Urban Sensing for Multi-Destination Workers via Deep Reinforcement Learning

Shuliang Wang[1], Song Tang[1], Sijie Ruan[1*], Cheng Long[2], Yuxuan Liang[3],
Qi Li[1], Ziqiang Yuan[1], Jie Bao[4,5], Yu Zheng[4,5]
[1]*Beijing Institute of Technology, Beijing, China* [2]*Nanyang Technological University, Singapore*
[3]*HKUST(GZ), Guangzhou, China* [4]*JD iCity, JD Technology, Beijing, China* [5]*JD Intelligent Cities Research, China*
{slwang2011,songtang,sjruan,liqi,ziqiangy}@bit.edu.cn; c.long@ntu.edu.sg;
yuxuanliang@hkust-gz.edu.cn; baojie@jd.com; msyuzheng@outlook.com

***Abstract*—Urban sensing aims to sense the status of the city, e.g., air quality, noise level, concentration of viruses, which can be completed by spatial crowdsourcing. Multi-destination people, who have many intermediate locations to visit before the final destination, e.g., couriers and tourists, are ideal recruitment candidates to conduct sensing tasks since they spend more time outside and have a wide spatio-temporal distribution. However, existing spatial crowdsourcing methods are only designed for workers who have single destinations, e.g., commuters, which are not applicable to recruit the multiple-destination people. Therefore, in this paper, we generalize the urban crowdsensing problem to the multi-destination scenario, namely, Urban Sensing for Multi-Destination Workers (USMDW). We prove its NP-hardness, and propose a framework Urban Sensing for Multi-destination Workers via Deep REinforcement learning, i.e., SMORE, to solve it effectively and efficiently. SMORE is composed of two steps: 1) candidate assignment initialization, which initializes all feasible sensing task-worker assignment pairs by a pre-trained reinforcement learning-based working route planning solver; and 2) reinforcement learning-based iterative selection, which iteratively selects a sensing task-worker pair to the current assignment via a novel policy network, i.e., Two-stage Assignment Selection Network (TASNet). Extensive experiments on three real-world datasets show SMORE outperforms the best baseline in data coverage by 5.2% on average with high efficiency.**

***Index Terms*—Urban Sensing, Spatial Crowdsourcing, Reinforcement Learning**

I. Introduction

Urban sensing is an essential prerequisite of urban computing [1], which aims to sense the status of the city, e.g., air quality, noise level, traffic condition, concentration of viruses and illegal parking events. The sensed data can be used as the input for prediction [2] and decision making [3]. Traditionally, static geospatial sensors or surveillance cameras are deployed to sense the urban state. With the development of positioning technology, humans equipped with mobile phones become a kind of mobile sensor. Sensing based on humans is known as mobile crowdsensing [4], which can be categorized into opportunistic sensing [5] and participatory sensing [6]. The sensed data collected by opportunistic sensing is random and of low quality since the movement cannot be controlled, while in participatory sensing, the platform can give incentives to workers, and thus obtain sensed data of high quality via controlling their mobility.

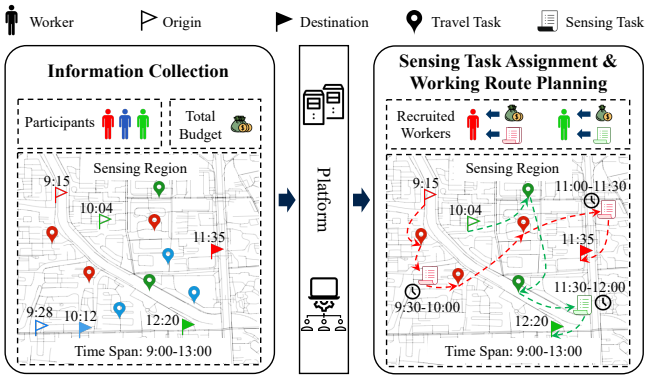


Fig. 1. Urban Sensing for Multi-Destination Workers

Existing participatory sensing schemes for urban sensing are usually designed for workers with a single travel destination [7], [8], e.g., commuters with commuting plans (an origin and a destination) [8]. However, there exists another group of people who also have great potential to complete urban sensing tasks, e.g., couriers and tourists, since they spend more time outside and have a wide spatio-temporal distribution. Since these people usually visit multiple locations in a trip, existing urban crowdsensing schemes do not support recruiting them.

Therefore, in this paper, we study the problem of Urban Sensing for Multi-Destination Workers (USMDW), so that the urban crowdsensing project can engage a wider group of people. More specifically, as shown in Figure 1, given a set of sensing tasks and a budget, based on the multi-destination information submitted by participants, we recruit some of them as workers and compute working routes for them to complete sensing tasks to maximize a pre-defined sensing objective. However, USMDW is challenging due to the following reasons - in fact, the problem is NP-hard, as we prove in this paper:

- **Large Action Space.** The process involves selecting multiple workers and assigning various sensing tasks to each worker. The number of tasks is considerable due to the need

*Sijie Ruan is the corresponding author who contributed the main idea and algorithms of this paper.

for collecting detailed data over the spatio-temporal range. Consequently, the action space becomes very large, leading to a huge computational burden in finding a good solution. Despite this hurdle, it is crucial to efficiently address USMDW since both the multi-destination information provided by participants and sensing tasks could be outdated.

- **Worker Heterogeneity.** USMDW shares similarities with the Team Orienteering Problem with Time Windows and Mandatory Visits (TOPTW-MV) [9] which aims to maximize the total collected scores at different locations by a group of travelers. In TOPTW-MV, the score at each location is associated with a time window and there are certain mandatory visit locations that must be included in the solution. However, the mandatory visits in TOPTW-MV are not specific to individual travelers, in our USMDW settings, each worker has unique mandatory visit locations tailored to their targets, making the workers heterogeneous. Considering the above distinction, the solutions devised for TOPTW-MV cannot be directly applied to tackle USMDW.
- **Dynamic Task Value.** In urban sensing, our concern extends beyond the mere quantity of collected sensed data; we also prioritize achieving a balanced distribution of data across the spatio-temporal landscape. This balance can be effectively characterized using a metric called Hierarchical Entropy-based Data Coverage [8]. Notably, this metric introduces interdependence among the values of individual sensing tasks, in contrast to the independent values considered in TOPTW-MV. As a result, this interdependence further complicates the optimization problem.

Inspired by the recent progress of deep reinforcement learning for effectively and efficiently solving complex combinatorial optimization problems [10], we propose a novel framework entitled SMORE, which stands for Urban Sensing for Multi-Destination Workers via Deep REinforcement learning.

To tackle the first challenge posed by the extensive action space, SMORE is proposed as an iterative selection framework, which first initializes all feasible sensing task-worker assignment pairs by a pre-trained reinforcement learning-based working route planning solver, and then iteratively selects a sensing task-worker assignment pair to the current assignment via a novel policy network until the budget is used up or there is no feasible working route satisfying the time constraints. The policy network is named as Two-stage Assignment Selection Network (TASNet), which sequentially selects a worker and a sensing task to further reduce the difficulties of the decision making. To tackle the challenge of worker heterogeneity, TASNet incorporates a separate encoding for the information pertaining to each worker. This personalized encoding serves as contextual information during the selection process of both the worker and the associated sensing task. To handle the dynamic task value, TASNet computes the value of each sensing task using a selection heuristic. This value calculation is then integrated into TASNet through data fusion and a soft mask function to boost the model performance.

Our contributions can be summarized as follows:

- We generalize the existing urban crowdsensing problem to the multi-destination scenario and prove its NP-hardness.
- We propose an iterative selection framework SMORE to tackle USMDW, which is an RL-based assignment strategy and is equipped with an existing route planning solver.
- We design TASNet to select sensing task-worker assignment pairs, which handles the large action space and integrates the selection heuristic by data fusion and a soft mask function.
- Extensive experiments on three real-world datasets demonstrate the effectiveness and efficiency of SMORE. SMORE outperforms the best baseline by 5.2% on average with high efficiency. We have released the code and data for public use[1].

II. Preliminaries

In this section, we introduce some definitions and formalize the problem of USMDW.

A. Definitions

Definition 1 (Travel Task). *A travel task, e.g., delivering a parcel, or visiting a tourist attraction, is denoted as* $d =< l, \tau >$. $d.l$ *is the geographical location of* d*, and* $d.\tau$ *is the period of time required to complete* d*.*

Definition 2 (Multi-destination Worker). *A multi-destination worker* w *is denoted as* $w =< l_s, l_e, t_s^{min}, t_e^{max}, \mathcal{D} >$*, where* $w.l_s$ *and* $w.l_e$ *are the origin and final destination of* w*,* $w.t_s^{min}$ *and* $w.t_e^{max}$ *are the feasible earliest departure and the latest arrival time, and* $w.\mathcal{D}$ *is the set of travel tasks* w *must completed during the trip.*

In the following, we refer to multi-destination workers as workers for abbreviation if there is no ambiguity.

Definition 3 (Sensing Task). *An urban sensing task, e.g., sensing the air quality of an area over a certain temporal range, is denoted as* $s =< l, tw_s, tw_e, \tau >$. $s.l$ *is the location of* s*, and* $s.tw_s$ *and* $s.tw_e$ *form the available time window that the task can be completed, respectively.* $s.\tau$ *is the period of time required to complete* s*.* s *can be completed by only one worker, and his/her sensing period must fully fall into the time window of* s*, i.e., the arrival time* t *of the worker at* $s.l$ *satisfies:* $s.tw_s \leq t \leq s.tw_e - s.\tau$*.*

The sensing task set $\mathcal{S}$ can be manually designated or automatically created: Given a region of interest and a sensing time span, $\mathcal{S}$ can be constructed by partitioning the spatio-temporal range with pre-defined spatial and temporal resolutions.

We employ the Hierarchical Entropy-based Data Coverage proposed in [8] to quantify the overall sensing quality, which cares about not only the number of completed sensing tasks, but also the degree of sensed data balance across the spatio-temporal landscape.

