Deep Reinforcement Learning for Multi-Period Facility Location: p_k-median Dynamic Location Problem

Changhao Miao[1,2,♠], Yuntian Zhang[1,2,♠], Tongyu Wu[1,2], Fang Deng[1,2] and Chen Chen[1,2,*]
[1]National Key Lab of Autonomous Intelligent Unmanned Systems, Beijing 100081, China
[2]School of Automation, Beijing Institute of Technology, Beijing 100081, China
[*]Corresponding Author: Chen Chen, xiaofan@bit.edu.cn
[♠]Both authors contributed equally to this research.

ABSTRACT

Facility location is a crucial aspect of spatial optimization with broad applications in urban planning. Specifically, the multi-period problem involves spatial and temporal information, making it challenging to solve. Existing research mainly focuses on heuristic methods, which depend on complex hand-crafted techniques. In this paper, we propose a novel method based on Deep Reinforcement Learning (DRL) to solve p_k-median Dynamic Location Problem (DLP-p_k). Different from classical heuristic methods, our method avoids intricate designs and considers the temporal impacts of decisions. We are the first to apply DRL to the multi-period facility location problem. Our method adopts the encoder-decoder architecture and utilizes a specialized structure to capture the temporal features across different periods. On the one hand, we introduce the Gated Recurrent Units (GRU) to encode temporal information, including dynamic coverage and dynamic costs. On the other hand, we design an attention-based decoder that allows the model to capture long-term dependencies in decision-making. Experimental results from small-size to large-size demonstrate that our method can quickly provide high-quality solutions without relying heavily on expert knowledge, offering opportunities for efficiently solving multi-period facility location problems. Additionally, our method is up to two orders of magnitude faster than the exact solver Gurobi and demonstrates great generalization abilities.

CCS CONCEPTS

• **Computing methodologies** → **Artificial intelligence**; **Machine learning**; • **Applied computing** → **Operations research**.

KEYWORDS

Facility Location, Spatial Optimization, Multi-Period, Deep Reinforcement Learning

ACM Reference Format:

Changhao Miao, Yuntian Zhang, Tongyu Wu, Fang Deng, and Chen Chen. 2024. Deep Reinforcement Learning for Multi-Period Facility Location: p_k-median Dynamic Location Problem. In *The 32nd ACM International Conference on Advances in Geographic Information Systems (SIGSPATIAL '24), October 29-November 1, 2024, Atlanta, GA, USA.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3678717.3691249

SIGSPATIAL '24, October 29–November 01, 2024, Atlanda, GA, USA

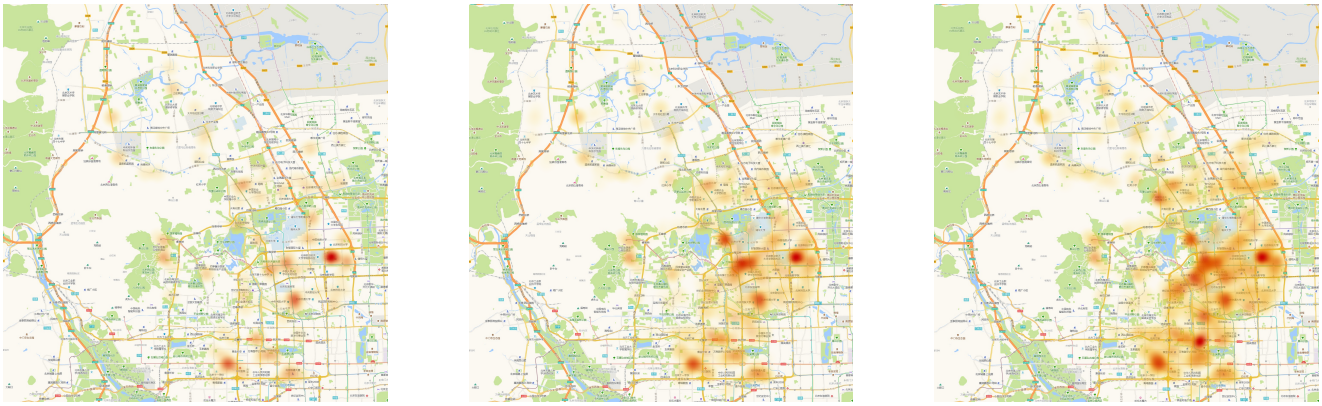
ACM ISBN 979-8-4007-1107-7/24/10
https://doi.org/10.1145/3678717.3691249

1 INTRODUCTION

Over the past decade, infrastructure within urban areas has expanded dramatically with the rapid development of urbanization. As Fig.1 illustrates, the emergency service facilities in the Haidian District of Beijing have exhibited a rapid expansion from 2005 to 2022. As the number of facilities increases, cities are facing various challenges such as traffic congestion, deteriorating infrastructure and issues related to the placement of new infrastructure. These issues pose significant challenges to the urban planning [19], which are caused by unreasonable utilization of urban space, uneven resource allocation and inadequate transportation planning. To ensure sustainable urban development, spatial optimization focuses on utilizing mathematical and computational approaches to resolve the geographic decision problems better, aiming to achieve more efficient urban planning and management.

Facility location is a classic and essential problem in spatial optimization, which has extensive applications in many real-world scenarios, including supply-chain logistics [1], logistics distribution [30], emergency management [31], and ubiquitous sensing [35]. The facility location problems can be primarily categorized into single-period and multi-period problems [13]. Single-period problems typically assume that certain parameters remain constant, even though they may be dynamic in real-world applications. Therefore, it is often beneficial to take the potential future adjustments into consideration if the parameters are predictable. Multi-period facility location problem is less-studied, but it holds significant practical importance.

Solving multi-period facility location problems is a non-trivial task, particularly when the parameters are dynamic related to various periods. In relevant studies, the methods for solving multi-period facility location problem can be primarily categorized into two main types: exact methods and heuristic methods. Exact methods [9] struggle to find the optimal solutions through modelling techniques and branch-and-bound (B&B) framework [14], while this may be computationally expensive when dealing with large-scale problems. Therefore, heuristics methods are proposed to provide satisfying solutions within acceptable time [6, 22]. However, heuristic methods heavily rely on hand-crafted designed techniques and ignore temporal correlations between periods, resulting in non-ideal performance. The decision variables are numerous and the search space is vast especially for large-scale multi-period problems, posing great challenges for both exact and heuristic methods.



(a) The year 2005 (b) The year 2012 (c) The year 2022

Figure 1: The density of emergency service facilities in the Haidian District of Beijing for the years 2005, 2012, and 2022 (from left to right), which demonstrates the rapid expansion of infrastructure in urban areas.

To address these issues, we propose an Attention-Dynamic Network (ADNet) based on Deep Reinforcement Learning (DRL) [23] and employ it to solve p_k-median dynamic location problem (DLP-p_k). Solving multi-period facility location problem can be modeled as a step-by-step construction process, which provides us with the opportunities to employ DRL. In order to effectively address both the spatial and temporal characteristics of the problem, we first reformulate DLP-p_k into a Markov decision process and adopt the encoder-decoder [2] architecture. Specifically, we introduce a specialized structure to capture the temporal features. On the one hand, we utilize the Gated Recurrent Units (GRU) to encode temporal information including dynamic coverage and dynamic costs. On the other hand, we design an attention-based decoder that allows the model to capture long-term dependencies during the selection process. The novelty and contributions of this paper are outlined as follows:

- Benefit from the ability of DRL to handle sequential decision-making, we are the first to employ DRL to solve multi-period facility location problem. Leveraging the advantages of DRL, we alleviate the heavy reliance on expert knowledge, enabling the model to obtain high-quality solutions fast.
- Considering the temporal correlations between variables, we introduce GRU to encode dynamic coverage and dynamic costs, aiming to extract the temporal characteristics of the problem. We also employ an attention-based decoder to capture long-term dependencies in decision-making.
- Experimental results demonstrate that our method achieves a gap of 0.1016%, 0.0994%, and 0.2728% for problem scales of 20, 50, and 100, where the dimension of decision variables is 1260, 12750, and 70700, respectively. Additionally, our method is up to two orders of magnitude faster than Gurobi and demonstrates great generalization abilities.

