A_{i,k})}. \tag{12}$$

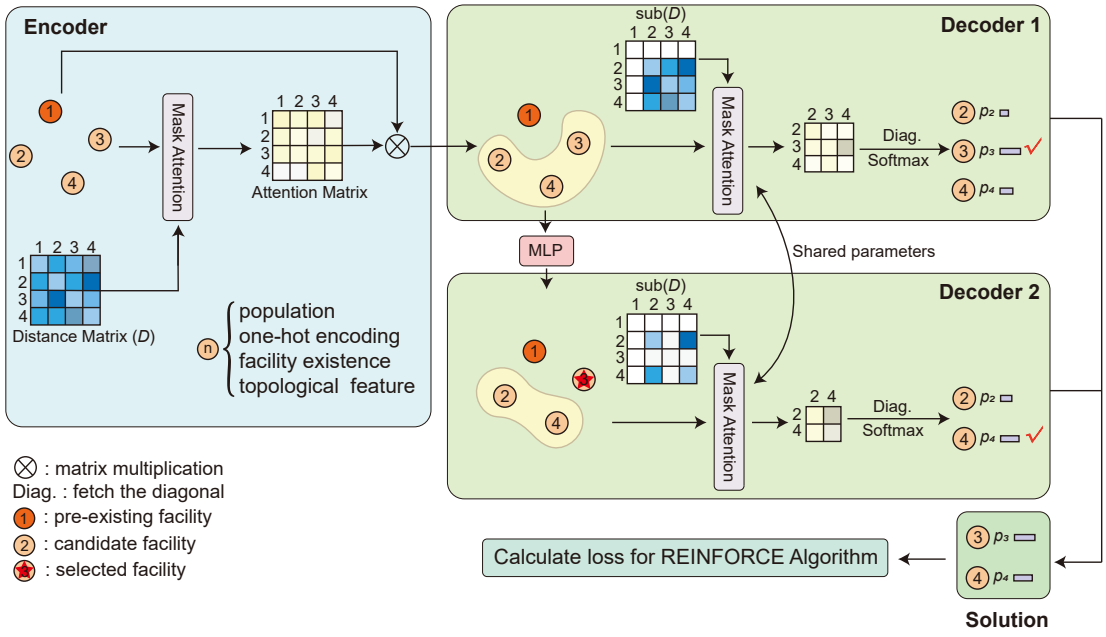


Fig. 2: The structure of Masking Attention Selector (MAS). The encoder processes node features and the OD mask via a Masking Attention Layer (MAT), producing latent node embeddings. The decoder count matches the number of new facilities, with each passing embeddings to the next via an MLP. Within each decoder, the subgraph and corresponding OD masks of candidate nodes ($\mathcal{V}\backslash(\mathcal{F}\cup\mathcal{P})$) are extracted to compute scores using the function in IV-C. The highest-scoring node is added to $\mathcal{P}$. Attention weights are shared across decoders. Sequential invocation of the decoders yields a solution and associated probabilities to compute the cost, and model parameters are updated via the REINFORCE algorithm.

We can obtain the updated node embedding matrix $\boldsymbol{H}^{l+1} \in \mathbb{R}^{d_{l+1}\times|\mathcal{V}|}$ through the output node embedding of each attention head :

$$head_n^{l+1} = \boldsymbol{V}_n \hat{\boldsymbol{A}}_n, \tag{13}$$

$$\boldsymbol{H}^{l+1} = \boldsymbol{W}^O[head_1^{l+1}||head_2^{l+1}||\cdots||head_N^{l+1}], \tag{14}$$

where $head_n^{l+1} \in \mathbb{R}^{d_v\times|\mathcal{V}|}$ denotes the output of the n-th attention head, $\boldsymbol{W}^O \in \mathbb{R}^{d_{l+1}\times N d_v}$ denotes a learnable matrix used to obtain the output. $\boldsymbol{H}_{:,i}^{l+1} = h_i^{l+1}$ denotes the output embedding of node i with dimension d_{l+1}. $[\cdot||\cdot]$ means concatenation.

C. Masking Attention Selector (MAS)

We design Masking Attention Selector (MAS) for node selection. As is shown in Figure 2, the encoder takes node embedding $\boldsymbol{H}^0$ and OD mask $\boldsymbol{M}$ as input to generate the node context vector. The decoder conducts node selection and updates node embedding.