Definition 4 (Hierarchical Entropy-based Data Coverage [8]). *The hierarchical entropy-based data coverage is calculated by* $\phi(\mathcal{S}') = \alpha E(\mathcal{S}') + (1-\alpha)\log_2|\mathcal{S}'|$*, where* $\mathcal{S}'$

[1]https://github.com/SongTunes/SMORE

is the set of completed sensing tasks and $E(\mathcal{S}')$ measures the degree of spatio-temporal balance of completed sensing tasks through the hierarchical entropy, $\alpha \in [0,1]$ is a trade-off hyperparameter to tune the importance of the balance of collected data to the number of collected data.

Definition 5 (Working Route & Route Travel Time). *The working route of a worker w describes his/her traveling sequence, which is denoted as $\mathcal{R}_w = w.l_s \rightarrow ta_1 \rightarrow ta_2 \rightarrow ... \rightarrow ta_{|w.\mathcal{D}|+|\mathcal{S}_w|} \rightarrow w.l_e$, where $\mathcal{S}_w$ is the set of assigned sensing tasks of worker w, and ta_i either is a travel task in $w.\mathcal{D}$ or an assigned sensing task in $\mathcal{S}_w$. The route travel time $rtt_{\mathcal{R}_w}$ is the summation of travel time between consecutive locations as well as the time spent waiting and completing different tasks, which is formally defined as follows:*

$$\begin{aligned} rtt_{\mathcal{R}_w} = & tt(w.l_s, ta_1.l) + tt(ta_{|w.\mathcal{D}|+|\mathcal{S}_w|}.l, w.l_e) \\ & + \sum_{i=1}^{|w.\mathcal{D}|+|\mathcal{S}_w|-1} tt(ta_i.l, ta_{i+1}.l) \\ & + \sum_{i=1}^{|w.\mathcal{D}|+|\mathcal{S}_w|} [wt(ta_i) + ta_i.\tau] \end{aligned} \tag{1}$$

where $tt(l_i, l_j)$ is the travel time between l_i and l_j, $wt(ta)$ is the waiting time for completing task ta. For a sensing task s, the waiting time is the difference between $s.tw_s$ and the arrival time of the worker at $s.l$ if he/she arrives earlier than $s.tw_s$, else 0. For travel tasks, the waiting time is always 0.

A working route $\mathcal{R}_w$ is feasible for the worker w only if the route travel time meets the time constraint of w, i.e., $w.t_s^{min} + rtt_{\mathcal{R}_w} \leq w.t_e^{max}$.

In this study, we assume workers are moving at a constant speed in free space, i.e., the travel time is proportional to the Euclidean distance between locations. Alternatively, the travel time can also be estimated based on historical data [11], [12].

Definition 6 (Incentive). *The incentive is the allowance given to the worker, which is proportional to the additional time cost of the worker brought by completing sensing tasks. The incentive of worker w is calculated as:*

$$in_{\mathcal{R}_w} = \mu \times [rtt_{\mathcal{R}_w} - rtt_{TSP(w.l_s, w.l_e, w.\mathcal{D})}] \tag{2}$$

where μ is the incentive per time unit, and the latter part is the time cost difference between the actual working route with sensing tasks assigned $\mathcal{R}_w$ and his/her original route with the minimum time cost. The original route is the solution of a Travelling Salesman Problem (TSP) starting from l_s, ending at l_e, traversing through all locations in $\mathcal{D}$ to finish travel tasks, i.e., TSP$(l_s, l_e, \mathcal{D})$.

Note that, different from [8], whose incentive is proportional to the slack time of workers, in this study, the incentive is dynamic to the actual sensing tasks assigned to the worker, which makes our assignment more flexible.

B. Problem Statement

USMDW is defined as follows: *Given the urban sensing task set $\mathcal{S}$, the total budget B, the incentive per time unit μ, and the multi-destination worker set $\mathcal{W} = \{w_1, w_2, ..., w_{|\mathcal{W}|}\}$, design feasible working routes for $\mathcal{W}$ to complete a set of sensing tasks $\mathcal{S}' \subseteq \mathcal{S}$, such that the hierarchical entropy-based data coverage $\phi(\mathcal{S}')$ is maximized. Formally,*

$$\max_{\{\mathcal{R}_w\}_w^{\mathcal{W}}} \quad \phi(\mathcal{S}') \tag{3a}$$

$$\text{s.t.} \quad \textstyle\sum_{w \in \mathcal{W}} in_{\mathcal{R}_w} \leq B \tag{3b}$$

Lemma 1. *The USMDW problem is NP-hard.*

Proof. We can prove the NP-hardness of USMDW by reducing Orienteering Problem (OP) [13], which is an already known NP-hard problem.

We first give the statement of OP: *Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be a graph where $\mathcal{V}$ is a set of vertices and $\mathcal{E}$ is the set of edges. Each vertex is associated with a non-negative score and each edge is associated with a non-negative travel time cost. OP is to determine a Hamiltonian path over a subset of $\mathcal{V}$ to maximize the total score collected from the visited vertices, while the travel time of the path cannot exceed a given threshold T_{max}.*

Consider OP instances where the score of each vertex in graph $\mathcal{G}$ is 1, we can transform it to an instance of USMDW in the following way. Let the traveler in OP represent a worker whose set of travel tasks is empty and each vertex in $\mathcal{V}$ represents a sensing task. For the available time window of sensing task s, the start time $s.tw_s$ is set to 0 and the end time $s.tw_e$ is set to the time cost constraint T_{max}, the period of time required to complete a travel task or a sensing task are both set to 0 and the budget B for the sensing project can be set to infinity. In the objective, the weight α in $\phi(\mathcal{S}')$ is set to 0, in which case the objective is only related to the number of completed sensing tasks $|\mathcal{S}'|$, which is equivalent to maximizing the total collected score in such OP instance.

If we have a polynomial solver for USMDW, then the maximum total collected scores can be obtained in polynomial time. However, since OP is NP-hard, the NP-hardness of USMDW is proven.

□

III. Framework

In this section, we describe the framework SMORE. Given that USMDW is an NP-hard problem, rather than struggling on computing the optimal working routes, we seek methods that can give ideal results efficiently.

A. Overview

Given the observation that the optimal working routes rely on the optimal sensing task assignment, a natural idea to solve USMDW is to decompose the working route design into two sub-problems: 1) assigning sensing tasks to workers; and 2) calculating the working route for each worker based on the

assigned sensing tasks, his/her travel tasks as well as the origin and the final destination.

Issues. Since the second sub-problem essentially is a Traveling Salesman Problem with Time Window (TSPTW) [14], many meta-heuristic methods [15] can be employed, we mainly focused on addressing the first sub-problem. A straightforward idea to solve the first sub-problem is to use the greedy algorithm. More specifically, we first initialize an empty assignment set, and then we iteratively pick a sensing task-worker assignment pair that brings the maximum data coverage gain until the budget is used up. However, such a strategy has the following two issues:

- *Obtaining feasible sensing task-worker assignment pairs is time-consuming.* A sensing task-worker assignment pair is feasible if and only if when the sensing task s is assigned to the worker w, the worker can still complete all his/her travel tasks and arrive at his/her final destination $w.l_d$ before the latest arrival time $w.t_e^{max}$ and would not use up the remaining budget based on his/her updated incentive after the candidate sensing task s is added. Therefore, as long as there exists one working route that satisfies the above constraints, the assignment pair is feasible. Since the working route produced by TSPTW solver takes the minimum traveling time, and its additional budget cost is also the minimum given a certain assigned sensing task set according to Equation 2, it is the lower bound working route. With such a working route at hand, we can easily know whether the time or the budget constraint is violated. However, as we know TSPTW itself is NP-hard, and checking the feasibility of each assignment pair is time-consuming.
- *Only considering the data coverage gain is myopic.* On one hand, the data coverage gain is only related to the selected sensing task, which is irrelevant to the worker conducting it. Therefore, if a sensing task can be completed by several workers, the greedy algorithm can not guarantee the sensing task is assigned to the most appropriate worker. On the other hand, even if there is only a worker feasible to conduct the sensing task with maximum data coverage gain, the algorithm is easy to be trapped into the local maxima, since the greedy algorithm only makes the decision based on the data coverage gain in the current iteration, which may fail to achieve a higher data coverage if a sensing task with smaller data coverage gain need to be selected first.

Intuitions. Since deep reinforcement learning has the ability to make decisions efficiently and maximize the long-term reward via the neural network, it has the potential to solve the above issues. More specifically, 1) to tackle the time-consuming issue, we introduce a pre-trained RL-based TSPTW solver to efficiently perform the feasibility check. Additionally, it can also be reused to solve the second sub-problem, which further boosts the performance of the proposed method. 2) to tackle the myopic issue, a novel RL-based assignment selection module is devised to select the sensing task-worker assignment pair among all feasible candidates, which takes various information into consideration, e.g., the information

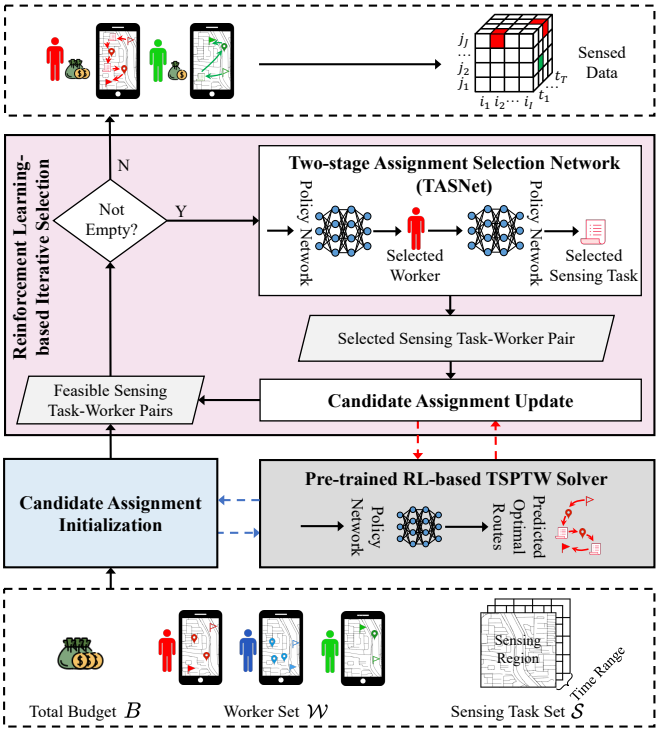


Fig. 2. Framework of SMORE.

of workers, the assignment status of all workers, remaining budget, rather than only the data coverage gain.