The rest of the paper is organized as follows. We briefly review the background and specify the motivation of our work in Section 2. Section 3 illustrates the mathematical formulation of DLP-p_k. ADNet is explained in detail in Section 4. Section 5 presents simulations and result analyses. Section 6 concludes this paper.

2 BACKGROUND AND MOTIVATION

In this section, we review the relevant research from two main aspects: multi-period facility location problems and deep learning for spatial optimization problems. Based on this, we illustrate the design motivation behind the proposed method.

2.1 Multi-period Facility Location Problems

In real-world practices of facility location problems, it is often necessary to consider multi-period facility deployment due to the factors such as resource constraints and long-term planning [18]. These considerations have led to the development of multi-period facility location problems.

The p-median problem [8] is one of the basic models in discrete location theory, which is proved to be NP-hard. To extend the single-period p-median problem to a multi-period one [18], a natural approach is to directly introduce time dimension, simply adding an extra dimension to the decision variables. However, such a model can be decoupled into several independent single-period ones for separate solutions. Therefore, Wesolowsky et al. [32] first included opening and closing costs for the facilities, making the model more practical. Especially, closing an open facility is unnecessary in certain scenarios such as infrastructure construction, Galvão et al. [6] did not consider facility closures and assumed that facilities would be gradually opened over the periods, which is named p_k-median dynamic location problem. Additionally, multi-period problems have been increasingly applied for real-world applications. Many scholars have considered more practical factors and proposed corresponding multi-period models for specific applications. Marín et al. [17] introduced a very general discrete covering location model that accounts for uncertainty and time-dependent aspects. Vatsa et al. [27] considered server uncertainty and proposed a capacitated multi-period MCLP for primary health centers. Furthermore, Zhang et al. [36] successfully applied the multi-period facility location problem to the location of public charging stations.

Due to the numerous decision variables and the large search space, exact methods are unable to prove the optimality for multi-period facility location problems within reasonable time. Therefore,

most aforementioned researches design heuristic methods to obtain high-quality solutions. Although heuristic methods have been widely applied in practical scenarios, these approaches heavily depend on manually crafted techniques and often fail to effectively capture the temporal characteristics of the problem.

2.2 Deep Learning for Spatial Optimization Problems

Due to the NP complexity of the mentioned problems, heuristic methods can provide feasible solutions within reasonable time, but the solutions may be unsatisfying especially for large-scale problems. With the rise of artificial intelligence, the development of deep learning has provided us with new opportunities for solving spatial optimization problems. Therefore, some scholars have made attempt to employ deep learning to solve spatial optimization problems.

Deep learning was initially applied to solve routing problems within the realm of spatial optimization [5], which has been extensively researched. Vinyals et al. [28] first utilized deep learning techniques to solve combinatorial optimization problems including classical travelling salesman problem (TSP), which laid a solid foundation for subsequent researches. Benefit from the introduction of transformer [26], Kool et al. [12] proposed an Attention Model (AM) to solve a series of routing problems, which laid foundation for further research. Furthermore, Peng et al. [20] included dynamic characteristics based on AM, enhancing the performance of model significantly. Despite the widespread applications of deep learning in routing problems, research related to location problems remains less-studied, which is promising for further studies. Specifically, Domínguez et al. [4] first tried to use Hopfield network to solve p-median problem, which can only tackle with small-scale problems. Liang et al. [15] combined a greedy algorithm and a graph convolution network to address the p-center problem. Since DRL can effectively handle sequential decision, some studies [16, 29] have started using DRL to solve facility location problems such as p-center problem, p-median problem and MCLP, which obtains promising results.

Applying deep learning to solve spatial optimization problems has demonstrated great potential, with many studies highlighting the importance of data-driven approaches for sustainable urban planning and management [21, 34]. However, the majority of research pays more attention to single-period facility location problems, while the practical meaning of multi-period location problems is also siginificant.

2.3 Motivation

Multi-period facility location is a crucial component in spatial optimization, which has widespread practical applications. Multi-period problems face challenges such as numerous decision variables and the need to consider temporal characteristics, which make them computationally difficult. It should be noted that, the optimal decision for the current period may not be optimal for the next period. To address these issues, we propose an innovative method based on DRL to solve DLP-p_k.

As mentioned above, existing heuristic methods always require complex hand-crafted techniques, posing challenges to the solution of problems. To avoid sophisticated designs and improve the efficiency, we are the first to employ DRL to solve multi-period facility location problem. Our proposed method can effectively handle sequential decision-making and provide high-quality solutions fast.

Furthermore, existing methods for multi-period problem seldom consider the temporal features during decision-making including dynamic coverage, and dynamic costs. Motivated by these, we aim to jointly consider spatial and temporal characteristics by an encoder-decoder architecture, where we introduce the specialized structure to capture the temporal relationships behind sequential decision-making.

3 PROBLEM FORMULATION

In this section, we introduce the mathematical formulation of DLP-p_k. The aim of this problem is to minimize the sum of transportation and installation cost. In the period k, the problem requires operating p_k facility points, considering only the opening of facilities and not their closure among the periods.

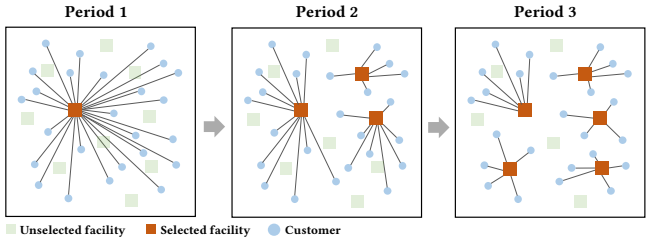


Figure 2: The illustration of a p_k-median dynamic location problem instance with 3 periods, where p_k=1, 3, and 5.

Due to the dynamic nature of transportation and installation cost, the number of decision variables in multi-period problems is very large. Additionally, in our decision-making process, we need to consider not only the static information of the candidate points but also their temporal relationships and characteristics. Fig. 2 illustrates an example with 10 candidate points and 3 periods, where the number of facilities operated in each period is 1, 3, and 5, respectively. It is clear that the number of facilities operated and the allocation relationships between facilities and customers change with each period.

Consider a set of points J, each point $j \in J$ has a demand that needs to be satisfied by an operating facility. These facilities operate over a finite multi-period planning horizon $|K|$, with K representing the set of periods. Consider a subset of points $I \subseteq J$, where facilities can be located. For each period $k \in K$, a specified number p_k of facilities must be operating. Therefore, the problem can be formulated as follows:

$$\min_{\mathbf{x},\mathbf{y}} \quad \sum_{k\in K}\sum_{i\in I}\sum_{j\in J} c_{ijk}x_{ijk} + \sum_{k\in K}\sum_{i\in I} g_{ik}y_{ik} \tag{1}$$

$$s.t. \quad \sum_{i\in I} x_{ijk} = 1, \quad k \in K, j \in J, \tag{1a}$$

$$\sum_{i\in I} x_{iik} = p_k, \quad k \in K, \tag{1b}$$

$$\sum_{j\in J} x_{ijk} \le |J| \cdot x_{iik}, \quad k \in K, i \in I, \tag{1c}$$

$$y_{ik} \ge x_{iik} - x_{ii,k-1}, \quad k \in K\backslash\{1\}, i \in I, \tag{1d}$$

$$x_{ijk}, y_{ik} \in \{0,1\}, \quad k \in K, i \in I, j \in J. \tag{1e}$$

where x_{ijk} and y_{ik} are decision variables, $x_{ijk} = 1$ is a binary variable that equals 1 if customer j is supplied by facility i in period k, and 0 otherwise; $y_{ik} = 1$ is a binary variable that equals 1 if facility i is opened in period k, and 0 otherwise; c_{ijk} represents the transportation cost of supplying customer j in period k; g_{ik} represents the installation cost of facility i in period k. Note that $x_{iik} = 1$ indicates that facility i is operating in period k, regardless of whether it is opened in this period. Additionally, we assume that the set of customers J and the set of candidate facility points I are the same, i.e., $I = J$. Since x_{ijk} and y_{ik} are decision variables, we can calculate the dimension of decision variables by $D = |I| \times |J| \times |K| + |I| \times |K|$, which reflects the complexity of problem. In this formulation, Objective (1) aims to minimize the sum of transportation and installation costs across all periods; Constraint (1a) guarantees that the demand of each customer is satisfied by exactly one facility in each period; Constraint (1b) limits the number of facilities operating in period k to p_k; Constraint (1c) indicates that customers can only be supplied by facilities that are operated in period k; Constraint (1d) ensures $y_{ik} = 1$ if the facility i is opened in period k.