Encoder: In the encoder, a single MAT layer is used to transform the node features into the latent space vector:

$$\boldsymbol{H}^1 = \text{MAT}(\boldsymbol{H}^0, \boldsymbol{M}, \mathbb{W}_1^Q, \mathbb{W}_1^K, \mathbb{W}_1^V, \boldsymbol{W}_1^O), \tag{15}$$

where $\mathbb{W}_1^Q, \mathbb{W}_1^K, \mathbb{W}_1^V$ each denotes a set of learnable weight matrices, $\boldsymbol{W}_1^O$ denotes a learnable weight matrix.

Decoder: The number of decoders in MAS is equal to the number of facilities to be selected. We use four learnable matrices $\boldsymbol{W}_2^Q \in \mathbb{R}^{d_k\times d_i}$, $\boldsymbol{W}_2^K \in \mathbb{R}^{d_k\times d_i}$, $\boldsymbol{W}_2^V \in \mathbb{R}^{d_v\times d_i}$ and $\boldsymbol{W}_2^M \in \mathbb{R}^{|\mathcal{V}|\times|\mathcal{V}|}$ shared across all decoders. We first perform a linear transformation on the mask by:

$$\boldsymbol{M}' = \boldsymbol{W}_2^M \boldsymbol{M}. \tag{16}$$

In the i-th decoder, we compute the set of candidate points $\mathcal{C}^i$ that have not yet been assigned facilities by $\mathcal{C}^i = \mathcal{V}\backslash(\mathcal{F}\cup\mathcal{P}^{i-1})$, where $\mathcal{P}^{i-1}$ denotes the set of newly selected facilities after $i-1$ decoders and $\mathcal{P}^0 = \emptyset$. We derive the subgraph $\mathcal{G}^i = (\mathcal{C}^i, \mathcal{E}^i)$ from the graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ where $\mathcal{E}^i = \{(u,v) \in \mathcal{E} | u \in \mathcal{C}^i, v \in \mathcal{C}^i\}$. We use $\mathcal{C}^i$ as index and intercept $\boldsymbol{H}^i$ and $\boldsymbol{M}'$ to obtain the node embeddings and OD mask of the subgraph $\mathcal{G}^i$:

$$\boldsymbol{H}_{sub}^i = sub_{\mathcal{C}^i}(\boldsymbol{H}^i), \boldsymbol{M}_{sub}^i = sub_{\mathcal{C}^i}(\boldsymbol{M}'), \tag{17}$$

where $sub_{\mathcal{C}^i}(\boldsymbol{H}^i)$ denotes the interception of columns indexed by $\mathcal{C}^i$, while $sub_{\mathcal{C}^i}(\boldsymbol{M}')$ denotes the interception of both rows and columns indexed by $\mathcal{C}^i$.

We take $\boldsymbol{W}_2^Q$, $\boldsymbol{W}_2^K$, $\boldsymbol{W}_2^V$, $\boldsymbol{H}_{sub}^i$ and $\boldsymbol{M}_{sub}$ as input to compute the attention matrix $\hat{\boldsymbol{A}}^i \in R^{|\mathcal{C}^i|\times|\mathcal{C}^i|}$ using Equation 10 and 11, arriving at the probability vector $\boldsymbol{p}^i$ of length $|\mathcal{C}^i|$:

$$\boldsymbol{p}^i = \text{Softmax}(\text{Diag}(\hat{\boldsymbol{A}}^i)), \tag{18}$$

where Diag() denotes extracting the diagonal of the matrix. The decoder selects the node with maximum probability,

updates the set of selected nodes $\mathcal{P}^i = \mathcal{P}^{i-1} \cup \{\arg\max(\boldsymbol{p}^i)\}$ and outputs the probability $p^i = \max(\boldsymbol{p}^i)$.

At the end of the i-th decoder, we update the latent vector by:

$$\boldsymbol{H}^i = \text{Softmax}(\text{MLP}(\boldsymbol{H}^{i-1})). \tag{19}$$

D. Reinforcement Learning Framework

In this section, we describe how we use the REINFORCE algorithm [31] to train MAS. We consider the graph, the set of existing facilities and the set of selected facilities as the state $(\mathcal{G}, \mathcal{F}, \mathcal{P})$. The operation of adding a node to $\mathcal{P}$ is considered as action. The set of selected facilities after all selection $\mathcal{P}^N$ is treated as a policy, where N denotes the number of demanded new facilities.