Main Idea. With the above intuitions, we propose SMORE, a reinforcement learning-based framework, to design working routes for multi-destination workers, which is depicted in Figure 2. SMORE follows the iterative assignment framework and is equipped with a pre-trained RL-based TSPTW solver. SMORE is composed of two steps: 1) candidate assignment initialization, which takes workers and all sensing tasks as input, and efficiently generates all feasible sensing task-worker assignment pairs based on the pre-trained TSPTW solver; and 2) reinforcement learning-based iterative selection, which iteratively selects sensing task-worker assignment pairs until no feasible sensing task-worker assignment pair exists. In each iteration, the most promising sensing task-worker assignment pair is selected based on RL, and feasible sensing tasks for the selected worker would be updated based on the pre-trained TSPTW solver. Next, we elaborate on the detailed implementation of two steps as well as the pre-trained solver.

B. Implementation

The pseudocode of SMORE in the inference stage is shown in Algorithm 1, which takes worker set $\mathcal{W}$, sensing task set $\mathcal{S}$, total budget B, incentive per time unit μ, the RL-based TSPTW solver f_{TSPTW} as well as the proposed RL-based task assignment network f_{TASNet}, and returns the working routes for workers $\{\mathcal{R}\}_w^{\mathcal{W}}$. The remaining budget B_{rest} is first set to the total budget B, and we create a set $\mathcal{S}'$ to store all

Algorithm 1: SMORE

Input : Worker set $\mathcal{W}$, sensing task set $\mathcal{S}$, total budget B, incentive per time unit μ, RL-based TSPTW solver f_{TSPTW}, RL-based task assignment selection network f_{TASNet}.

Output: Working routes of workers $\{\mathcal{R}\}_w^{\mathcal{W}}$.

1 $B_{rest} \leftarrow B$; $\mathcal{S}' \leftarrow \emptyset$; $\mathcal{C} \leftarrow \emptyset$; $\mathcal{M} \leftarrow \emptyset$;
2 **for** $w \in \mathcal{W}$ **do**
3 $\quad \mathcal{M}[w] \leftarrow (\emptyset, \text{null}, 0)$;
/* Step 1. Candidate Assignment Init. */
4 **for** $w \in \mathcal{W}$ **do**
5 $\quad$ **for** $s \in \mathcal{S}$ **do**
6 $\quad\quad \mathcal{R}_w, rtt_{R_w} \leftarrow f_{\text{TSPTW}}(w.l_s, w.l_d, w.\mathcal{D} \cup \{s\}, w.t_s^{min}, w.t_e^{max})$;
7 $\quad\quad \Delta_{in} \leftarrow \texttt{Incentive}(rtt_{R_w}, w, \mu) - 0$;
8 $\quad\quad$ **if** $\Delta in < B_{rest}$ **then**
9 $\quad\quad\quad \mathcal{C}[w][s] \leftarrow (\mathcal{R}_w, \Delta_{in})$;
/* Step 2. RL-based Iterative Selection */
10 **while** $\mathcal{C} \neq \emptyset$ **do**
11 $\quad (w^*, s^*) \leftarrow f_{\text{TASNet}}(\mathcal{C}, \mathcal{M}, \mathcal{W}, B_{rest})$;
12 $\quad B_{rest} \leftarrow B_{rest} - \mathcal{C}[w^*][s^*].\Delta_{in}$;
13 $\quad \mathcal{M}[w^*] \leftarrow (\mathcal{M}[w^*].\mathcal{S}_w \cup \{s^*\}, \mathcal{C}[w^*][s^*].\mathcal{R}_w, \mathcal{M}[w^*].in + \mathcal{C}[w^*][s^*].\Delta_{in})$;
14 $\quad \mathcal{S}' \leftarrow \mathcal{S}' \cup \{s^*\}$;
15 $\quad$ **for** $w \in \mathcal{W}$ **do**
16 $\quad\quad$ remove s^* from $\mathcal{C}[w]$;
17 $\quad$ **for** $s \in \mathcal{S} \setminus \mathcal{S}'$ **do**
18 $\quad\quad \mathcal{R}_w, rtt_{R_w} \leftarrow f_{\text{TSPTW}}(w^*.l_s, w^*.l_d, w^*.\mathcal{D} \cup \mathcal{M}[w^*].\mathcal{S}_w \cup \{s\}, w^*.t_s^{min}, w^*.t_e^{max})$;
19 $\quad\quad \Delta_{in} \leftarrow \texttt{Incentive}(rtt_{R_w}, w^*, \mu) - \mathcal{M}[w^*].in_{R_w}$;
20 $\quad\quad$ **if** $\Delta_{in} < B_{rest}$ **then**
21 $\quad\quad\quad \mathcal{C}[w^*][s] \leftarrow (\mathcal{R}_w, \Delta in)$;
22 $\quad\quad$ **else**
23 $\quad\quad\quad$ remove s from $\mathcal{C}[w^*]$;
24 **return** $\{\mathcal{M}[w].\mathcal{R}_w | \forall w \in \mathcal{M}\}$

selected sensing tasks, and a hashmap $\mathcal{C}$ to store all feasible sensing task-worker assignment pairs, whose key is the sensing task-worker assignment pair, and value contains the working route after the sensing task is assigned to the worker and its corresponding additional incentive (i.e., the incentive difference before and after assigning the sensing task). A hashmap $\mathcal{M}$ is also created to store the assignment of sensing tasks, whose key is the worker, and value contains his/her current assigned sensing task set, corresponding working route as well as the incentive currently given to the worker (Lines 2-3). After that, candidate assignment initialization and reinforcement learning-based iterative selection are executed.

Candidate Assignment Initialization. In this step, we check the feasibility of assigning arbitrary sensing task to each worker by calling the pre-trained solver f_{TSPTW}. For a worker w, based on his/her intermediate locations to visit $w.\mathcal{D}$ as well as the current sensing task s to assign to, if there exists a route that meets the time constraint, and the additional incentive to give is also smaller than the remaining budget, the assignment pair, the calculated route, and the corresponding additional incentive would be inserted to the hashmap $\mathcal{C}$ (Lines 5-9).

Reinforcement Learning-based Iterative Selection. This step would loop until $\mathcal{C}$ is empty. In each iteration, a sensing task-worker assignment pair is selected from $\mathcal{C}$ based on candidate assignments $\mathcal{C}$, current assignments $\mathcal{M}$, set of workers $\mathcal{W}$, and remaining budget B_{rest} by a novel policy network, i.e., TASNet, which would be discussed later in detail in Section IV (Line 11). Once such a pair of assignment (w^*, s^*) is selected, the remaining budget, current assignment status, and assigned sensing tasks would be updated (Lines 12-14), and the affected candidate assignments in $\mathcal{C}$ would be recomputed. More specifically, s^* would first be removed from candidates of other workers, and then since the feasible assignment of w^* is affected by the adding of s^*, the feasible sensing tasks for w^* would be recomputed (Lines 15-23).

After the candidate assignment hashmap $\mathcal{C}$ is empty, the current working routes stored in $\mathcal{M}$ are returned (Line 24).

C. Pre-trained Working Route Planning Solver

When we need to plan the working route of a worker w, he/she usually has both travel tasks and sensing tasks. Though travel tasks do not contain time windows, we can set their time windows as $[w.t_s^{min}, w.t_e^{max}]$. In this way, each intermediate location has a time window. Therefore, the working route planning problem essentially is a TSPTW. An efficient and accurate TSPTW solver is essential in our problem, since once a sensing task is assigned to a worker, his/her feasibility sensing tasks would change. Here we employ a hierarchical reinforcement learning (HRL)-based TSPTW solver [16]. Specifically, the solver is divided into two DRL models, which enables the model to learn a better policy by enhancing time-window-constraint handling using the lower model. Since there is only one depot in [16], we adapt their method by including both the origin and the final destination information into the query vector to make the selection. TSPTW solver is pre-trained for calling by SMORE. It involves two steps: 1) lower model training, which is optimized by the lower rewards, i.e., the number of nodes that meet the time window constraint; and 2) upper model training, which is optimized by the upper rewards, i.e., adding a penalty for the length of the route to the lower rewards. Refer to [16] for more training details.

D. Complexity Analysis

We give a complexity analysis of SMORE. Since the RL-based TSPTW solver decodes the route location by location, its time complexity is $\mathcal{O}(|\mathcal{S}|)$, where the number of travel tasks of a worker usually is not large and can be ignored.

In the first step, SMORE needs to check the feasibility of any sensing task-worker assignment pair, and each feasibility check needs to call the TSPTW solver. Therefore, the time complexity of the first step is $\mathcal{O}(|\mathcal{W}||\mathcal{S}|^2)$.

In the second step, the iterations would be conducted for $|\mathcal{W}||\mathcal{S}|$ times in the worst case. In each iteration, TASNet is first called to select a sensing task-worker assignment pair among candidates, which takes $\mathcal{O}(1)$. After an assignment is selected, the candidate assignments for the selected worker need to be updated, which takes $\mathcal{O}(|\mathcal{W}| + |\mathcal{S}|^2)$. Hence, the complexity of the second step is $\mathcal{O}(|\mathcal{W}|^2|\mathcal{S}| + |\mathcal{W}||\mathcal{S}|^3)$.

Therefore, the overall time complexity of SMORE is $\mathcal{O}(|\mathcal{W}|^2|\mathcal{S}|+|\mathcal{W}||\mathcal{S}|^3)$. In practice, the "for" loop for workers and sensing tasks can be implemented in parallel by batching the data and then passing through the pre-trained TSPTW solver, which can further boost the inference speed.

IV. Reinforcement Learning-based Iterative Selection

In this section, we provide a detailed description of reinforcement learning-based iterative selection.

A. Iterative Selection Modeled as an Markov Decision Process

Since making the selection based on the data coverage gain is myopic, we model the iterative sensing task-worker assignment pair selection process as a Markov Decision Process (MDP), which can consider various information to make the decision to maximize the long-term reward. It consists of four components: states, actions, transitions, and rewards, which are defined as follows.