4 PROPOSED ALGORITHMS

In this section, we reformulate and represent the mathematical model as a Markov Decision Process (MDP). We also illustrate the details of our ADNet, which has specialized structure for better capturing dynamic characteristics. The model is trained using REINFORCE algorithm with a rollout baseline. The pipeline of our proposed algorithm is illustrated in Fig. 3.

4.1 Markov Decision Process

To apply DRL, we employ the constructive method to model DLP-p_k, thus we essentially transform it into a form of MDP. To model the MDP, we introduce a five-tuple representation, which includes state, action, transition, reward, and discount factor. Assuming a total of p nodes need to be selected and one node is chosen at each step until p nodes are selected. The five-tuple at step t is defined as follows:

State: The current solution set is defined as the state s_t, where s_t comprises all selected facility points up to step t. Specifically, s_t remains a partial solution until step $t = p$, while s_0 is an empty set.

Action: During the solution construction, each step requires selecting a node π_t from candidate points to serve as the facility point. Therefore, the action a_t is defined as π_t.

Transition: The probability distribution representing the transition from state s_t to the next state s_{t+1} after taking action a_t.

Reward: The immediate reward r_t received when state s_t transitions to state s_{t+1} after taking action a_t, which determines the optimization direction of the policy.

Discount factor: The discount factor γ is a value between 0 to 1, which decides the degree of importance given to future rewards. Specifically, the final reward is directly obtained by the objective function.

Since the number of facilities operating in each period is known in advance, the solution is a permutation of selected points. Therefore, the solution can be denoted by $\pi = \{\pi_1, ...\pi_p, p \le n\}$, where p denotes the total number of facilities need to be operated at the last period and n is the total number of candidate points. For a given instance s, ADNet defines a policy $p_\theta(\pi|s)$ for selecting node π at each step, where θ denotes the parameters of ADNet.

$$p_\theta(\pi|s) = \prod_{t=1}^{p} p_\theta(\pi_t|s, \pi_{1\sim t-1}), p \le n \tag{2}$$

To train ADNet by REINFORCE algorithm, we have to define loss $L(\theta|s) = E_{p_{\theta(\pi|s)}}[L(\pi)]$, which indicates the expectation of cost $L(\pi)$. Specifically, the cost function $L(\pi)$ for DLP-p_k can be defined as follows

$$L(\pi) = \sum_{t\in T}\sum_{i\in I}\sum_{j\in J} c_{ijt}x_{ijt} + \sum_{t\in T}\sum_{i\in I} g_{it}z'_{it} \tag{3}$$

The REINFORCE algorithm iteratively update parameters using gradient descent method, aiming to directly minimize the cost function.

4.2 ADNet

The ADNet adopts an encoder-decoder architecture [26] to better handle the characteristics of the problem. The encoder converts the original features of all input nodes into hidden embeddings. The decoder then generates a sequence of input nodes one node at a time. Therefore, the decoder takes theses embeddings as input and outputs the probability of selecting the next node. The ADNet model is shown in Fig. 4.

4.2.1 Encoder. The encoder takes original features as input and produces a representation for each node, which generates the *key* of each node. Specifically, the key can be partitioned into: 1) spatial embeddings obtained through the *attention layer* that represent the structural patterns within the problem, and 2) temporal embeddings capturing the dynamic patterns of the problem.

Spatial Embeddings: The spatial embeddings represent the spatial information including coordinates and the installation cost of first period for each node. A linear mapping is first performed to obtain the node embedding, followed N layers of attention layers, finally producing the spatial embeddings for each node. Specifically, each attention layer [12] comprises two sublayers: a Multi-Head Attention (MHA) layer [26] for message passing between nodes and a feed-forward layer. Each sublayer is connected to a skip-connection and a normalized layer BN. Therefore, the output of each sublayer can be expressed as $BN(x + sublayer(x))$. The detail of encoder can be illustrated as follows:

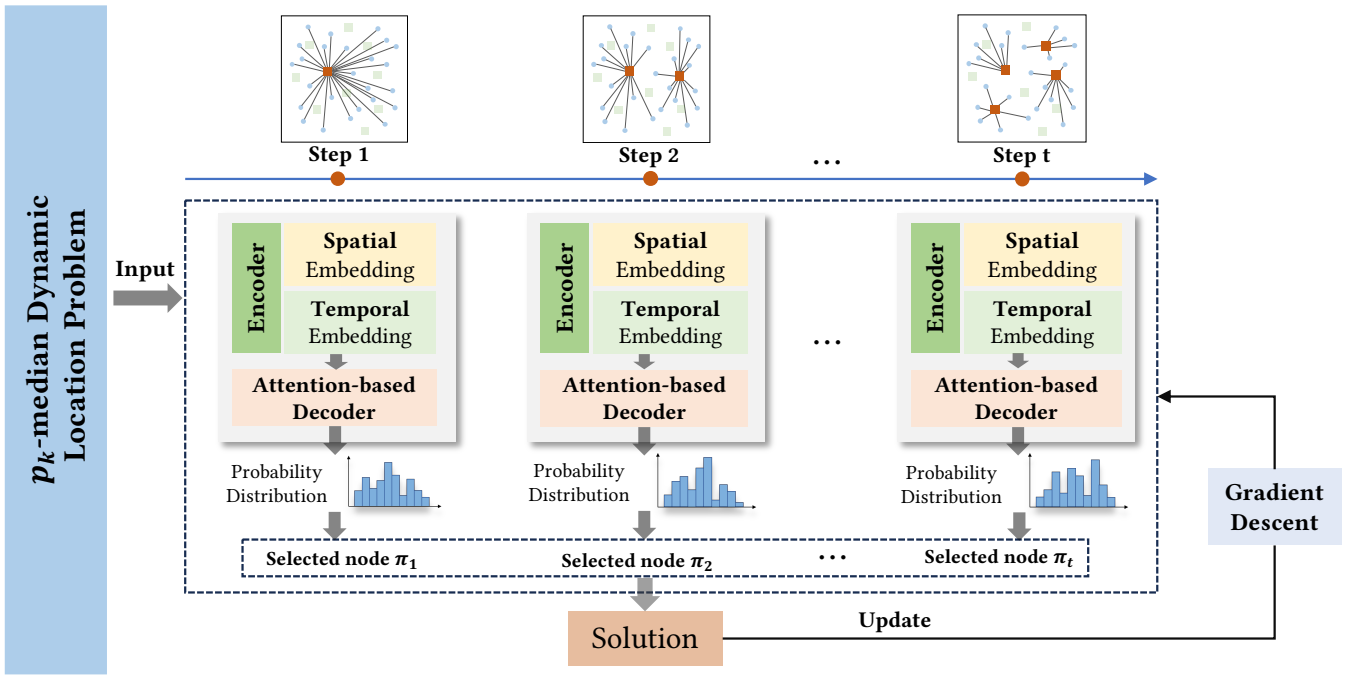


Figure 3: The pipeline of our proposed algorithm.