Algorithm 1 Training Masking Attention Selector(MAS) using the REINFORCE algorithm

Require: Graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, Node features $\boldsymbol{H}^0$, OD mask $\boldsymbol{M}$, set of nodes $\mathcal{V}$, set of existing facility $\mathcal{F}$, number of facilities to be selected n

for $epoch$ in many epochs **do**
  Initial the set of newly selected facilities $\mathcal{P}^0 = \emptyset$, the set of probability $\mathbb{S}^0 = \emptyset$;
  Compute latent space vector $\boldsymbol{H}^1$ by Equation 15;
  for $i = 1, 2, ..., n$ **do**
    Compute intercepted node embeddings $\boldsymbol{H}^i_{sub}$ and intercepted mask $\boldsymbol{M}^i_{sub}$ by Equation 17;
    Compute probability $\boldsymbol{p}^i$ by Equation 10, 11, 18;
    Update $\mathcal{P}^i = \mathcal{P}^{i-1} \cup \{\arg\max(\boldsymbol{p}^i)\}$;
    Update $\mathbb{S}^i = \mathbb{S}^{i-1} \cup \{\max(\boldsymbol{p}^i)\}$;
    Update node embeddings $\boldsymbol{H}^i$ by Equation 19;
  end for
  Calculate $cost$, $reward$ and $loss$ by Equation 20, 21;
  if $cost$ ¡ $cost_{\text{baseline}}$ **then**
    Update $cost_{\text{baseline}}$ to $cost$;
  end if
  Update all parameters in MAS by reducing $loss$;
end for

We compute the cost by:

$$cost = \sum_{i \in \mathcal{V}} \min_{j \in \mathcal{F} \cup \mathcal{P}^N} D_{ij}, \tag{20}$$

where $\boldsymbol{D}$ denotes the OD matrix. We maintain the current best cost as baseline $cost_{\text{base}}$, and treat $cost_{\text{base}} - cost$ as reward. We compute the loss by :

$$\begin{aligned} Loss &= (cost - cost_{\text{base}}) \sum_{t=1}^{N} \log \pi(a_t | s_t) \\ &= (cost - cost_{\text{base}}) \log(\prod_{t=1}^{N} p^N) \end{aligned} \tag{21}$$

We train the MAS by reducing the loss. The whole pipeline is described in Algorithm 1.

V. Experiments

In this section, we process real urban data to obtain road networks with population distribution. We simulate logistics and delivery scenarios and open new logistics centers based on user demand. In our experiments, we simulate user demand using real population distribution. We undertake a two-part experiment with the selection of new facilities on randomly generated existing facilities and real-world existing facilities respectively.

A. Datasets

We use real road network data and population distribution data as raw data. We treat road intersections as nodes and roads as edges between nodes. In order to manipulate the size of graphs, the city maps are gridded by a preset size, and at most one node is preserved in each grid according to node degree. The population and external edges of other nodes in the grid are aggregated to the representative for simplification. This selection strategy maintains the topology of the road network, while controlling the graph size, as illustrated in Figure 3.

In the experiment, we process data for three cities: Doha, Boston and Los Angeles. As shown in Figure 3, we choose different grid sizes and obtain 5 processed road network data: Doha-3km, Boston-500m, Boston-1km, Boston-3km, Los Angeles-1km, Los Angeles-3km and Los Angeles-9km. Statistics of different graph data are presented in Table I.

	Doha-3	**Boston-0.5**	**Boston-1**	**Boston-3**	**LA-1**	**LA-3**	**LA-9**
# of nodes	87	1435	417	46	2656	349	41
Population	5079078	1051314	1051314	1051314	6573516	6573516	6573516
# of edges	269	5416	1038	141	12282	1071	129

TABLE I: Statistics of the real city dataset

B. Hyperparameters

In the model, we set the number of attention heads to 4 and $d_k = d_v = 128$ for the learnable matrices. For training, we set the number of epochs to 1000. The patience of early stopping is set to 100 to reduce the risk of overfitting while giving the model some chance to explore. The baseline in the REINFORCE algorithm is updated every 50 epochs. We use the Adam [34] optimizer with a learning rate of 1e-4. For each instance, we run the MAS 5 times and select the best solution as the result.