States. The state includes all information related to the decision making, which can be denoted as $\mathtt{s}_t = (\mathcal{C}_t, \mathcal{M}_t, \mathcal{W}, B_t)$, where t is the current decision time step, $\mathcal{C}_t$ and $\mathcal{M}_t$ are candidate sensing tasks and assigned sensing tasks for each worker at time step t, respectively, B_t is the remaining budget at time step t. We also incorporate the static information of workers $\mathcal{W}$ which would not change across different decision time steps, i.e., the origin, final destination, as well as the set of travel tasks, to enrich the state representation.

Actions. An action is a sensing task assignment decision, which can be denoted as $\mathtt{a}_t =< w_i^t, s_j^t >$, meaning assigning sensing task s_j to worker w_i at time step t.

Transitions. After $\mathtt{a}_t$ is conducted, i.e., a candidate sensing task assignment is selected at time step t, $\mathtt{s}_t$ is updated to $\mathtt{s}_{t+1}$. Based on the selected sensing task and the selected worker, we update the state, i.e., candidate sensing tasks, assigned sensing tasks as well as the remaining budget in the same way as we already explained in Lines 12-23 of Algorithm 1.

Reward. Since our optimization goal is to maximize the data coverage, the reward at time step t is defined as the data coverage gain after completing the sensing task selected in $\mathtt{a}_t$, i.e., the task value of the sensing task, which is denoted as $r_t = \phi(\mathcal{S}'_{t+1}) - \phi(\mathcal{S}'_t)$, where $\mathcal{S}'_t$ and $\mathcal{S}'_{t+1}$ are the selected sensing tasks at time step t and time step $t+1$, respectively.

After modeling the iterative selection process as an MDP, it can be solved by reinforcement learning. Since the state contains various information and is rather complex, we propose a policy network, i.e., TASNet, to make the decision. We next introduce TASNet and the learning algorithm.

B. Overview of TASNet

Given the states and actions defined in MDP, we can now design a policy network to achieve the mapping from a state to an action.

Issues. Given all candidate sensing task-worker pairs, a straightforward idea is that we produce the probability for each pair through the network all at once. However, given $|\mathcal{W}|$ workers and $|\mathcal{S}|$ candidate sensing tasks, there are at most $|\mathcal{W}||\mathcal{S}|$ possible choices. Leveraging the above strategy is not only time-consuming, but also difficult to learn a good policy given massive choices.

Intuitions. Since the assignment selection requires both picking a worker and picking a candidate sensing task, if we can make those two decisions by turns, the complexity of the selection can be reduced, i.e., we first pick a worker w from $\mathcal{W}$ and then pick a sensing task from his/her feasible sensing tasks. In this case, the number of choices can be roughly reduced to $|\mathcal{W}| + |\mathcal{S}|$.

Main Idea. Therefore, to tackle the above issue, we propose TASNet as shown in Figure 3, which is composed of three modules, 1) *Worker & Sensing Task Representation*, which separately encodes the information about workers and sensing tasks into latent embeddings; 2) *Worker Selection*, which takes the embeddings of workers, their assigned sensing tasks, and remaining budget to select a worker to be assigned; and 3) *Sensing Task Selection*, which takes the embedding of the selected worker, the embeddings of his/her assigned sensing tasks and feasible sensing tasks, and selects a sensing task assigned to him/her. We next give a comprehensive description of each module.

C. Worker & Sensing Task Representation

Worker & Sensing Task Representation aims to transform workers and sensing tasks into dense vector representations, which can be later used to encode the current state.

Worker Embedding. Since different workers are heterogeneous and are characterized by their origins, final destinations as well as travel tasks, the assignment selection should also consider the convenience of others. Therefore, when calculating the worker embedding, both the spatial distributions of locations as well as the correlation among workers should be considered. For each worker, we first partition the region of interest into uniform grids and create a corresponding matrix initialized with all 0 to store the travel information of the worker. The value of the entry is set to 1, 2, 3 if its corresponding grid cell contains the origin, destination, travel tasks of the worker, respectively. Then we apply a convolutional layer and a fully connected (FC) layer on the matrix to encode the spatial distributions of the travel information of the worker. Finally, a Transformer [17] like encoder, which is composed of a multi-head attention (MHA) layer and a node-wise fully connected feed-forward (FF) layer, is applied to fuse the information of different workers. The output of which serves as the embeddings of workers $\{\mathbf{w}_i\}_{i=1}^{\mathcal{W}}$.

Sensing Task Embedding. Sensing task embedding aims to capture the spatio-temporal closeness of different sensing

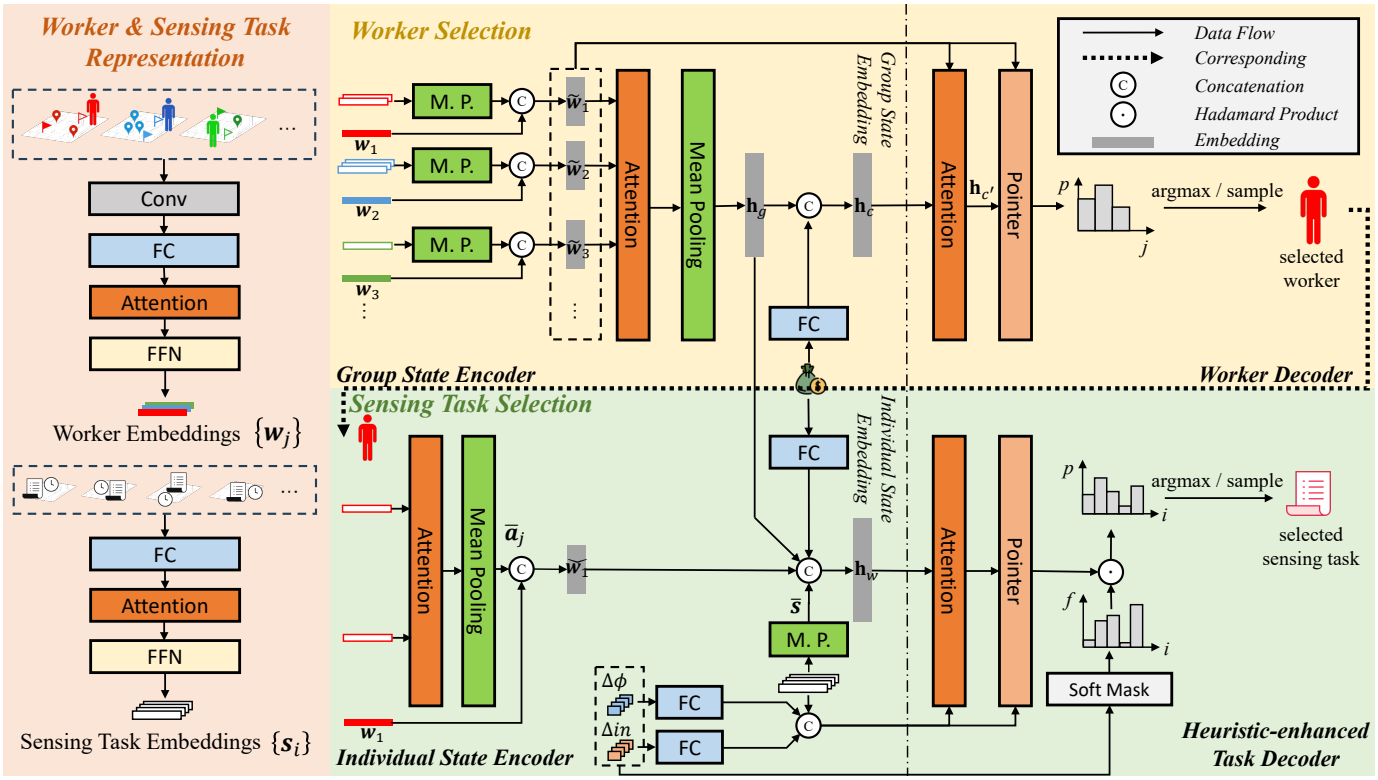


Fig. 3. Architecture of Two-stage Assignment Selection Network.

tasks. Those embeddings would serve as the initial representations for both assigned sensing tasks as well as candidate sensing tasks to be assigned. The input of a sensing task contains its location as well as its time window, we apply another Transformer like encoder to capture their spatio-temporal closeness and obtain the embeddings $\{\mathbf{s}_i\}_{i=1}^{\mathcal{S}}$.

D. Worker Selection

Worker Selection takes the embeddings of workers, their assigned sensing tasks, and the remaining budget, and aims to decide which worker to select. This module is composed of a group state encoder and a worker decoder.

Group State Encoder. Group state encoder fuses the current state into a vectorized group state embedding, i.e., $\mathbf{h}_c$.

Firstly, for each worker j, given his/her current assigned sensing tasks, we first gather their embeddings $\{\mathbf{s}_1^j, \mathbf{s}_2^j, \cdots, \mathbf{s}_{|\mathcal{S}_j|}^j\}$ based on the previous module, and apply a mean pooling layer to obtain a uniform assigned task representation $\bar{\mathbf{s}}_j$ given that the assigned task set is varying in length among workers. Then, $\bar{\mathbf{s}}_j$ is concatenated with the embedding of the worker $\mathbf{w}_j$ to create the worker state embedding, i.e., $\tilde{\mathbf{w}}_j = [\bar{\mathbf{s}}_j; \mathbf{w}_j]$, where ; denotes the concatenation operation. After that, a MHA layer followed by a mean pooling layer is applied among all workers to obtain the group worker embedding $\mathbf{h}_g$ as Equation 4 shown:

$$\mathbf{h}_g = \text{MeanPool}(\text{MHA}(\{\tilde{\mathbf{w}}_1, \tilde{\mathbf{w}}_2, \cdots, \tilde{\mathbf{w}}_{|\mathcal{W}|}\})) \quad (4)$$

At last, the group worker embedding $\mathbf{h}_g$ is concatenated with the remaining budget B_{rest} to obtain the group state embedding, i.e., $\mathbf{h}_c = [\mathbf{h}_g; \text{FC}(B_{rest})]$.

Worker Decoder. The worker decoder aims to select a worker based on the given group state embedding $\mathbf{h}_c$. Inspired by the pointer mechanism in Attention Model [10], we first calculate a context embedding based on $\mathbf{h}_c$ that represents the current decoding state. Then we obtain the probability distribution of the selection to the input sequence (i.e., the worker set) using an attention-based pointer decoding method to derive the selected worker.