First, we map the spatial information from a d_x-dimensional vector to a d_h-dimensional vector. For each node, we take the coordinates and the installation cost of first period as spatial information ($d_x = 3$), and we set the hidden dimension to 128 ($d_h = 128$). Therefore, the initial node embeddings f_i^0 can be calculated as follows:

$$f_i^0 = W^x x_i + b^x \tag{4}$$

where x_i indicates the spatial information of the i-th node, W^x and b^x are learnable parameters. After passing the initial node embeddings $f_i^{(0)}$ through N attention layers, we can obtain the final output embeddings. Specifically, the output embeddings from the l-th layer are given by:

$$\hat{f} = BN(f^{(l-1)} + MHA^l(f^{(l-1)})) \tag{5}$$

$$f^l = BN(\hat{f} + FF^l(\hat{f})) \tag{6}$$

where BN, FF, MHA denote batch normalization layer, feed-forward layer and multi-head attention layer, respectively. The details of the above layers are the same as the study [12]. After passing through N attention layers, the final spatial embedding can be denoted as f^N, a d_h-dimensional vector of the i-th node.

Temporal Embeddings: The spatial embeddings f^N can serve as the key directly for routing problems [12]. However, in the context of DLP-p_k, the states of nodes are not static during the decoding process. Therefore, we have to take some temporal factors into the consideration either for location problems, including dynamic coverage and dynamic installation cost.

First, once a facility is opened near an already opened point, the possibility of opening a facility at that point may decrease. It is essential to incorporate the concept of dynamic coverage into

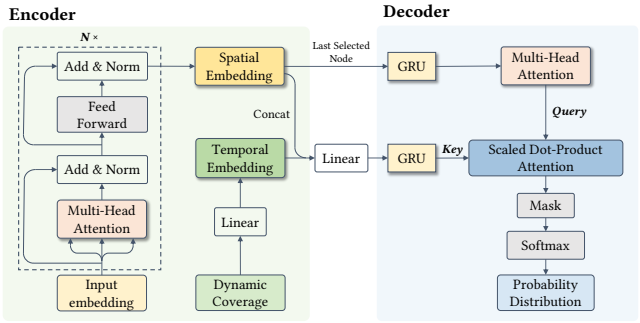


Figure 4: The architecture of our ADNet.

the construction of the key. To address this issue, we define an indicator function δ_i to represent the coverage status for each node i, $i = 1, 2, ...n$. We assume each facility point has a maximum service radius r and use π_t to denote the point at the step t, so that the points under the service radius of point π_t are denoted as $C_r(\pi_t)$. At each step, the indicator function δ_i will be dynamically updated as follows:

$$\delta_i = \begin{cases} 0, & i \in C_r(\pi_t) \\ 1, & otherwise \end{cases} \tag{7}$$

According to the Equation (7), the indicator function δ_i includes the coverage information for each point, which serves as a guide for selecting the next facility point. For example, the points closer to those already selected facility points may have a lower probability of selection.

Additionally, the installation cost varies with each period for DLP-p_k, which should also be considered together with dynamic coverage. However, in certain instances, neighboring points to the already selected facility points may also be good choices. Therefore, we let ADNet learn how to utilize these dynamic characteristics through a series of transformation including linear projection and Gate Recurrent Unit (GRU) [3]:

$$H_i = f_{GRU}((\delta_i||f_{it}) \cdot W^d) \tag{8}$$

where $\cdot||\cdot$ denotes the concatenation between two vectors, W^d is a learnable parameter, H_i is the final temporal embeddings, and f_{GRU} indicates the function of GRU. Specifically, We adopt the structure of GRU to capture the temporal relationships among dynamic characteristics. Therefore, we can construct the *key* for each node as follows:

$$k_i = W^k f_i^N H_i, \quad i = 1, 2, ..., n \tag{9}$$

where W^k is the learnable parameter, and k_i denotes the *key* of node i, which is taken as the input to the decoder. The model can learn a better policy by combining both spatial and temporal embeddings.

4.2.2 Decoder. To decode the embeddings provided by the encoder, we propose a learnable decoder based on the attention mechanism, thereby selecting facility points and constructing solutions. The decoder comprises two components: 1) a GRU for constructing the *query* and 2) an attention mechanism [26] for computing the selection probability of each facility.

Construction of query: For the multi-period location problem, the construction of the solution is a sequential decision-making process, so it is necessary to take temporal information into account. We employ GRU to construct the *query*, which can handle such sequential information. We initialize the hidden state of the GRU using the mean of the encoder embeddings, calculated as $\overline{\mathbf{h}} = \frac{1}{n}\sum_{i=1}^{n} \mathbf{h_i}^{(N)}$, and denote this as $d_0 = \overline{h}$. At the first step $t = 1$, a learnable parameter $\mathbf{v}_1$ is used as the input to the GRU. For subsequent steps $t > 1$, the input to the GRU is the node embedding $\mathbf{h}^{(N)}_{\pi_{t-1}}$ of the previously selected node π_{t-1}. At the step t, the input I_t can be denoted as follows:

$$I_t = \begin{cases} \mathbf{v}_1, & t = 1 \\ \mathbf{h}^{(N)}_{\pi_{t-1}}, & t > 1 \end{cases} \tag{10}$$

Since the embedding obtained by GRU contains information about the previously selected points $\pi_{1\sim t-1}$, the embedding o_t can be used directly as the *query*. However, the relationships between the previously selected points $\pi_{1\sim t-1}$ should also not be overlooked. To enhance the processing of o_t, we apply an attention mechanism as follows:

$$q_t = Attention(o_t, K_1, V_1) = softmax(\frac{o_t K_1^T}{\sqrt{d_k}})V_1 \tag{11}$$

where K_1 and V_1 are the linear projections of spatial embeddings $\mathbf{h}^{(N)}$ from the encoder, $\sqrt{d_k} = \sqrt{\frac{d_h}{M}}$ is used as the scaling factor, d_h denotes the hidden dimension, and M indicates the number of heads in the MHA. By introducing the attention mechanism, the *query* contains richer information, which aids subsequent decision-making and improves the effectiveness of the decisions.

Calculation of Attention: At each step of decoding, the probability of selection for all points $i = 1, 2, ..., n$ are computed based on the pair of *query - key* $(\mathbf{q_t}, \mathbf{k_i})$ through scaled dot-product attention:

$$u_i^t = \frac{\mathbf{q_t^T k_i}}{\sqrt{d_k}} \tag{12}$$

where $\sqrt{d_k} = \sqrt{\frac{d_h}{M}}$ is used as the scaling factor, d_h denotes the hidden dimension, and M indicates the number of heads in the MHA. Specifically, the *query* q_t is produced based on GRU and attention mechanism according to the Equation (11), while the *key* k_i is calculated as Equation (9). Furthermore, the same point cannot be selected more than once, thus the u_i^t should be masked to include the coverage information as follows:

$$\overline{u}_i^t = \begin{cases} -\infty, & i \in \pi_{1\sim t-1} \\ \dfrac{\mathbf{q_t^T k_i}}{\sqrt{d_k}}, & otherwise \end{cases} \tag{13}$$

After applying the softmax function, the probability of already selected points with $\overline{u}_i^t = -\infty$ will be zero. Therefore, we utilize the softmax function to scale $\overline{u}_i^t$, which is calculated as follows:

$$p_i = \frac{e^{\overline{u}_i^t}}{\sum_{j=1}^{n} e^{\overline{u}_j^t}}, \quad i = 1, 2, ..., n \tag{14}$$

The probability is computed using the attention mechanism, leveraging the *query* q_t generated by the GRU and the *key* produced by the encoder. For the decoding strategy, we offer two methods: 1) greedy decoding, and 2) sampling decoding. At each time step t, greedy decoding means opening a facility at the point with the highest probability p, while sampling decoding samples from the current probability distribution to select the point.