C. Evaluation Metrics and Baselines

The evaluation metrics for methods include cost gap and time consumption. The cost gap of a method i is computed by calculating the gap between the cost of its solution and the optimal solution:

$$gap^i = \frac{cost^i - cost^{gt}}{cost^{gt}} \times 100\% \tag{22}$$

where $cost^i$ denotes method i's cost, $cost^{gt}$ denotes the optimal cost. In the experiment, we compare MAS with three methods: Gurobi [17], greedy algorithm and GraphSage [35]. Gurobi constructs a mixed-integer programming model for facility location problem. The greedy algorithm picks new facilities one by one, traversing all candidates at each step to

	Doha-3km		Boston-500m		Boston-1km		Boston-3km		LA-1km		LA-3km		LA-9km	
Method	Gap	Time (s)	Gap	Time (s)	Gap	Time (s)	Gap	Time (s)	Gap	Time (s)	Gap	Time (s)	Gap	Time (s)
Gurobi	0	0.3061	0	123.3600	0	7.2981	0	0.1164	0	512.8442	0	5.1193	0	0.0961
Greedy	0.00%	0.0460	0.00%	123.3787	0.00%	2.2275	0.00%	0.0109	0.71%	756.9790	0.00%	1.3782	0.00%	0.0084
GraphSAGE	6.09%	0.0348	10.94%	0.1166	9.44%	0.1299	12.64%	0.0167	22.73%	0.3717	3.46%	0.0900	29.56%	0.0152
MAS	0.04%	0.0297	3.54%	0.0384	0.79%	0.0606	0.63%	0.0297	4.56%	0.0547	1.03%	0.0538	4.03%	0.0283

TABLE II: Prediction for next three facilities based on random existing facilities

	Doha-3km		Boston-500m		Boston-1km		Boston-3km		LA-1km		LA-3km		LA-9km	
Method	Gap	Time (s)	Gap	Time (s)	Gap	Time (s)	Gap	Time (s)	Gap	Time (s)	Gap	Time (s)	Gap	Time (s)
Gurobi	0	0.3268	0	152.1684	0	8.2641	0	0.1167	0	597.1231	0	5.4605	0	0.1011
Greedy	0	0.0256	0	67.1070	0	1.1937	0	0.0056	0	381.4174	0	0.7249	0	0.0043
GraphSAGE	24.22%	0.0344	6.42%	0.1381	8.34%	0.1497	2.97%	0.0141	6.56%	0.3838	1.92%	0.0946	1.54%	0.0143
MAS	0.08%	0.0350	0.19%	0.0522	0.09%	0.0505	0.00%	0.0266	1.03%	0.0655	0.54%	0.0494	0.00%	0.0343

TABLE III: Prediction for next facility based on real-world existing facilities

	Doha-3km		Boston-500m		Boston-1km		Boston-3km		LA-1km		LA-3km		LA-9km	
Method	Gap	Time (s)	Gap	Time (s)	Gap	Time (s)	Gap	Time (s)	Gap	Time (s)	Gap	Time (s)	Gap	Time (s)
Gurobi	0	0.3129	0	157.4688	0	8.0271	0	0.1393	0	602.4718	0	5.0837	0	0.0977
Greedy	0.00%	0.0385	0.00%	101.0021	0.00%	1.8917	0.00%	0.0086	0.50%	527.2910	0.00%	1.1616	0.00%	0.0063
GraphSAGE	34.06%	0.0377	14.20%	0.1381	6.71%	0.1437	9.63%	0.0159	22.71%	0.3892	5.82%	0.1052	2.35%	0.0175
MAS	4.60%	0.0410	2.99%	0.0513	1.06%	0.0582	0.25%	0.0301	2.87%	0.0679	0.08%	0.0561	2.35%	0.0371

TABLE IV: Prediction for next two facilities based on real-world existing facilities