Firstly, according to the worker state embeddings $\{\tilde{\mathbf{w}}_1, \tilde{\mathbf{w}}_2, \cdots, \tilde{\mathbf{w}}_{|\mathcal{W}|}\}$ and the group state embedding $\mathbf{h}_c$, we apply a dot-product attention mechanism to calculate a new decoding context embedding $\mathbf{h}_{c'}$. This step aims to further extract the relationship information between group state embedding and worker state embeddings. Note that the attention weight of a worker j would be set to $-\infty$ if his/her candidate sensing task set is empty to make sure we would not select a worker without any feasible assignment.

Based on the calculated new decoding context embedding $\mathbf{h}_{c'}$, we calculate the probability distribution using a single-head attention mechanism, and the worker who has the maximum probability is selected. Formally,

$$\mathbf{q}_c = \mathbf{W}^Q \mathbf{h}_{c'}, \mathbf{k}_j = \mathbf{W}^K \tilde{\mathbf{w}}_j, \mathbf{v}_j = \mathbf{W}^V \tilde{\mathbf{w}}_j \ \forall j \in 1 \cdots |\mathcal{W}| \quad (5)$$

$$u_{cj}^{'} = \begin{cases} C \cdot \tanh(\frac{\mathbf{q}_c^T \mathbf{k}_j}{\sqrt{d_k}}) & \text{if } j \text{ is not masked} \\ -\infty & \text{otherwise} \end{cases} \tag{6}$$

$$p_j = \frac{\exp(u_{cj}^{'})}{\sum_{j'} \exp(u_{cj'}^{'})} \tag{7}$$

where $\mathbf{W}^Q$, $\mathbf{W}^K$, $\mathbf{W}^V$ are learnable matrices. Note that we also clip the result within $[-C, C]$ using tanh, similar to previous works [10], [18].

E. Sensing Task Selection

Sensing Task Selection takes the selected worker from the previous module, and selects one of his/her candidate tasks based on his/her embeddings and assigned sensing tasks, which is composed of an individual state encoder and a heuristic-enhanced task decoder.

Individual State Encoder. Individual state encoder fuses the current state of the selected worker into a vectorized individual state embedding, i.e., $\mathbf{h}_w$. More specifically, both the individual information about the worker and the global information are considered. 1) For the individual information, similar to the worker selection, for a selected worker j, we consider its worker embedding $\mathbf{w}_j$ as well as the assigned sensing tasks. For assigned sensing tasks, their embedings $\{\mathbf{s}_1^j, \mathbf{s}_2^j, \cdots, \mathbf{s}_{|\mathcal{S}_j|}^j\}$ are first fed into an attention layer followed by a mean pooling layer to obtain a semantic richer representation of the assignment status of the worker $\bar{\mathbf{a}}_j$. Then, those two parts are concatenated to generate an enhanced worker state embedding, i.e., $\check{\mathbf{w}}_j = [\bar{\mathbf{a}}_j; \mathbf{w}_j]$. 2) For the global information, in addition to considering the remaining budget B_{rest}, we further consider the group worker embedding $\mathbf{h}_g$ obtained from the worker selection to have a global view of the overall assignment status, as well as the global view of all sensing tasks $\bar{\mathbf{s}}$, which is calculated based on the mean pooling over all sensing task embeddings. Formally, the individual state embedding $\mathbf{h}_w$ is obtained as follows:

$$\mathbf{h}_w = [\check{\mathbf{w}}_j; \text{FC}(B_{rest}); \mathbf{h}_g; \bar{\mathbf{s}}] \tag{8}$$

Heuristic-enhanced Task Decoder. Heuristic-enhanced task decoder aims to select a sensing task based on the individual state embedding, i.e., $\mathbf{h}_w$.

A straightforward idea is to use the same decoder structure of the worker decoder to complete the sensing task selection (by replacing the worker embeddings with feasible sensing task embeddings and replacing group state embedding with individual state embedding). However, 1) some effective heuristics which are widely used by greedy algorithms, e.g., making the decision which brings the maximum objective gain, are not explicitly used by the RL algorithm; and 2) the action space of the sensing task selection is often much larger than worker selection, which makes it more difficult if the same strategy is applied.

To tackle the first issue, for each candidate sensing task i, we introduce two auxiliary signals, i.e., coverage gain $\Delta\phi_i$ and incentive cost Δin_i, which serve as the heuristics to enhance the selection performance. Δin_i is already explained in Section III, and $\Delta\phi_i$ is the data coverage difference if the sensing task is selected, which is computed on the fly to avoid unnecessary computations. More specifically, we concatenate Δin_i and $\Delta\phi_i$ with the original sensing task representations, which would be used for the attention query.

To tackle the second issue, we propose a soft mask mechanism based on those signals to modify the selection probability before the output to assist decisions. More specifically, we design a heuristic score, namely, coverage-incentive ratio β_i for each sensing task s_i: $\beta_i = \Delta\phi_i / \Delta in_i$, where $\Delta\phi_i$ and Δin_i are the difference of data coverage and incentive after and before the sensing task is selected, respectively. Intuitively, a sensing task with a higher coverage-incentive ratio means it takes less budget and can achieve a higher coverage gain. Based on the coverage-incentive ratio, the soft mask function is devised as follows:

$$f(\Delta\phi_i, \Delta in_i) = \exp(-\frac{\lambda^2}{\epsilon + (\hat{\beta}_i)^2}) \tag{9}$$

$$\hat{\beta}_i = \frac{\beta_i - \min(\boldsymbol{\beta})}{\max(\boldsymbol{\beta}) - \min(\boldsymbol{\beta})} \tag{10}$$

where $\boldsymbol{\beta}$ are all coverage-incentive ratios of sensing tasks at each decision time step, λ is a hyperparameter and ϵ is a tiny positive value to avoid zero division.

By integrating the soft mask function, the modified probabilities become:

$$p_i = \frac{\exp(u'_{ci} \odot f(\Delta\phi_i, \Delta in_i))}{\sum_{i'} \exp(u'_{ci'} \odot f(\Delta\phi_{i'}, \Delta in_{i'}))} \tag{11}$$

F. Training Process

The training process of TASNet is explained as follows. We use the REINFORCE algorithm with a critic baseline [19] because we find that using a critic baseline has higher training efficiency compared to some self-critic methods (e.g., rollout baseline [10]). We first randomly initialize the parameters of TASNet. Then for each training iteration, a batch of instances of USMDW would be sampled. For each instance, after candidate assignments are calculated by the pre-trained RL-based TSPTW solver, we feed them into TASNet together with state update module to produce the "assignment trajectory" step by step, i.e., $\pi = \{(w^1, s^1), (w^2, s^2), ...\}$ until the candidate assignment set is empty. After that, we collect the assignment results and corresponding decoding probabilities of a batch, and update the model parameters using policy gradient [20]:

$$\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}|\mathbf{s}) = \mathbb{E}_{p_\theta(\pi|\mathbf{s})} \left[(\phi(\pi) - b(\mathbf{s})) \nabla_{\boldsymbol{\theta}} \log p_{\boldsymbol{\theta}}(\pi|\mathbf{s}) \right] \tag{12}$$

where $\boldsymbol{\theta}$ are parameters of TASNet, $\phi(\pi)$ is the data coverage given π, $b(\mathbf{s})$ is the baseline value estimated by the critic network for state $\mathbf{s}$.

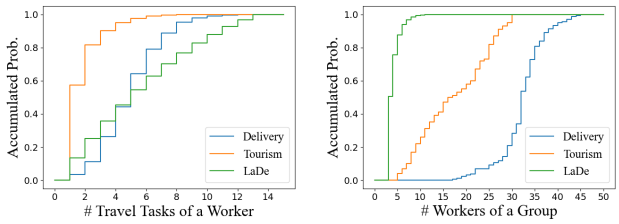
V. Experiment

A. Dataset

The experiments are carried out on three real-world datasets, i.e., two moderate datasets from JD Logistics and Flickr, and a large-scale dataset from Cainiao Network.

- **Delivery [21].** Delivery is from JD Logistics, which includes 3 months of delivery data from Jun. 1st, 2019 to Aug. 31st, 2019 and covers a delivery region of 2km×2.4km in Beijing, China. The dataset consists of the parcel delivery order of each courier, which includes 5,167 delivery trips.
- **Tourism.** Tourism is collected via Flickr API[2], which includes geo-tagged photos uploaded by users from 2001 to 2021 of an 8km × 8km region in Melbourne, Australia. The dataset contains 1,734 users' check-in sequences.
- **LaDe [22].** LaDe is from Cainiao Network, which includes 6 months of last-mile delivery data from May. 1st, 2022 to Oct. 31st, 2022. The dataset also consists of the parcel delivery order of each courier, which includes 66,375 trips after preprocessing.

Figure 4 shows the distribution of the number of travel tasks and the number of workers on three datasets.



(a) # of Travel Tasks. (b) # of Workers.

Fig. 4. Data Distributions.

B. Experimental Settings

Problem Setup. Experiments are set up as follows.

- **Worker Setup:** The time for a courier to complete a delivery task is set to 10 minutes, and we assume tourists stay for 20 minutes at each POI. The movement speed of workers is set to 60 meters per minute.
- **Sensing Task Setup:** The sensing tasks are uniformly created in the spatio-temporal space. In Delivery and LaDe, the time span of the sensing project is set to 4 hours to synchronize with the working hours of delivery trips. In Tourism, the time span is set to 6 hours based on our analysis of the tourists' activities. The region in Delivery, Tourism, and LaDe is partitioned into a 10 × 12, 10 × 10 and 10 × 10 grid, respectively.
- **Other Setup:** The budget is set to 300 by default, and the incentive per unit time μ is 1. The time window of a sensing task is set to 30 minutes by default. In the objective function, the trade-off parameter α is 0.5 by default, which means that both data amount and data balance are equally important.

[2]https://www.flickr.com/services/api/

Baselines. Though USMDW is the general case of [8], the method of the latter cannot be used due to multiple mandatory visits. Therefore, we designed three heuristic algorithms and two modified meta-heuristic algorithms for comparison.