4.3 Training of ADNet

Given an instance s, Equation (1) defines the model with learnable parameter θ, which produces the probability distribution $p_\theta(\pi|s)$ at each step. By employing the decoding strategy, we can obtain the solution $\pi|s$ based on $p_\theta(\pi|s)$. Our goal is to minimize the objective function, which is calculated by the sum of transportation and installation costs. Therefore, we utilize REINFORCE algorithm to train our ADNet and optimize the policy.

To reduce the variance, the parameters of our model are updated through gradient descent with the baseline $b(s)$. Specifically, the gradient can be calculated as follows [33]:

$$\nabla_{\boldsymbol{\theta}} \mathcal{L}(\theta|s) = \boldsymbol{E}_{p_\theta(\pi|s)} \left[\nabla_{\boldsymbol{\theta}} \log p_{\boldsymbol{\theta}} \left(\boldsymbol{\pi}|s\right) \left(L\left(\boldsymbol{\pi}\right) - b(s)\right)\right] \tag{15}$$

where $b(s)$ is the baseline, representing the average performance of the policy, thus a policy will be encouraged when it outperforms the baseline.

Algorithm 1 describes the training process using REINFORCE, which employs a greedy rollout baseline [12] and the Adam optimizer [10]. The baseline $b(s)$ denotes the objective function of the solution provided by the baseline policy p_{θ^*} greedily. Specifically, the baseline policy is the best-performing model up to that point in training and is updated at the end of each epoch.

Algorithm 1: The training algorithm based on REINFORCE

Input : training set $\mathcal{S}$, batch size B, number of epochs N_{epoch}, number of steps per epoch N_{steps};

Initialize the parameters of model for the current policy $\boldsymbol{\theta}$ and the baseline policy $\boldsymbol{\theta}^* \leftarrow \boldsymbol{\theta}$;
for $epoch \leftarrow 1 : N_{epoch}$ **do**
  for $step \leftarrow 1 : N_{step}$ **do**
    $s_i \leftarrow \text{RandomInstance}(\mathcal{S})$ for $i \in \{1, \ldots, B\}$
    $\boldsymbol{\pi}_i \leftarrow p_{\boldsymbol{\theta}}(s_i)$. Execute the policy by sampling
    $\boldsymbol{\pi}_i^* \leftarrow p_{\boldsymbol{\theta}^*}(s_i)$. Execute the policy greedily
    $\nabla \mathcal{L} \leftarrow \sum_{i=1}^{B} \left(L\left(\boldsymbol{\pi}_i\right) - L\left(\boldsymbol{\pi}_i^*\right)\right) \nabla_{\boldsymbol{\theta}} \log p_{\boldsymbol{\theta}}\left(\boldsymbol{\pi}_i\right)$
    Update $\boldsymbol{\theta}$ based on $\nabla \mathcal{L}$ by Adam
  end
  if $p_{\boldsymbol{\theta}}$ *outperforms* $p_{\boldsymbol{\theta}^*}$ **then**
    $\boldsymbol{\theta}^* \leftarrow \boldsymbol{\theta}$
  end
end

5 EXPERIMENTAL STUDY

In this section, we conduct extensive experiments to evaluate the performance of our proposed algorithm, and compare the proposed ADNet with exact solvers, heuristics and metaheuristics. The code is available online[1].

5.1 Experimental Setting

First, we introduce the experimental setting including experimental scenarios, comparison algorithms, hyperparameter configurations, and implementation.

Table 1: Experimental Scenarios.

Settings	Small	Medium	Large
Graph size	20	50	100
Problem size	$20 \times 20 \times 3$	$50 \times 50 \times 5$	$100 \times 100 \times 7$
Dimension of decision variables	1260	12750	70700
Number of periods	3	5	7
Medians	[2, 3, 4]	[2, 3, 4, 6, 8]	[2, 4, 7, 9, 10, 13, 15]
Discount rate	0.12-0.2	0.12-0.2	0.12-0.2
Maximum service radius	0.32	0.24	0.16

Experimental Scenarios: To evaluate the performance of our method, we employ the proposed method to solve randomly generated DLP-p_k instances with various scales. The experimental scenarios are shown in Table 1. Specifically, we assume three different problem sizes, each containing 20, 50, and 100 candidate points, with periods of 3, 5, and 7, respectively. The dimension of decision variables for each scale is 1260, 12750, and 70700, which is calculated by $D = |I| \times |J| \times |K| + |I| \times |K|$ as Section 3. All candidate points are generated randomly within a unit square $[0, 1]$.

As to the dynamic factors including transportation and installation cost, we follow the setting as the study [6, 25]. The transportation cost is kept constant along the planning horizon, which is calculated by the Euclidean distance between candidate points. Additionally, we randomly generate a fixed cost for each point at the first period within the range of 2 to 4, and each point has a discount rate of 12% to 20% for periods $k = 2, 3, ..., |K|$. To represent the dynamic coverage information as illustrated in Section 4.2.1, we assume each facility point has a maximum service radius r of 0.32, 0.24, and 0.16 for the problems with 20, 50, and 100 points, respectively.

Hyperparameter Configurations: The experimental setup involves a range of hyperparameters as detailed in the Table 3. To ensure fair comparison, all hyperparameters about training and model architecture were kept consistent with the baseline methods Attention Model (AM) [12]. The training phase spans 250 epochs with a total of 128,000 instances, and the batch size for training is set to 512. For evaluation, 12,800 instances are used, with a batch size of 1280. The Adam optimizer is employed, with a learning rate of 1×10^{-4}. The model utilizes a hidden dimension of 128 and consists of 3 encoder layers in total. This configuration ensures a comprehensive evaluation of the algorithm's performance across different scenarios.

Comparison Algorithms: To evaluate the effectiveness of our proposed method ADNet, we take several representative methods as the baseline algorithms. To provide more comprehensive results, we compare ADNet with various types of methods, including exact solvers, state-of-the-art deep learning methods, classical heuristic methods, and metaheuristics. Specifically, the baseline algorithms selected for comparison are listed as follows:

1) The exact solver *Gurobi* [7].

2) The baseline of deep learning, *Attention Model* (AM) [12].

3) The interchange heuristic method called *Teitz and Bart* (Teitz-Bart) algorithm [24].

4) The classical metaheuristic, *Simulated Annealing* (SA) algorithm [11].

To evaluate the performance of our ADNet, we consider two performance metrics: *execution time* and *optimal gap*. Both of the metrics are recorded as the average of all instances for each scenario. Specifically, the *optimal gap* is defined as (*best* - *optimal*)/*optimal*, where *best* and *optimal* denote the best and optimal objective value, respectively.

To obtain the optimal gap of various methods, we utilize the exact solver gurobi [7] to calculate the optimal solution. The AM model [12] is the first deep learning model who employs the attention mechanism to solve routing problems. AM is one of the most representative deep learning model and is widely used as the baseline. We also compare our ADNet with the heuristic method Teitz-Bart [24], which is first proposed for solving p-Median Problem. Additionally, classical metaheuristic is also included as the competitor such as SA [11] algorithm. It should be noticed that AM and Teitz-Bart cannot be used to solve DLP-p_k, so that we apply necessary modifications to adapt them to the current problem.

All experiments are conducted on the machine with two GTX 3090 GPU and Intel(R) Xeon(R) Silver 4210R CPU @ 2.40GHz, and the code is implemented in Python.

5.2 Comparison Analysis

Our ADNet is compared with all the comparison algorithms mentioned before for three different problem scales. We evaluate the performance through the metrics including objective value, optimal

[1] https://github.com/JackeyMiao/ADNet

Table 2: The comparison results for ADNet with different scales.