	Doha-3km		Boston-500m		Boston-1km		Boston-3km		LA-1km		LA-3km		LA-9km	
Method	Gap	Time (s)	Gap	Time (s)	Gap	Time (s)	Gap	Time (s)	Gap	Time (s)	Gap	Time (s)	Gap	Time (s)
Gurobi	0	0.3207	0	151.1841	0	7.8513	0	0.1183	0	587.1932	0	5.1661	0	0.0991
Greedy	0.90%	0.0534	0.42%	139.9278	0.00%	2.5860	0.00%	0.0114	0.72%	763.9729	0.00%	1.5875	0.00%	0.0083
GraphSAGE	16.31%	0.0355	22.71%	0.1377	7.21%	0.1252	25.09%	0.0199	29.97%	0.3870	6.28%	0.0957	29.00%	0.0157
MAS	3.59%	0.0429	3.80%	0.0578	1.19%	0.0652	2.22%	0.0333	4.72%	0.0699	0.32%	0.0450	0.00%	0.0509

TABLE V: Prediction for next three facilities based on real-world existing facilities

obtain the location of the facility that minimizes the cost. We use GraphSage and treat facility selection as a node prediction task. We use the same node embedding $\boldsymbol{h}_v^0$ as in Section IV-A and directly use the length of the edge as the edge embedding. We implement a two-layer GraphSage with the hidden layer dimension of 16. We apply CrossEntropyLoss as the loss function, using the Adam optimizer with a learning rate of 0.01. The best among 5 runs of GraphSAGE are used for each instance.

D. Selection on Random Facilities

In this section, we test the performance of MAS with randomly generated existing facilities $\mathcal{F}$, fixing $|\mathcal{F}| = 15$ and the number of new facilities $|\mathcal{P}| = 3$.

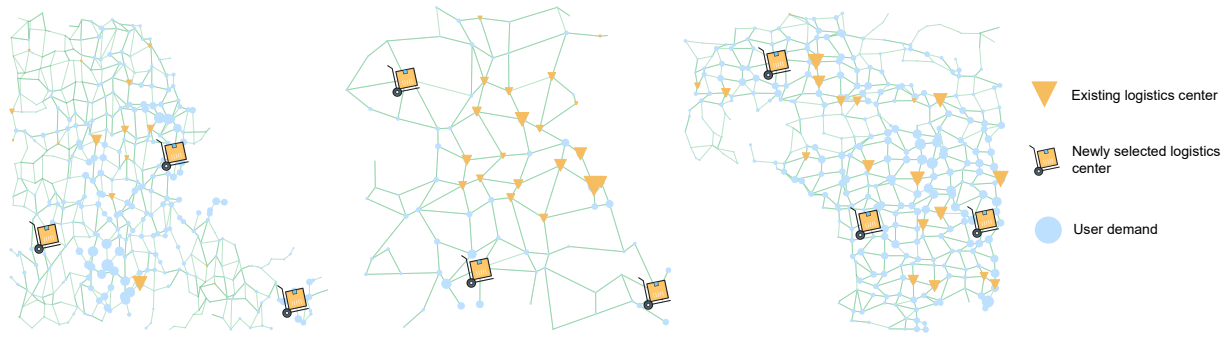
For training, 15 nodes are randomly selected as existing facilities to predict the next three. For testing, 15 facilities are selected by Gurobi from scratch and subsequently used as existing facilities to evaluate methods for selecting three new facilities, thereby avoiding extreme test cases. Results are shown in Table II. As Gurobi provides the theoretical ground truth, the gap is denoted as 0 rather than 0%, the latter denoting the actual experimental results.

In the experiment, the greedy approach reaches the optimum for the majority of the next three facility predictions. The prediction model GraphSAGE exhibits limited efficacy, while MAS manages to learn useful selection strategy given the same input features and reaches gaps below 5% in all settings. The results are close to optimal even on large graphs such as Boston-500m and LA-1km (with 1435 and 2656 nodes respectively), with a maximum gap of 4.56%.

Regarding the time performance, Gurobi sacrifices efficiency for solution quality. The greedy algorithm successfully speeds up while maintaining accuracy, but as the scale of the problem increases, its speed does not demonstrate a significant improvement compared to Gurobi. However, MAS and GraphSAGE go further and are significantly faster than the greedy algorithm in the large graphs, which suggests their potential for acceleration in larger-scale problems. Overall, the experiments on random facilities prove the learning ability of MAS and it provides a significant speedup, enabling problem-solving to approach quasi-real-time efficiency, which is even more pronounced on large-scale problems.