- **Random algorithm (RN):** Since a completely random selection algorithm may generate infeasible working routes. Therefore, for each worker, we first design a working route based on its origin, final destination as well as travel tasks via the Nearest Neighbor algorithm [10], i.e., we always select the nearest location as the next location. Based on the initial working route of each worker, we can iteratively randomly select a worker, randomly select a sensing task, and randomly select a position of the original working route to insert the sensing task, until the budget is used up.
- **Task value priority greedy algorithm (TVPG):** TVPG also leverages the Nearest Neighbor algorithm to generate the initial working route, and then the worker who contributes the most data coverage is selected [8]. After that, we calculate the maximum coverage gain for each sensing task if it is assigned to the worker by trying to insert it into any consecutive locations in the route. The sensing task with the highest maximum coverage gain is selected. If there are multiple sensing tasks with the same maximum coverage gain, we select the lowest incentive cost one.
- **Task cost priority greedy algorithm (TCPG):** The only difference between TCPG and TVPG is that we give priority to the sensing task with the lowest incentive cost when selecting the sensing task, and if there are multiple sensing tasks with the same incentive cost, the sensing task with the highest coverage gain is selected.
- **Multi-start simulated annealing (MSA):** MSA is a meta-heuristic algorithm which is proposed to solve TOPTW-MV [9]. MSA searches for a better solution by swapping, inserting and reversing from the current solution. To adapt MSA, we modify the rules to generate neighborhood solutions by checking whether the operation violates the constraints of USMDW, e.g., moving the travel task of a worker to another. If it happens, we redo a new operation (until a legal neighborhood solution is generated).
- **Multi-start simulated annealing with Greedy Initialization (MSAGI):** In MSAGI, we use the results of TVPG to construct initial solutions, rather than the random initialization in MSA.
- **JDRL [23]:** JDRL is a Multi-Agent Reinforcement Learning (MARL) framework used in ride-hailing platforms for order dispatching and driver repositioning. We adapt it by beginning to assign sensing tasks under the prerequisite that all travel tasks can be completed.

Evaluation Metrics. Both effectiveness and efficiency are evaluated. Since USMDW is an optimization problem, the objective, i.e., hierarchical entropy-based data coverage, is used to evaluate the effectiveness.

Training Details & Hyperparameters. In USMDW, multiple multi-destination workers should participate in the project. Therefore, to construct problem instances, we group users by

TABLE I
EFFECT OF SENSING TASK TIME WINDOW

Method	Delivery						Tourism						LaDe					
	Interval=30		Interval=60		Interval=120		Interval=30		Interval=60		Interval=120		Interval=30		Interval=60		Interval=120	
	Obj.	Time	Obj.	Time	Obj.	Time	Obj.	Time	Obj.	Time	Obj.	Time	Obj.	Time	Obj.	Time	Obj.	Time
RN	4.882	5 (s)	4.819	3 (s)	4.569	2 (s)	4.083	3 (s)	3.647	2 (s)	3.610	1 (s)	5.133	5 (s)	5.123	5 (s)	5.203	5 (s)
TVPG	6.129	5 (m)	6.047	3 (m)	5.926	2 (m)	5.175	2 (m)	4.978	1 (m)	4.948	1 (m)	5.857	2 (m)	5.907	2 (m)	5.887	1 (m)
TCPG	5.993	8 (m)	6.012	5 (m)	5.908	3 (m)	5.014	2 (m)	4.989	2 (m)	4.852	1 (m)	5.752	2 (m)	5.702	2 (m)	5.742	1 (m)
MSA	5.305	1 (h)	5.106	50 (m)	4.911	25 (m)	4.219	45 (m)	4.003	45 (m)	4.146	35 (m)	5.211	45 (m)	5.373	40 (m)	5.438	30 (m)
MSAGI	6.153	20 (m)	6.050	18 (m)	5.939	14 (m)	5.182	13 (m)	5.096	15 (m)	5.001	12 (m)	5.860	12 (m)	5.908	11 (m)	5.891	8 (m)
JDRL	5.461	3 (s)	5.295	3 (s)	5.018	2 (s)	5.126	1 (s)	5.101	1 (s)	4.709	1 (s)	5.346	2 (s)	5.26	2 (s)	5.007	1 (s)
SMORE	**6.296**	8 (s)	**6.195**	6 (s)	**6.136**	5 (s)	**5.385**	5 (s)	**5.463**	5 (s)	**5.283**	3 (s)	**6.005**	6 (s)	**6.036**	6 (s)	**6.189**	5 (s)

trip time intervals. In Delivery, the number of instances for training, validation, and testing is 120, 20, and 20 respectively. In Tourism, the number of instances for training, validation, and testing is 100, 10, and 10, respectively. In LaDe, the number of instances for training, validation, and testing is 13,327, 1,665, and 1,664 respectively. In worker and sensing task representation, the encoder contains 3 attention layers, each of which contains 8 heads. We use the Adam [24] optimizer, and the initial learning rate is set to 1e-4. We sample actions from the predicted probability distribution during training and select the action with the highest probability during validation and testing. The hyperparameter λ in the soft mask function is set to 0.5. In MSA & MSAGI, we set the number of starting points to 3. The initial temperature is set to 3.0 and the decay rate is set to 0.9. The algorithm iterates 3000 times per round and the algorithm ends when no better solution is found in 10 consecutive rounds. We limit the running time of the algorithm to a maximum of 1 hour for each problem instance.

Implementations. Our algorithms are implemented in Python with PyTorch. Experiments were conducted on a server with 8 Cores@3.0GHz, 56GB memory, and models were trained using a GeForce RTX 3090(24GB) GPU.

C. Experimental Results

Our experiments aim to find out:

1) Can we achieve higher data coverage with SMORE compared to baselines under settings of different sensing task time window, budget and data coverage weights (Section V-C1, Section V-C2 and Section V-C3)?
2) How is the running efficiency of SMORE (Section V-C4)?
3) What is the importance of main designs of SMORE, i.e., reinforcement-learning-based sensing task assignment, TASNet and the soft mask (Section V-C5)?

1) Effect of Sensing Task Time Window: Although we would like to collect spatio-temporal data as fine-grained as possible, finer-grained spatio-temporal segmentation also brings a larger problem scale. We fixed the budget to 300 and the α in data coverage to 0.5, and varied the time windows of sensing tasks to 30, 60 and 120 minutes, respectively, and the results of our experiments are shown in Table I. Although RN is efficient, it performs the worst as expected. The results of TVPG are similar to those of TCPG, but due to TVPG prioritizing the coverage gain of tasks, it outperforms TCPG in data coverage. MSA performs poorly, we guess the reason is that MSA is not carefully designed for mandatory visits of heterogeneous workers and our problem scale is much larger given massive sensing tasks. JDRL also performs worse than SMORE, since it is not specifically designed to consider the budget constraints and multi-destination mobility needs, which do not exist in ride-hailing. The experimental results prove that SMORE can adapt well to sensing tasks of different scales. SMORE outperforms the best baseline by 3.3%, 7.1%, 5.1% in data coverage, and the running time compared to the best non-RL-based baseline is saved by 168 times, 180 times, 96 times on Delivery, Tourism and LaDe, respectively. On average, SMORE outperforms the best baseline by 5.2%, while the running time compared to the best non-RL baseline is saved by about 150 times.

2) Effect of Budget: We fixed the sensing task time window to 30 minutes and the weight in data coverage to 0.5 and varied the budget from 200 to 400. The experimental results are shown in Table II. A higher budget allows for higher data coverage, meaning higher quality of sensed urban data. However, in the real world, the budget is often tight, which makes effective sensing task assignment and working route planning more important. In addition, a higher budget also allows us to recruit more workers to participate in the urban sensing project, which makes it more difficult to handle competition and cooperation among workers. From the experimental results, we can observe that as the budget increases, the data coverage of the collected data also increases. SMORE outperforms the best baseline by 3.1%, 5.4%, 3.4% in data coverage, and the running time compared to the best non-RL-based baseline is saved by 180 times, 240 times, 150 times on Delivery, Tourism and LaDe, respectively. Note that the gap of data coverage between each algorithm narrows as the budget increases, this is because the hierarchical entropy-based data coverage is non-linear, i.e., as the data continues to be collected, the increase of the data coverage becomes slow.

3) Effect of Weights in Data Coverage: By varying the weight α in data coverage ϕ, we can achieve different urban data collection purposes. The experimental results are shown in Table III. For α =0.2, which means that we care about the amount of data more than the degree of the balance of data,

TABLE II
EFFECT OF BUDGET

Method	Delivery						Tourism						LaDe					
	Budget=200		Budget=300		Budget=400		Budget=200		Budget=300		Budget=400		Budget=200		Budget=300		Budget=400	
	Obj.	Time	Obj.	Time	Obj.	Time	Obj.	Time	Obj.	Time	Obj.	Time	Obj.	Time	Obj.	Time	Obj.	Time
RN	4.261	4 (s)	4.882	5 (s)	5.374	7 (s)	3.372	1 (s)	4.083	3 (s)	4.478	5 (s)	4.289	2 (s)	5.133	5 (s)	5.462	7 (s)
TVPG	5.609	3 (m)	6.129	5 (m)	6.412	6 (m)	4.669	2 (m)	5.175	2 (m)	5.647	3 (m)	5.509	2 (m)	5.857	2 (m)	6.334	4 (m)
TCPG	5.341	6 (m)	5.993	8 (m)	6.307	11 (m)	4.585	2 (m)	5.014	2 (m)	5.615	4 (m)	5.116	2 (m)	5.752	2 (m)	6.112	5 (m)
MSA	4.482	52 (m)	5.305	1 (h)	5.583	1 (h)	3.818	40 (m)	4.219	45 (m)	4.817	54 (m)	4.991	35 (m)	5.211	45 (m)	5.634	50 (m)
MSAGI	5.615	18 (m)	6.153	20 (m)	6.464	23 (m)	4.708	12 (m)	5.182	13 (m)	5.697	15 (m)	5.521	10 (m)	5.86	12 (m)	6.358	15 (m)
JDRL	4.665	2 (s)	5.461	3 (s)	6.029	4 (s)	4.816	1 (s)	5.126	1 (s)	5.509	2 (s)	4.948	2 (s)	5.346	2 (s)	5.731	3 (s)
SMORE	**5.788**	6 (s)	**6.296**	8 (s)	**6.576**	11 (s)	**5.075**	3 (s)	**5.385**	5 (s)	**5.802**	8 (s)	**5.706**	4 (s)	**6.005**	6 (s)	**6.411**	12 (s)