	N = 20, D = 1260			N = 50, D = 12750			N = 100, D = 70700		
	Obj.	Gap	Time (s)	Obj.	Gap	Time (s)	Obj.	Gap	Time (s)
Gurobi	21.1109	0.0000%	0.1030	60.7017	0.0000%	1.5084	123.9154	0.0000%	15.0527
Teitz-Bart	24.4385	15.7621%	0.0347	74.4875	22.7107%	0.2117	155.2195	25.2625%	1.1278
SA	21.7965	3.2475%	0.1616	65.3274	7.6204%	0.7805	137.1386	10.6711%	3.2026
AM-greedy	22.8983	8.4666%	0.0094	64.9258	6.9588%	0.0157	134.5530	8.5845%	0.0239
AM-sample128	22.1029	4.6985%	0.0107	63.3512	4.3647%	0.0201	132.5360	6.9568%	0.0307
AM-sample1280	21.8976	3.7263%	0.0104	62.9939	3.7762%	0.0202	132.0334	6.5512%	0.0360
ADNet-greedy	21.2333	0.5794%	0.0180	61.0154	0.5167%	0.0334	125.1168	0.9695%	0.0337
ADNet-sample128	21.1470	0.1708%	0.0243	60.7960	0.1553%	0.0386	124.3984	0.3897%	0.0403
ADNet-sample1280	**21.1324**	**0.1016%**	**0.0263**	**60.7621**	**0.0994%**	**0.0620**	**124.2534**	**0.2728%**	**0.0930**

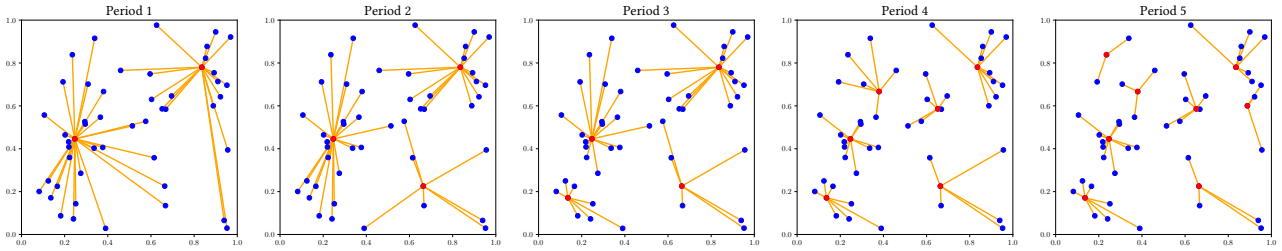


Figure 5: Visualizing the solution produced by ADNet for DLP-p_k on $N = 50$, $p_k = [2, 3, 4, 6, 8]$.

Table 3: Hyperparameter configurations.

Hyperparameters	Value
Number of epoch	250
Number of instances per training	128,000
Batch size of training	512
Number of instances per evaluation	12,800
Batch size of evaluation	1280
Hidden dimension	128
Number of Encoder layers	3
Optimizer	Adam
Learning rate	1e-4

gap, and execution time, where gap is calculated with respect to the optimal solution obtained by exact solver Gurobi.

For each problem size, the results are recorded in average by 1,000 randomly generated instances. Additionally, we test and compare the results about different decoding strategies, including greedy and sampling strategies. For the sampling strategy, experiments are conducted with 128 and 1280 samples, which are widely used in experiments involving learning-based methods.

As illustrated in Table 2, our ADNet achieves the optimal gap of 0.1016%, 0.0994%, and 0.2728% for the scale of 20, 50, and 100, respectively, which outperforms all the results of heuristic methods and learning-based methods.

Comparison to exact solver: Modern commercial solver Gurobi can obstain and prove the optimal solution for moderate-sized instances. However, Gurobi requires significant computation time as the problem scale increases. Especially when the problem scale N comes to 200, 300, 400, and 500, gurobi cannot provide an optimal solution within the acceptable time, which will be discussed in Section 5.3. Although our ADNet cannot always provide the optimal solution, it can provide satisfactory solutions within a very short time. Even for a problem scale of 100, the average gap value of the solutions provided by ADNet does not exceed 0.28%.

Comparison to (meta)heuristic methods: The performance of Teitz-Bart and SA depends on the parameters, where longer execution times lead to smaller gaps. In this study, we present the results within an acceptable execution time. Due to the large search space in multi-period location problems and the need to consider dynamic characteristics over time, (meta)heuristic algorithms are unable to provide satisfactory solutions rapidly.

Comparison to learning-based methods: We compared the results of ADNet and AM across different problem sizes and tested the impact of various decoding strategies. Since the sampling strategy samples the probability distribution multiple times to obtain the optimal value, it is expected to outperform the greedy strategy. The experimental results indicate that even with the greedy strategy, ADNet still surpasses AM with 1280 samples, validating the effectiveness of our method. Although AM has a shorter execution time compared to ADNet, the trade-off between the time and accuracy is acceptable.

Visualization of the solution: For better illustration, we visualize a solution generated by ADNet in Fig. 5, which includes five periods and 50 candidate points in total. At each period, 2, 3, 4, 6, and 8 facility points need to be operated, respectively. Blue points

Table 4: The transfer study for DLP-p_k of larger scales.

	N = 200, D=281400			N = 300, D=632100			N = 400, D=1122800			N = 500, D=1753500		
	Obj.	Gap*	Time (s)	Obj.	Gap*	Time (s)	Obj.	Gap*	Time (s)	Obj.	Gap*	Time (s)
Gurobi	236.9436	/	349.2178	405.0616	/	591.0001	857.0402	/	600	1153.9332	/	600
Teitz-Bart	291.4445	23.0016%	4.6253	423.9251	4.6570%	12.5227	554.6339	-35.2850%	23.8306	691.4827	-40.0760%	38.5215
SA	259.3565	9.4592%	6.6296	383.4852	-5.3267%	9.8654	507.9941	-40.7269%	13.3137	627.8539	-45.5901%	16.3294
AM-greedy	248.6893	4.9572%	0.0388	363.2037	-10.3337%	0.0388	477.3739	-44.2997%	0.0387	591.8513	-48.7101%	0.0453
AM-sample128	246.3471	3.9687%	0.0441	360.5228	-10.9956%	0.0445	474.5660	-44.6273%	0.0455	588.7814	-48.9761%	0.0571
AM-sample1280	245.5997	3.6532%	0.0632	359.6363	-11.2144%	0.0787	473.7127	-44.7269%	0.0948	587.7136	-49.0687%	0.1134
ADNet-greedy	237.4019	0.1934%	0.0396	350.9106	-13.3686%	0.0546	465.2448	-45.7149%	0.0523	578.2392	-49.8897%	0.0413
ADNet-sample128	235.9404	-0.4234%	0.0474	348.4251	-13.9822%	0.0527	461.5852	-46.1419%	0.0623	574.1844	-50.2411%	0.0662
ADNet-sample1280	**235.5656**	**-0.5816%**	**0.1511**	**347.8713**	**-14.1189%**	**0.2113**	**460.7433**	**-46.2402%**	**0.2677**	**573.0270**	**-50.3414%**	**0.3319**

* Since Gurobi can not find the optimal solution for some instances in the dataset with the limit of time, we use "/" to represent Gurobi's gap value for those cases. The gap values for other algorithms are still calculated based on Gurobi as the benchmark, which may result in negative values.

Table 5: The transfer study for different distribution of p_k.