E. Selection on Real-world Facilities

In this section, we evaluate MAS on real-world existing facilities $\mathcal{F}$. Unlike Section V-D, we use real-world logistics centers as $\mathcal{F}$ and perturb node populations to construct the training set, fixing $|\mathcal{F}| = 20$ by sampling. This approach reflects the relative stability of road networks compared to dynamic population distributions. Population perturbation comprises two



Boston Doha Los Angeles

Fig. 3: MAS selection results for real-world facilities.

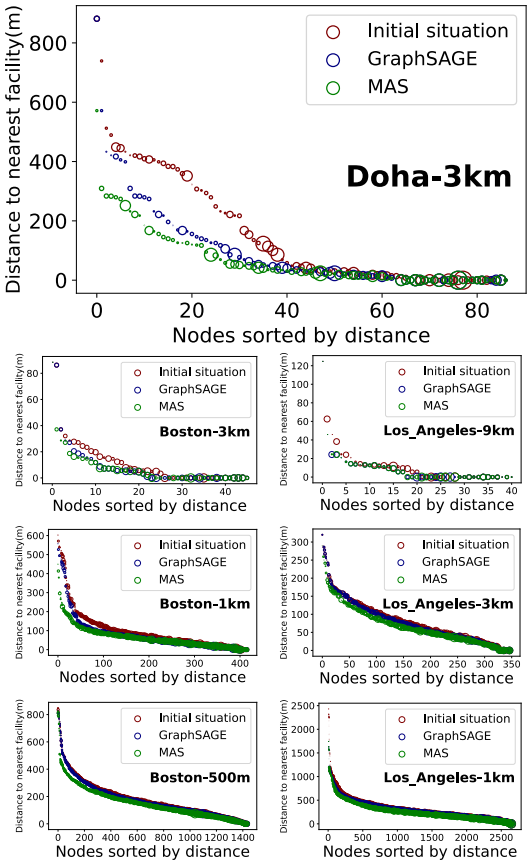


Fig. 4: Performance comparison of GraphSAGE and MAS for selecting next three facilities in real-world experiment. Nodes are sorted by descending distance to the nearest facility.

components: growth and flow. We first increase each node's population by a growth rate $r_1 \sim \text{Uniform}(0.01, 0.05)$. We then simulate population flow over $\mathcal{N}$ epochs, where in each epoch we randomly transfer a portion $r_2 \sim \text{Uniform}(0.1, 0.3)$ of one node's population to another randomly selected node. Training targets the next three facilities, while testing uses unperturbed real populations for selecting next 1 to 3 new facilities. Figure 3 illustrates MAS-predicted facility locations, with numerical results shown in Tables III, IV, V.

In the experiments, the greedy algorithm continues to achieve the ground truth value on the majority of datasets. MAS maintains a gap below 5% in all experiments and has a maximum gap of 4.72 % on the LA-1km datasets. GraphSAGE performs unstable, with gaps exceeding 20% on the LA-1km and LA-9km datasets. MAS has a significant time performance advantage over the greedy algorithm and achieves quasi-real-time efficiency on datasets with a large number of nodes. The experiment illustrates the stability of MAS when population distribution changes.

We show in Figure 4 the distribution of the distance to the nearest facility for all nodes before the selection and after the selection of the three new facilities using MAS and GragpSAGE. The nodes are sorted in descending order of distance, with their size representing the population size. MAS has an overall lower curve compared to GraphSAGE, representing that it has better optimization results. In the curve of MAS, the larger nodes are more concentrated towards the right side, which indicates that MAS, in reducing the total cost, makes the more populated nodes gain a closer distance to the nearest facility, satisfying the problem.

F. Studies on Time Performance

In previous experiments, we observe that running time scale differently across methods as the graph size increases. We therefore conduct further experiments varying both the total number of nodes and existing facilities. Since time performance is independent of specific data, we randomly generate node populations and inter-node distances, and randomly select existing facilities for next-three-facility selection. Each method is run five times with mean values reported. Figure 5 illustrates the time performance across different scales.