TABLE III
EFFECT OF WEIGHT IN DATA COVERAGE

Method	Delivery						Tourism						LaDe					
	α=0.2		α=0.5		α=0.8		α=0.2		α=0.5		α=0.8		α=0.2		α=0.5		α=0.8	
	Obj.	Time	Obj.	Time	Obj.	Time	Obj.	Time	Obj.	Time	Obj.	Time	Obj.	Time	Obj.	Time	Obj.	Time
RN	4.500	5 (s)	4.882	5 (s)	5.352	6 (s)	3.864	4 (s)	4.083	3 (s)	4.446	3 (s)	4.588	2 (s)	5.113	5 (s)	5.449	8 (s)
TVPG	5.491	4 (m)	6.129	5 (m)	6.704	4 (m)	4.580	2 (m)	5.175	2 (m)	5.677	2 (m)	5.368	2 (m)	5.857	2 (m)	6.508	2 (m)
TCPG	5.607	7 (m)	5.993	8 (m)	6.386	8 (m)	4.716	3 (m)	5.014	2 (m)	5.356	3 (m)	5.512	4 (m)	5.752	2 (m)	6.465	3 (m)
MSA	4.807	54 (m)	5.305	1 (h)	5.502	40 (m)	4.142	1 (h)	4.219	45 (m)	4.678	55 (m)	4.896	30 (m)	5.211	45 (m)	5.676	55 (m)
MSAGI	5.511	16 (m)	6.153	20 (m)	6.709	14 (m)	4.615	16 (m)	5.182	13 (m)	5.737	12 (m)	5.369	15 (m)	5.860	12 (m)	6.513	10 (m)
JDRL	4.926	3 (s)	5.461	3 (s)	5.909	3 (s)	4.644	1 (s)	5.126	1 (s)	5.650	1 (s)	4.795	2 (s)	5.346	2 (s)	5.67	2 (s)
SMORE	**5.712**	9 (s)	**6.296**	8 (s)	**6.803**	8 (s)	**4.905**	4 (s)	**5.385**	5 (s)	**5.815**	8 (s)	**5.531**	4 (s)	**6.005**	6 (s)	**6.625**	11 (s)

TCPG outperforms TVPG. This may be attributed to the cost priority in TCPG, which indicates it is able to finish more tasks even if the value is small. In the scenario when α is small, such a strategy shows more advantage. SMORE outperforms the best baseline by 2.3%, 4.0%, 3.0% in data coverage, and the running time compared to the best non-RL-based baseline is saved by 150 times, 156 times, 120 times on Delivery, Tourism and LaDe, respectively. The experimental results prove that our method is a universal strategy for different data collection purposes.

4) Running Time: The running time of each algorithm under different experimental settings is also shown in Table I, Table II, and Table III. As a meta-heuristic algorithm, MSA takes dozens of minutes to complete its calculation. In contrast, MSAGI utilizes greedy results to initialize the solution, reducing the running time to slightly more than ten minutes to obtain the final result. TVPG and TCPG are both greedy heuristic algorithms that require coverage gain and incentive cost calculations for all candidate assignments during each selection, which brings additional computational costs. RN can be completed in just a few seconds since it does not consider any heuristic about task selection at all, resulting in little computational cost. Benefiting from the RL-based decision, JDRL and SMORE can complete the calculation in a few seconds, but SMORE has better data coverage gain.

5) Ablation Study: We evaluate the effect of each component of SMORE by removing each of our main designs: Reinforcement Learning-based sensing task assignment (w/o RL-AS), two-stage assignment selection network (w/o TASNet), and soft mask function (w/o Soft Mask). The experimental results are shown in Figure 5. We can observe that our RL-based sensing task assignment shows an advantage over the greedy-based one (w/o RL-AS) since the latter is myopic and not able to achieve the best overall performance in the long term. In w/o TASNet, the sensing task-worker assignment pairs are obtained directly through a single selection, the performance of which is even worse than the greedy-based one. It indicates that simply applying RL to our problem does not work well due to the large action space and the under-utilization of information, while TASNet tackles those issues. By using the soft mask function empowered by the heuristics, we enhance the exploration efficiency of the agent and thus are able to obtain a higher data coverage.

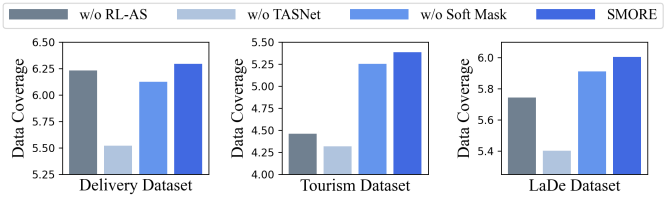
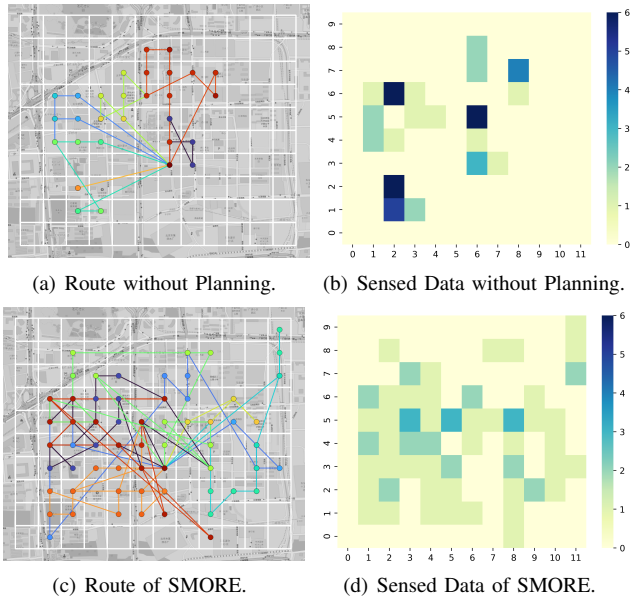


Fig. 5. Ablation Study

D. Case Study

To further show the advantage of SMORE, we give a case study of the studied region. Figure 6(a) and Figure 6(b) show



(a) Route without Planning. (b) Sensed Data without Planning.

(c) Route of SMORE. (d) Sensed Data of SMORE.

Fig. 6. Case Study.

the actual movement routes and the completion of the sensing tasks for the case if we do not re-plan the movement routes for workers, i.e., the workers only perform sensing tasks along the routes. It can be observed that the distribution of workers does not actually cover the entire sensing region, and thus the sensed data are also distributed highly skewed over the spatio-temporal landscape, which is not ideal for urban sensing. The results produced by SMORE are shown in Figures 6(c) and Figure 6(d), which indicate we can collect much more balanced urban data by guiding workers to complete sparsely distributed sensing tasks while guaranteeing their own travel tasks can still be completed.

VI. Related Work

A. Spatial Crowdsourcing (SC)

Spatial crowdsourcing has attracted substantial attention in recent years [25]–[30], and one of the most important problems in spatial crowdsourcing is the task assignment.

Task assignment problems can be categorized into matching models or planning models [31]. In matching models, task assignment can be formulated as a bipartite graph-based problem and the goal is to obtain an optimal matching in the bipartite graph [32], [33]. In planning models, the platform requires planning an actual working route for each worker, which is the case for task assignment in some real-world applications such as logistics delivery [34], food delivery, and ride-hailing [23], [35], [36]. Some works [37], [38] studied task assignment for a single worker who can be assigned many tasks. A more complex scenario is assigning many tasks to many workers like ours, including maximizing the total number of assignments [7], [35], [39]–[41], the total satisfaction of workers [42], [43], the data coverage of collected data [8], etc.

The ridersharing [35] and urban crowdsensing for commuters [8] are two problems close to USMDW: 1) As for the ridersharing [35], it also considers multiple destinations. The major differences between ridesharing and USMDW are the former is for the online scenario, which requests efficient algorithms for timely response, while the latter is for the offline scenario, which prefers optimal/near-optimal solutions to utilize the budget well. Specifically, in ridesharing, when a request is assigned to a taxi, most ridesharing algorithms are heuristic-based, which keep the existing traveling order unchanged and try to insert the new request, while in USMDW, when a sensing task is assigned to a worker, we recalculate the estimated optimal route based on all assigned tasks expecting to achieve a higher objective. Besides, there is no budget constraint in the ridersharing. 2) As for the urban crowdsensing for commuters [8], we have the same objective. However, there is no mandatory visits in [8], which makes the feasibility check much easier than ours. And the proposed method also cannot guarantee any mandatory visits. We generalize [8] to the multi-destination scenario and propose corresponding solutions.

B. Reinforcement Learning for Combinatorial Optimization

Many spatial crowdsourcing problems can be regarded as the orienteering problem [13]. Recent research in combinatorial optimization has made tremendous progress using deep reinforcement learning to learn effective policies on large-scale problems, which originated from the Pointer Network (PN) [44]. Bello et al. [18] combined PN with Reinforcement Learning, which allows models to be trained without supervised solutions. Kool et al. [10] proposed the Attention Model (AM), a variant of PN, to address TSP, VRP, and OP. In addition to AM, there are also some methods can tackle OP and its vairants [45]–[48]. Although our problem shares some similarities with TOPTW-MV [9], existing methods cannot directly solve our problem due to worker heterogeneity.

VII. Conclusion

In this paper, we generalize the urban crowdsensing problem to the scenario of multi-destination workers, and prove its NP-hardness. A RL-based framework SMORE is proposed to select sensing tasks and design working routes for them. SMORE first initializes all feasible sensing task-worker pairs efficiently by calling a pre-trained RL-based TSPTW solver, then employs a novel policy network, i.e., TASNet, to select sensing task-worker pairs iteratively. Experiments on two delivery and a tourism datasets demonstrate that SMORE outperforms the best baseline in data coverage by 5.2% on average with high efficiency.

In this work, we use the pre-trained TSPTW solver to perform the feasibility check, which may produce "false alarm". It is worthwhile to further incorporate the approximation errors to achieve better results in the future.

Acknowledgment

This work was supported by the National Key R&D Program of China (No. 2023YFC2308703), the National Natural Science Foundation of China (No. 62306033) and Beijing Institute of Technology Research Fund Program for Young Scholars (No. 6120220113).