	N = 20, D = 1260				N = 50, D = 12750				N = 100, D = 70700			
	$p_k = [2, 4, 4]$		$p_k = [3, 3, 4]$		$p_k = [2, 4, 6, 6, 8]$		$p_k = [3, 4, 6, 7, 8]$		$p_k = [2, 2, 4, 8, 8, 10, 15]$		$p_k = [6, 8, 9, 10, 12, 14, 15]$	
	Gap	Time (s)	Gap	Time (s)	Gap	Time (s)	Gap	Time (s)	Gap	Time (s)	Gap	Time (s)
Gurobi	0.0000%	0.1101	0.0000%	0.1604	0.0000%	1.4122	0.0000%	1.9311	0.0000%	13.6279	0.0000%	13.2314
Teitz-Bart	13.3949%	0.0300	14.2173%	0.0305	21.4848%	0.2115	20.6277%	0.2211	28.1783%	1.1157	22.8267%	1.1351
SA	1.8153%	0.1641	2.8721%	0.1668	7.9680%	0.8401	7.8825%	0.8811	10.4303%	2.7922	10.3323%	3.8539
AM-greedy	8.9974%	0.0100	9.7478%	0.0098	8.1875%	0.0175	9.1354%	0.0152	7.5527%	0.0235	11.7430%	0.0241
AM-sample128	4.8685%	0.0106	5.4786%	0.0102	5.1260%	0.0180	5.7555%	0.0189	6.2243%	0.0326	9.2473%	0.0310
AM-sample1280	3.8177%	0.0105	4.4094%	0.0102	4.3658%	0.0181	4.9690%	0.0173	5.8769%	0.0359	8.6088%	0.0356
ADNet-greedy	0.8121%	0.0207	1.0942%	0.0241	0.8678%	0.0461	1.0862%	0.0417	0.9387%	0.0330	1.8531%	0.0340
ADNet-sample128	0.2511%	0.0195	0.5024%	0.0219	0.3367%	0.0454	0.4995%	0.0477	0.3508%	0.0409	0.8763%	0.0436
ADNet-sample1280	**0.1571%**	**0.0291**	**0.3752%**	**0.0297**	**0.2329%**	**0.0587**	**0.3625%**	**0.0672**	**0.2357%**	**0.0954**	**0.6677%**	**0.0949**

represent customer locations, red points represent opened facility locations, and the lines represent the allocation relationships between facilities and customers.

5.3 Transfer Study

In Section 5.2, we train for the problems with various scales, with fixed distributions of p_k for each size. However, in real-world applications, the generalization performance of the model is a key factor to consider. Therefore, we consider the evaluation under two partitions: 1) larger scales and 2) different distributions of p_k.

Transfer to DLP-p_k of larger scales: The approach based on deep learning offers the advantage of adaptively handling large-scale problems. We train the model with the scale of 100 nodes and evaluate its performance on instances containing 200, 300, 400 and 500 nodes, where p_k remains to be the same with 100 nodes $p_k = [2, 4, 7, 9, 10, 13, 15]$. To obtain the optimal solution, we use Gurobi with a time limit of 600 seconds per instance. To save time, we solve 100 randomly generated instances for each problem size.

As illustrated in Table 4, ADNet remains to be competitive compared to other baseline algorithms. Especially for Gurobi, it often cannot provide an optimal solution for larger instances within the time of limit. However, ADNet can offer a solution that outperforms Gurobi in a significantly shorter time.

Transfer to different distribution of p_k: In real-world applications, the distribution of p_k is not fixed. Therefore, when testing the effectiveness of our ADNet, it is important to evaluate it against different distributions of p_k. Specifically, we conduct similar experiments on three problem scales: 20, 50, and 100, respectively. We select p_k distributions different from the training dataset and compare our results with other baseline algorithms.

As shown in Table 5, ADNet demonstrates excellent generalization performance when encountering different distributions of p_k, consistently maintaining a dominant position across various baseline algorithms.

5.4 Real-World Scenario Analysis

In this section, we utilize ADNet to address a significant urban planning issue by optimizing the location of infrastructures in Haidian District, Beijing, China. The aim is to minimize the combined transportation and installation costs over multiple periods while ensuring the efficient response of essential services.

The dataset consists of 4,749 residential housing units, from which we sample 100 points for better visualization. All data in the dataset are collected from real-world sources and undergo essential preprocessing including coordinate transformation and normalization, before the analysis. Specifically, we assume there are three periods in total and set $p_k = [2, 3, 4]$. To provide a clear comparison of the differences, we employ Gurobi, ADNet, and SA to solve the instance, as shown in Fig. 6 and Fig. 7. Both Gurobi and ADNet solve the instance optimally, while SA only provides a near-optimal solution. For the Period 1, the optimized solution provided by Gurobi and ADNet are more rational in terms of the balanced allocation for customers. On the contrary, the solution produced by SA may result in higher transportation cost for certain pairs of facility and customer. The visualized phenomenon emphasizes the strategic importance of multi-period dynamic facility location, which may significantly impact the efficiency of subsequent logistics decisions.

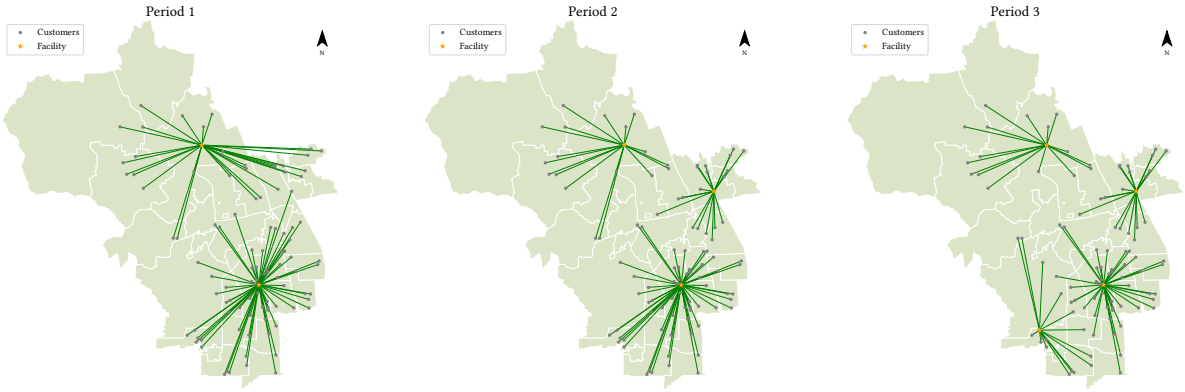


Figure 6: Visualization of the solution achieved by Gurobi and ADNet (both solved optimally) for the real-world instance.

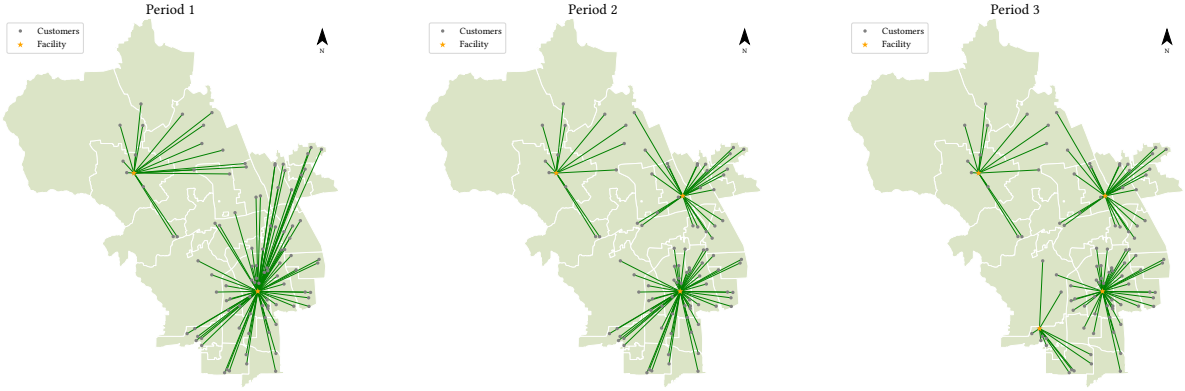


Figure 7: Visualization of the solution achieved by SA for the real-world instance.

This straightforward example illustratively demonstrates the flexibility of ADNet model in tackling real-world problems, emphasizing its significant potential for future research and practical applications.

6 CONCLUSION

In this paper, we introduce a novel method based on DRL for DLP-p_k. To address the temporal information of multi-period problems, we specifically design a structure to capture dynamic characteristics, thereby enhancing the performance of model. Additionally, we are the first to apply DRL techniques to solve multi-period location problems. Experiment results demonstrate that the proposed method exhibits excellent performance and generalization abilities.

In the future, we will attempt to extend ADNet to some variants of the multi-period facility location problem, such as considering facility opening and closure simultaneously as well as facility service capacity constraints. We will also design more effective model structures based on the characteristics of the problem to improve the algorithm performance. Additionally, we will take the uncertainty factor in real-world practices into consideration to design a robust and adaptive learning systems.

