Decision-Oriented Learning with Differentiable Submodular Maximization for Vehicle Routing Problem

Guangyao Shi, Pratap Tokekar

***Abstract*— We study the problem of learning a function that maps context observations (input) to parameters of a submodular function (output). Our motivating case study is a specific type of vehicle routing problem, in which a team of Unmanned Ground Vehicles (UGVs) can serve as mobile charging stations to recharge a team of Unmanned Ground Vehicles (UAVs) that execute persistent monitoring tasks. We want to learn the mapping from observations of UAV task routes and wind field to the parameters of a submodular objective function, which describes the distribution of landing positions of the UAVs . Traditionally, such a learning problem is solved independently as a prediction phase without considering the downstream task optimization phase. However, the loss function used in prediction may be misaligned with our final goal, i.e., a good routing decision. Good performance in the isolated prediction phase does not necessarily lead to good decisions in the downstream routing task. In this paper, we propose a framework that incorporates task optimization as a differentiable layer in the prediction phase. Our framework allows end-to-end training of the prediction model without using engineered intermediate loss that is targeted only at the prediction performance. In the proposed framework, task optimization (submodular maximization) is made differentiable by introducing stochastic perturbations into deterministic algorithms (i.e., stochastic smoothing). We demonstrate the efficacy of the proposed framework using synthetic data. Experimental results of the mobile charging station routing problem show that the proposed framework can result in better routing decisions, e.g. the average number of UAVs recharged increases, compared to the prediction-optimization separate approach.**

I. Introduction

Many multi-robot decision-making problems can be formulated as combinatorial optimization problems, among which the objectives in some problems (e.g., mutual information [1], area explored, number of targets tracked [2], detection probability [3] etc.) have diminishing returns property i.e., submodularity. Intuitively, submodularity formalizes the notion that adding more robots to a larger multi-robot team cannot yield a smaller marginal gain in the objective than adding the same robot to a smaller team.

If the submodular objective is known and fixed, the multi-robot decision-making problem boils down to a submodular maximization problem, which is NP-hard but can be solved with an $(1-\frac{1}{e})$-approximation by the greedy algorithm [4]. However, in practice, there are several parameters that affect the objective function that may not be known exactly.

Consider the following illustrative example of a vehicle routing problem shown in Figure 2. Here, a team of Unmanned Ground Vehicles (UGVs) are tasked with servicing a set of requests that appear throughout the environment. We have a set of candidate routes of which we must select one for each UGV. The objective is to maximize the number of requests serviced. If a request location lies on more than one UGVs path (the paths may overlap as the UGVs move on a road network), it only counts once in the objective function. Thus, the objective function is a coverage function, which is a special case of the submodular function.

This work is supported in part by National Science Foundation Grant No. 1943368 and Army Grant No. W911NF2120076.

Guangyao Shi, Pratap Tokekar are with the University of Maryland, College Park, MD 20742 USA `[gyshi, tokekar]@umd.edu`

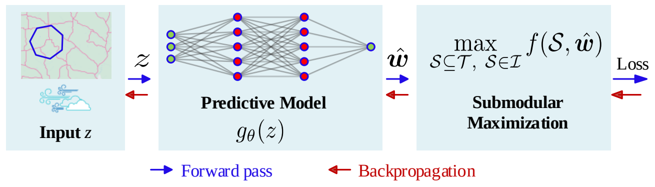


Fig. 1. Decision-Oriented Learning framework. The training loss is defined after the downstream task.

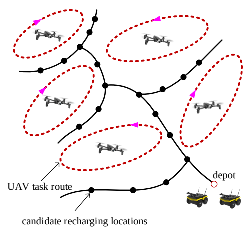
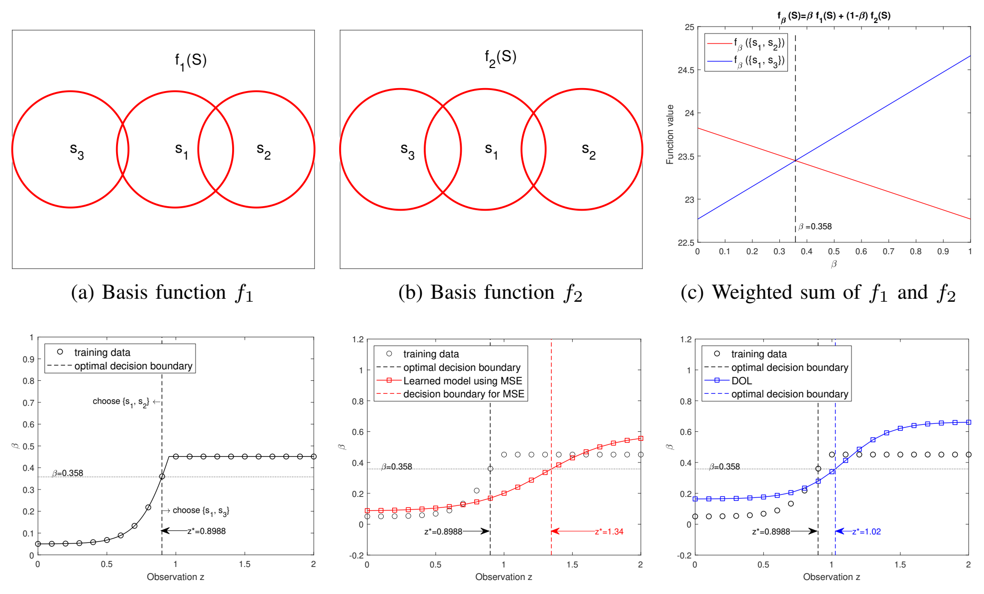