REFERENCES

[1] Y. Zheng, *Urban computing*. MIT Press, 2019.

[2] Y. Cui, S. Li, W. Deng, Z. Zhang, J. Zhao, K. Zheng, and X. Zhou, "Roi-demand traffic prediction: A pre-train, query and fine-tune framework," in *ICDE*, pp. 1340–1352, 2023.

[3] T. He, J. Bao, Y. Li, H. He, and Y. Zheng, "Crowd-sensing enhanced parking patrol using sharing bikes trajectories," *TKDE*, 2021.

[4] A. Capponi, C. Fiandrino, B. Kantarci, L. Foschini, D. Kliazovich, and P. Bouvry, "A survey on mobile crowdsensing systems: Challenges, solutions, and opportunities," *IEEE communications surveys & tutorials*, vol. 21, no. 3, pp. 2419–2465, 2019.

[5] A. T. Campbell, S. B. Eisenman, N. D. Lane, E. Miluzzo, and R. A. Peterson, "People-centric urban sensing," in *Proceedings of the 2nd annual international workshop on Wireless internet*, pp. 18–es, 2006.

[6] J. A. Burke, D. Estrin, M. Hansen, A. Parker, N. Ramanathan, S. Reddy, and M. B. Srivastava, "Participatory sensing," 2006.

[7] Y. Zhao, Y. Li, Y. Wang, H. Su, and K. Zheng, "Destination-aware task assignment in spatial crowdsourcing," in *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, pp. 297–306, 2017.

[8] S. Ji, Y. Zheng, and T. Li, "Urban sensing based on human mobility," in *Proceedings of the 2016 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, pp. 1040–1051, 2016.

[9] S.-W. Lin and F. Y. Vincent, "Solving the team orienteering problem with time windows and mandatory visits by multi-start simulated annealing," *Computers & Industrial Engineering*, vol. 114, pp. 195–205, 2017.

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

[11] H. Wen, Y. Lin, F. Wu, H. Wan, Z. Sun, T. Cai, H. Liu, S. Guo, J. Zheng, C. Song, *et al.*, "Enough waiting for the couriers: Learning to estimate package pick-up arrival time from couriers' spatial-temporal behaviors," *ACM Transactions on Intelligent Systems and Technology*, vol. 14, no. 3, pp. 1–22, 2023.

[12] S. Ruan, C. Long, J. Bao, C. Li, Z. Yu, R. Li, Y. Liang, T. He, and Y. Zheng, "Learning to generate maps from trajectories," in *AAAI*, vol. 34, pp. 890–897, 2020.

[13] B. L. Golden, L. Levy, and R. Vohra, "The orienteering problem," *Naval Research Logistics (NRL)*, vol. 34, no. 3, pp. 307–318, 1987.

[14] M. Gendreau, A. Hertz, and G. Laporte, "New insertion and postoptimization procedures for the traveling salesman problem," *Operations Research*, vol. 40, no. 6, pp. 1086–1094, 1992.

[15] İ. Küçükoğlu, R. Dewil, and D. Cattrysse, "Hybrid simulated annealing and tabu search method for the electric travelling salesman problem with time windows and mixed charging rates," *Expert systems with applications*, vol. 134, pp. 279–303, 2019.

[16] Q. Ma, S. Ge, D. He, D. Thaker, and I. Drori, "Combinatorial optimization by graph pointer networks and hierarchical reinforcement learning," *arXiv preprint arXiv:1911.04936*, 2019.

[17] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *NeurIPS*, vol. 30, 2017.

[18] I. Bello, H. Pham, Q. V. Le, M. Norouzi, and S. Bengio, "Neural combinatorial optimization with reinforcement learning," *arXiv preprint arXiv:1611.09940*, 2016.

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

[20] R. S. Sutton, D. McAllester, S. Singh, and Y. Mansour, "Policy gradient methods for reinforcement learning with function approximation," *NeurIPS*, vol. 12, 1999.

[21] S. Ruan, C. Long, X. Yang, T. He, R. Li, J. Bao, Y. Chen, S. Wu, J. Cui, and Y. Zheng, "Discovering actual delivery locations from mis-annotated couriers' trajectories," in *ICDE*, pp. 3241–3253, IEEE, 2022.

[22] L. Wu, H. Wen, H. Hu, X. Mao, Y. Xia, E. Shan, J. Zhen, J. Lou, Y. Liang, L. Yang, R. Zimmermann, Y. Lin, and H. Wan, "Lade: The first comprehensive last-mile delivery dataset from industry," 2023.

[23] J. Sun, H. Jin, Z. Yang, L. Su, and X. Wang, "Optimizing long-term efficiency and fairness in ride-hailing via joint order dispatching and driver repositioning," in *KDD*, pp. 3950–3960, 2022.

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

[25] Y. Yang, Y. Cheng, Y. Yang, Y. Yuan, and G. Wang, "Batch-based cooperative task assignment in spatial crowdsourcing," in *ICDE*, pp. 1180–1192, 2023.

[26] X. Chen, Y. Zhao, K. Zheng, B. Yang, and C. S. Jensen, "Influence-aware task assignment in spatial crowdsourcing," in *ICDE*, pp. 2141–2153, IEEE, 2022.

[27] C. Shan, N. Mamoulis, R. Cheng, G. Li, X. Li, and Y. Qian, "An end-to-end deep rl framework for task arrangement in crowdsourcing platforms," in *ICDE*, pp. 49–60, IEEE, 2020.

[28] X. Zhou, S. Liang, K. Li, Y. Gao, and K. Li, "Bilateral preference-aware task assignment in spatial crowdsourcing," in *ICDE*, pp. 1687–1699, IEEE, 2022.

[29] Y. Zhao, K. Zheng, Z. Wang, L. Deng, B. Yang, T. B. Pedersen, C. S. Jensen, and X. Zhou, "Coalition-based task assignment with priority-aware fairness in spatial crowdsourcing," *The VLDB Journal*, pp. 1–22, 2023.

[30] D. Zhai, Y. Sun, A. Liu, Z. Li, G. Liu, L. Zhao, and K. Zheng, "Towards secure and truthful task assignment in spatial crowdsourcing," *World Wide Web*, vol. 22, pp. 2017–2040, 2019.

[31] Y. Tong, Z. Zhou, Y. Zeng, L. Chen, and C. Shahabi, "Spatial crowdsourcing: a survey," *The VLDB Journal*, vol. 29, pp. 217–250, 2020.

[32] Y. Zhao, K. Zheng, J. Guo, B. Yang, T. B. Pedersen, and C. S. Jensen, "Fairness-aware task assignment in spatial crowdsourcing: Game-theoretic approaches," in *ICDE*, pp. 265–276, IEEE, 2021.

[33] T. Ren, X. Zhou, K. Li, Y. Gao, J. Zhang, and K. Li, "Efficient cross dynamic task assignment in spatial crowdsourcing," in *ICDE*, pp. 1420–1432, 2023.

[34] S. Ruan, C. Long, Z. Ma, J. Bao, T. He, R. Li, Y. Chen, S. Wu, and Y. Zheng, "Service time prediction for delivery tasks via spatial meta-learning," in *KDD*, pp. 3829–3837, 2022.

[35] Z. Liu, Z. Gong, J. Li, and K. Wu, "Mobility-aware dynamic taxi ridesharing," in *ICDE*, pp. 961–972, 2020.

[36] X. Tang, Z. Qin, F. Zhang, Z. Wang, Z. Xu, Y. Ma, H. Zhu, and J. Ye, "A deep value-network based approach for multi-driver order dispatching," in *KDD*, pp. 1780–1790, 2019.

[37] C. F. Costa and M. A. Nascimento, "In-route task selection in crowdsourcing," in *Proceedings of the 26th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, pp. 524–527, 2018.

[38] D. Deng, C. Shahabi, U. Demiryurek, and L. Zhu, "Task selection in spatial crowdsourcing from worker's perspective," *GeoInformatica*, vol. 20, pp. 529–568, 2016.

[39] Y. Zhao, K. Zheng, Y. Li, H. Su, J. Liu, and X. Zhou, "Destination-aware task assignment in spatial crowdsourcing: A worker decomposition approach," *TKDE*, vol. 32, no. 12, pp. 2336–2350, 2019.

[40] C. H. Liu, Y. Zhao, Z. Dai, Y. Yuan, G. Wang, D. Wu, and K. K. Leung, "Curiosity-driven energy-efficient worker scheduling in vehicular crowdsourcing: A deep reinforcement learning approach," in *ICDE*, pp. 25–36, IEEE, 2020.

[41] S. Ruan, J. Bao, Y. Liang, R. Li, T. He, C. Meng, Y. Li, Y. Wu, and Y. Zheng, "Dynamic public resource allocation based on human mobility prediction," *Proceedings of the ACM on interactive, mobile, wearable and ubiquitous technologies*, vol. 4, no. 1, pp. 1–22, 2020.

[42] J. She, Y. Tong, and L. Chen, "Utility-aware social event-participant planning," in *Proceedings of the 2015 ACM SIGMOD international conference on management of data*, pp. 1629–1643, 2015.

[43] D. Gao, Y. Tong, Y. Ji, and K. Xu, "Team-oriented task planning in spatial crowdsourcing," in *APWeb-WAIM*, pp. 41–56, Springer, 2017.

[44] O. Vinyals, M. Fortunato, and N. Jaitly, "Pointer networks," *NeurIPS*, vol. 28, 2015.

[45] W. Liu, T. Zhang, R. Wang, K. Li, W. Li, and K. Yang, "Deep reinforcement learning for orienteering problems based on decomposition," *arXiv preprint arXiv:2204.11575*, 2022.

[46] R. Gama and H. L. Fernandes, "A reinforcement learning approach to the orienteering problem with time windows," *Computers & Operations Research*, vol. 133, p. 105357, 2021.

[47] W. Liu, R. Wang, T. Zhang, K. Li, W. Li, and H. Ishibuchi, "Hybridization of evolutionary algorithm and deep reinforcement learning for multi-objective orienteering optimization," *IEEE Transactions on Evolutionary Computation*, 2022.

[48] P. Sankaran, K. McConky, M. Sudit, and H. Ortiz-Pena, "Gamma: graph attention model for multiple agents to solve team orienteering problem with multiple depots," *TNNLS*, 2022.