ACKNOWLEDGMENTS

This work was supported by the National Key Research and Development Program of China No.2022ZD0119703; in part by the National Natural Science Foundations of China (NSFC) under Grant 62273044 and 62022015; in part by the National Natural Science Foundation of China National Science Fund for Distinguished Young Scholars under Grant 62025301; in part by the National Natural Science Foundation of China Basic Science Center Program under Grant 62088101.

REFERENCES

[1] Abhranil Chatterjee, Janit Anjaria, Sourav Roy, Arnab Ganguli, and Krishanu Seal. 2016. SAGEL: smart address geocoding engine for supply-chain logistics. In *Proceedings of the 24th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*. 1–10.

[2] Kyunghyun Cho, Bart van Merrienboer, Çaglar Gülçehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation. In *Conference on Empirical Methods in Natural Language Processing*. 1724—-1734.

[3] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555* (2014).

[4] Enrique Domínguez and José Muñoz. 2008. A neural model for the p-median problem. *Computers & Operations Research* 35, 2 (2008), 404–416.

[5] Shi Dong, Ping Wang, and Khushnood Abbas. 2021. A survey on deep learning and its applications. *Computer Science Review* 40 (2021), 100379.

[6] Roberto Diéguez Galvão and Ernesto del R Santibañez-Gonzalez. 1992. A Lagrangean heuristic for the pk-median dynamic location problem. *European Journal of Operational Research* 58, 2 (1992), 250–262.

[7] Gurobi Optimization, LLC. 2023. Gurobi Optimizer Reference Manual. https://www.gurobi.com

[8] S Louis Hakimi. 1964. Optimum locations of switching centers and the absolute centers and medians of a graph. *Operations Research* 12, 3 (1964), 450–459.

[9] Amir M Hormozi and Basheer M Khumawala. 1996. An improved algorithm for solving a multi-period facility location problem. *IIE Transactions* 28, 2 (1996), 105–114.

[10] Diederik P. Kingma and Jimmy Ba. 2014. Adam: A Method for Stochastic Optimization. *CoRR* abs/1412.6980 (2014). https://api.semanticscholar.org/CorpusID:6628106

[11] Scott Kirkpatrick, C Daniel Gelatt Jr, and Mario P Vecchi. 1983. Optimization by simulated annealing. *Science* 220, 4598 (1983), 671–680.

[12] Wouter Kool, Herke van Hoof, and Max Welling. 2019. Attention, Learn to Solve Routing Problems!. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*.

[13] Gilbert Laporte, Stefan Nickel, and Francisco Saldanha-da Gama. 2019. *Introduction to Location Science*. Springer International Publishing, Cham, 1–21.

[14] Eugene L Lawler and David E Wood. 1966. Branch-and-bound methods: A survey. *Operations Research* 14, 4 (1966), 699–719.

[15] Haojian Liang, Shaohua Wang, Huilai Li, Huichun Ye, and Yang Zhong. 2022. A trade-off algorithm for solving p-center problems with a graph convolutional network. *ISPRS International Journal of Geo-Information* 11, 5 (2022), 270.

[16] Haojian Liang, Shaohua Wang, Huilai Li, Liang Zhou, Hechang Chen, Xueyan Zhang, and Xu Chen. 2024. Sponet: solve spatial optimization problem using deep reinforcement learning for urban spatial decision analysis. *International Journal of Digital Earth* 17, 1 (2024), 2299211.

[17] Alfredo Marín, Luisa I Martínez-Merino, Antonio M Rodríguez-Chía, and Francisco Saldanha-da Gama. 2018. Multi-period stochastic covering location problems: Modeling framework and solution approach. *European Journal of Operational Research* 268, 2 (2018), 432–449.

[18] Stefan Nickel and Francisco Saldanha-da Gama. 2019. Multi-period facility location. *Location Science* (2019), 303–326.

[19] Susan Hesse Owen and Mark S Daskin. 1998. Strategic facility location: A review. *European Journal of Operational Research* 111, 3 (1998), 423–447.

[20] Bo Peng, Jiahai Wang, and Zizhen Zhang. 2020. A deep reinforcement learning algorithm using dynamic attention model for vehicle routing problems. In *Artificial Intelligence Algorithms and Applications: 11th International Symposium, ISICA 2019, Guangzhou, China, November 16–17, 2019, Revised Selected Papers 11*. Springer, 636–650.

[21] Marcelo O Rosa, Keiko VO Fonseca, Nádia P Kozievitch, Anderson A De-Bona, Jeferson L Curzel, Luciano U Pando, Olga M Prestes, and Ricardo Lüders. 2020. Advances on urban mobility using innovative data-driven models. *Handbook of Smart Cities* (2020), 1–38.

[22] Christophe Sauvey, Teresa Melo, and Isabel Correia. 2020. Heuristics for a multi-period facility location problem with delayed demand satisfaction. *Computers & Industrial Engineering* 139 (2020), 106171.

[23] Richard S. Sutton and Andrew G. Barto. 2018. *Reinforcement Learning: An Introduction* (second ed.). The MIT Press. http://incompleteideas.net/book/the-book-2nd.html

[24] Michael B Teitz and Polly Bart. 1968. Heuristic methods for estimating the generalized vertex median of a weighted graph. *Operations Research* 16, 5 (1968), 955–961.

[25] Tony J Van Roy and Donald Erlenkotter. 1982. A dual-based procedure for dynamic facility location. *Management Science* 28, 10 (1982), 1091–1105.

[26] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is All you Need. In *Advances in Neural Information Processing Systems*, Vol. 30. 5998–6008.

[27] Amit Kumar Vatsa and Sachin Jayaswal. 2021. Capacitated multi-period maximal covering location problem with server uncertainty. *European Journal of Operational Research* 289, 3 (2021), 1107–1126.

[28] Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. 2015. Pointer Networks. In *Advances in Neural Information Processing Systems*, Vol. 28.

[29] Shaohua Wang, Haojian Liang, Yang Zhong, Xueyan Zhang, and Cheng Su. 2023. DeepMCLP: Solving the MCLP with Deep Reinforcement Learning for Urban Spatial Computing. In *UC Santa Barbara: Center for Spatial Studies*.

[30] Shaohua Wang, Zezhi Zhang, Cheng Su, Liang Zhou, Haojian Liang, and Wenda Wang. 2023. Spatial Optimization Site Selection of Beijing Cainiao Station Based on Multi-Source Geospatial Data. In *Proceedings of the 1st ACM SIGSPATIAL International Workshop on Geocomputational Analysis of Socio-Economic Data*. 20–23.

[31] Wei Wang, Shining Wu, Shuaian Wang, Lu Zhen, and Xiaobo Qu. 2021. Emergency facility location problems in logistics: Status and perspectives. *Transportation Research Part E: Logistics and Transportation Review* 154 (2021), 102465.

[32] George O Wesolowsky and William G Truscott. 1975. The multiperiod location-allocation problem with relocation of facilities. *Management Science* 22, 1 (1975), 57–65.

[33] Ronald J Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning* 8 (1992), 229–256.

[34] Min Wu, Bingxin Yan, Ying Huang, and Md Nazirul Islam Sarker. 2022. Big data-driven urban management: Potential for urban sustainability. *Land* 11, 5 (2022), 680.

[35] Tongyu Wu, Yuntian Zhang, Changhao Miao, Chen Chen, and Shuxin Ding. 2024. Mixed-Variable Correlation-Aware Metaheuristic for Deployment Optimization of 3-D Sensor Networks. In *Proceedings of the Genetic and Evolutionary Computation Conference*. 1390–1398.

[36] Jin Zhang, Zhenpo Wang, Eric J Miller, Dingsong Cui, Peng Liu, Zhaosheng Zhang, and Zhenyu Sun. 2023. Multi-period planning of locations and capacities of public charging stations. *Journal of Energy Storage* 72 (2023), 108565.