With fixed graph size, the greedy algorithm's running time increases significantly as the number of existing facilities grows, as it must traverse all selected facilities during computation, unlike Gurobi. GraphSAGE and MAS exhibit stable time performance because the number of existing facilities does not affect their model parameters.

As the problem size increases, greedy algorithm runtime exceeds Gurobi. At scale (4000,100), the time consumption

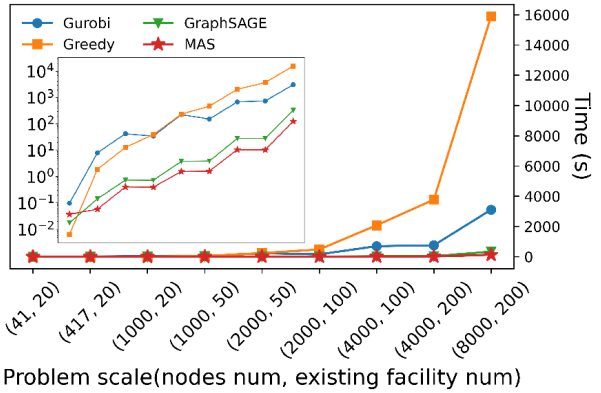


Fig. 5: Time performance evaluation for predicting three new facilities in problems of different scales. (41, 20) and (417, 20) represent Los Angeles-9km and Boston-1km. Problems of other scales are randomly generated.

of Gurobi is more than 50 times that of MAS and GraphSAGE, while the greedy algorithm requires over 3 times that of Gurobi—a gap widening further at larger scales. This advantage stems from avoiding redundant cost computations during prediction, demonstrating MAS's substantial temporal superiority over traditional solvers and heuristics for large-scale problems.

VI. Conclusion

In this paper, we present MAS, a GNN-based facility selection network, and a framework for training it by REINFORCE algorithm. MAS can handle real inter-city distances on non-Euclidean spaces and can adapt to pre-existing facilities and population changes. MAS achieves substantial speedups on real datasets, achieving quasi-real-time efficiency with small performance gaps. Future research avenues encompass comprehensive simulations of human mobility patterns and meticulous modeling of facilities across diverse contexts.

References

[1] M. T. Gastner and M. E. Newman, "Optimal design of spatial distribution networks," *Physical Review E*, vol. 74, no. 1, p. 016117, 2006.

[2] Y. Xu, L. E. Olmos, S. Abbar, and M. C. González, "Deconstructing laws of accessibility and facility distribution in cities," *Science advances*, vol. 6, no. 37, p. eabb4112, 2020.

[3] T. Abbiasov, C. Heine, S. Sabouri, A. Salazar-Miranda, P. Santi, E. Glaeser, and C. Ratti, "The 15-minute city quantified using human mobility data," *Nature Human Behaviour*, 2024.

[4] Y. Zheng, Y. Lin, L. Zhao, T. Wu, D. Jin, and Y. Li, "Spatial planning of urban communities via deep reinforcement learning," *Nature Computational Science*, vol. 3, no. 9, pp. 748–762, 2023.

[5] C. Wang, C. Han, T. Guo, and M. Ding, "Solving uncapacitated p-median problem with reinforcement learning assisted by graph attention networks," *Applied Intelligence*, 2022.

[6] D. Matis and P. Tarábek, "Reinforcement learning for weighted p-median problem," in *2023 International Conference on Information and Digital Technologies (IDT)*, 2023, pp. 293–298.

[7] R. R. Mettu and C. G. Plaxton, "The online median problem," *SIAM Journal on Computing*, vol. 32, no. 3, pp. 816–832, 2003.

[8] M. Chrobak, C. Kenyon, J. Noga, and N. E. Young, "Oblivious medians via online bidding," in *Latin American Symposium on Theoretical Informatics*. Springer, 2006, pp. 311–322.

[9] M. Chrobak and M. Hurand, "Better bounds for incremental medians," *Theoretical Computer Science*, vol. 412, no. 7, pp. 594–601, 2011.

[10] M. J. Schuetz, J. K. Brubaker, and H. G. Katzgraber, "Combinatorial optimization with physics-inspired graph neural networks," *Nature Machine Intelligence*, vol. 4, no. 4, pp. 367–377, 2022.