Fig. 2. An illustrative example for vehicle routing problems.

If we know the location of the requests, then we can solve this problem greedily to obtain a $(1-\frac{1}{e})$–approximation. The greedy algorithm requires the capability to compute the objective function $f(S)$. However, there are many scenarios where we may not know where the requests show up and as such not know $f(S)$. For example, the requests could correspond to Unmanned Aerial Vehicles (UAVs) that are carrying out persistent monitoring missions that land when out of charge so as to be recharged by mobile recharging stations [5]–[10]. Here, even if we know the routes followed by each UAV, we may not know their exact landing locations since the energy consumption is stochastic [9], [10] and communication between UAVs and UGVs is not available (e.g., due to stealth). In such cases, we may be able to predict $f(S)$ using all the available information. We call the latter as *context* z, which can include the routes of the UAVs, the environmental conditions including the wind conditions, etc.

The traditional pipeline here would be to use the context information and predict $f(S)$ and then solve the downstream



(a) Basis function f_1 (b) Basis function f_2 (c) Weighted sum of f_1 and f_2

(d) Ground truth data (e) Learned model using MSE (f) Decision-Oriented-Learning results

Fig. 3. An illustrative example to show the misalignment between the prediction model that achieves high predictive accuracy and the one that results in good decisions.

UGV route selection problem, $\operatorname{argmax}_S \hat{f}(S)$, using this predicted $\hat{f}(S)$. However, as the following example shows a good predictor of $f(S)$ does not necessarily align with making good decisions on the downstream task. On the other hand, a predictor that does not necessarily yield the best predictions of $f(S)$ may still yield the best decisions for the downstream $\operatorname{argmax}_S \hat{f}(S)$ problem.

We present an illustrative example of such misalignment in Fig. 3. Let f_1, f_2 be two coverage functions (coverage function is submodular by definition) defined over set $\{s_1, s_2, s_3\}$. Given a subset $\mathcal{S} \subseteq \{s_1, s_2, s_3\}$, $f_i(\mathcal{S}), i = 1, 2$ will return the area covered by the selection. The submodular objective that we are interested in is defined as $f_\beta(\mathcal{S}) = \beta f_1(\mathcal{S}) + (1-\beta) f_2(\mathcal{S}), \beta \in [0, 1]$, which is also submodular by definition. Suppose that we want to maximize f_β with a partition matroid: $|\mathcal{S} \cap \{s_1\}| \leq 1$, $|\mathcal{S} \cap \{s_2, s_3\}| \leq 1$. Then the optimal solution is either $\{s_1, s_2\}$ or $\{s_1, s_3\}$. In Fig. 3c, we show how the optimal decision changes w.r.t. β. When $\beta \geq 0.358$, the optimal decision is $\{s_1, s_3\}$ since $f_\beta(\{s_1, s_3\}) \geq f_\beta(\{s_1, s_2\})$ (the blue line is above the red line). By contrast, when $\beta < 0.358$, the optimal decision is $\{s_1, s_2\}$. Next, let us look at the learning problem for f_β. We want to find a mapping from the observation z to β. In Fig. 3d, we show the training data sampled from the ground truth and the optimal decision boundary $z^* = 0.8988$, which is obtained by finding the intersection between the ground truth curve and $\beta = 0.358$. If we use Mean Square Error (MSE) as the objective for learning without considering the downstream task, we will get two lines as shown in Fig. 3e. The decision boundary (dashed vertical red line, $z^* = 1.34$, passing the intersection of the learned red line and $\beta = 0.358$) is on the right of the optimal boundary, thus not optimal. By contrast, if we consider the downstream optimization, we will get two lines as shown in Fig. 3f and the decision boundary (dashed vertical blue line, $z^* = 1.02$, passing the intersection of the learned blue line and $\beta = 0.358$) is closer to the optimal boundary, thus reducing the regions of suboptimal decisions. Such an observation motivates us to incorporate the decision process (submodular maximization) into the learning process.