[11] J. Toenshoff, M. Ritzert, H. Wolf, and M. Grohe, "Run-csp: unsupervised learning of message passing networks for binary constraint satisfaction problems," *CoRR, abs/1909.08387*, 2019.

[12] S. Amizadeh, S. Matusevych, and M. Weimer, "Learning to solve circuit-sat: An unsupervised differentiable approach," in *International Conference on Learning Representations*, 2018.

[13] P. Hansen and N. Mladenović, "Variable neighborhood search for the p-median," *Location Science*, vol. 5, no. 4, pp. 207–226, 1997.

[14] O. Alp, E. Erkut, and Z. Drezner, "An efficient genetic algorithm for the p-median problem," *Annals of Operations research*, vol. 122, pp. 21–42, 2003.

[15] E. Rolland, D. A. Schilling, and J. R. Current, "An efficient tabu search procedure for the p-median problem," *European Journal of Operational Research*, vol. 96, no. 2, pp. 329–342, 1997.

[16] G. Cornuejols, M. L. Fisher, and G. L. Nemhauser, "Exceptional paper—location of bank accounts to optimize float: An analytic study of exact and approximate algorithms," *Management science*, vol. 23, no. 8, pp. 789–810, 1977.

[17] Gurobi Optimization, LLC, "Gurobi Optimizer Reference Manual," 2023. [Online]. Available: https://www.gurobi.com

[18] A. A. Kuehn and M. J. Hamburger, "A heuristic program for locating warehouses," *Management science*, vol. 9, no. 4, pp. 643–666, 1963.

[19] F. Maranzana, "On the location of supply points to minimize transport costs," *Journal of the Operational Research Society*, vol. 15, no. 3, pp. 261–270, 1964.

[20] M. F. Goodchild and V. T. Noronha, *Location-allocation for small computers*. Department of Geography, University of Iowa, 1983, no. 8.

[21] H. Luo, Z. Bao, J. S. Culpepper, M. Li, and Y. Zhao, "Facility relocation search for good: When facility exposure meets user convenience," in *Proceedings of the ACM Web Conference 2023*, ser. WWW '23. New York, NY, USA: Association for Computing Machinery, 2023, pp. 3937–3947.

[22] O. Vinyals, M. Fortunato, and N. Jaitly, "Pointer networks," *Advances in neural information processing systems*, vol. 28, 2015.

[23] N. Karalias and A. Loukas, "Erdos goes neural: an unsupervised learning framework for combinatorial optimization on graphs," *Advances in Neural Information Processing Systems*, vol. 33, pp. 6659–6672, 2020.

[24] E. Khalil, H. Dai, Y. Zhang, B. Dilkina, and L. Song, "Learning combinatorial optimization algorithms over graphs," in *Advances in Neural Information Processing Systems*, vol. 30. Curran Associates, Inc., 2017.

[25] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing atari with deep reinforcement learning," *arXiv preprint arXiv:1312.5602*, 2013.

[26] H. Mao, M. Schwarzkopf, S. B. Venkatakrishnan, Z. Meng, and M. Alizadeh, "Learning scheduling algorithms for data processing clusters," in *Proceedings of the ACM special interest group on data communication*, 2019, pp. 270–288.

[27] C. Liu, R. Wang, Z. Jiang, and J. Yan, "Deep reinforcement learning of graph matching," *arXiv preprint arXiv:2012.08950*, 2020.

[28] X. Chen and Y. Tian, "Learning to perform local rewriting for combinatorial optimization," *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[29] W. Kool, H. Van Hoof, and M. Welling, "Attention, learn to solve routing problems!" *arXiv preprint arXiv:1803.08475*, 2018.

[30] P. Velickovic, G. Cucurull, A. Casanova, A. Romero, P. Lio, Y. Bengio *et al.*, "Graph attention networks," *stat*, vol. 1050, no. 20, pp. 10–48 550, 2017.

[31] R. J. Williams, "Simple statistical gradient-following algorithms for connectionist reinforcement learning," *Machine learning*, vol. 8, pp. 229–256, 1992.

[32] E. Dijkstra, "A note on two problems in connexion with graphs," *Numerische Mathematik*, vol. 1, no. 1, pp. 269–271, 1959.

[33] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.

[34] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[35] W. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," *Advances in neural information processing systems*, vol. 30, 2017.