To this end, we propose a Decision-Oriented-Learning (DOL) framework for learning context-aware parameterized submodular objectives. We focus on submodular functions that can be parameterized, i.e., $f(S, \boldsymbol{w})$, where the parameters are to be learned from the context. As described earlier and pointed out in [11]–[13], the best estimator of $\boldsymbol{w}$ does not necessarily yield the best decisions for the downstream task. Instead, in the proposed framework, the decision-making problem (submodular maximization) is treated as a differentiable layer that takes as input the output from the prediction module as shown in Fig. 1. The prediction module takes as input a context observation z and predicts $\boldsymbol{w}$, i.e., the parameterized submodular function. By using a differentiable submodular optimization layer, we can train the prediction module using the loss from the downstream task, thereby yielding aligned predictions.

In summary, the main contribution is this paper is:

- We propose a decision-oriented learning framework for mobile charging routing problems. We show how to formulate the learning problem for mobile charging routing and how to solve it.
- We demonstrate the effectiveness of our framework in several examples through simulation.

The rest of the paper is organized as follows. We first give a brief overview of the related work in Section II. Then, we explain the problem setup and formulation in Section III. We introduce the learning algorithm in Section V and validate the formulation and the proposed framework in Section VI.

II. Related Work

Most existing multi-robot decision-making work consider the case where the optimization objective is well-defined and known. Wilde et al. [14] consider the case where the optimization objective is hard to quantitatively specify and may be subjective and proposed an interactive learning framework to learn the objectives. Our work shares a similar stance with [14] but differs in two aspects. First, we consider the fact that the task objective may change in different contexts, for example in different weather conditions, and aim at learning a context-aware objective. Second, our learning framework integrates the downstream decision-making process into the learning process.

Another line of research related to this work is decision-oriented learning. The key idea is to embed the decision-making problem as a differentiable layer in the learning pipelines. The main advantage is that it allows end-to-end training and reduces the engineering efforts to design some intermediate learning objectives. Such an idea was initially explored for continuous optimization problems [15], [16] and has gained popularity in control and robotics [17]–[20]. The idea was later extended to the combinatorial problems [12], [13], [21], [22]. Our work is inspired by [12], [21] and our framework integrates the decision-making process for mobile charging station routing, which is modeled as submodular maximization, into the learning process.

This work is also closely related to differentiable submodular maximization. Submodular maximization and its variants have been widely used in multi-robot decision-making problems including coverage, target tracking, exploration, and information gathering. These studies are all based on the fact that the greedy algorithm and its variants can solve submodular maximization problems its variants efficiently with a provable performance guarantee. Since the submodular objective and greedy algorithm are tightly coupled, it is better to take into account the influence of the greedy algorithm when we consider learning submodular functions [23]. To this end, several differentiable versions of the greedy algorithms have been proposed [23], [24]. The core idea behind these algorithms is stochastic smoothing, i.e., perturb the algorithm by introducing some probability distribution in the intermediate steps. Our framework is built on these differentiable greedy algorithms but is targeted specifically for context-dependent routing problems.

III. Problem Formulation

In this section, we first introduce the formulation of the learning problem. Then, we explain the setup of the case study and the parameterization of the objective.

We are interested in parameterized submodular objective function $f(\mathcal{S}, \boldsymbol{w})$, where $\boldsymbol{w}$ is the parameter vector. Readers are referred to [14] for a formal definition. In practice, such an objective is usually unknown and context-dependent, i.e., the parameters $\boldsymbol{w} \in \mathcal{W}$ depend on the environment features. Our goal is to learn a function $g_{\boldsymbol{\theta}} : \mathcal{Z} \to \mathcal{W}$ that maps the context observation $z \in \mathcal{Z}$ to the objective parameters $\boldsymbol{w}$. Traditionally, finding the mapping $g_{\boldsymbol{\theta}}$ and optimizing the downstream objective $f(S, \boldsymbol{w})$ are considered separately: given the training data $\mathcal{D} = \{(\boldsymbol{z}_1, \boldsymbol{w}_1), (\boldsymbol{z}_2, \boldsymbol{w}_2), \ldots, (\boldsymbol{z}_{|\mathcal{D}|}, \boldsymbol{w}_{|\mathcal{D}|})\}$, first find the mapping $g_{\boldsymbol{\theta}}$ by optimizing over $\boldsymbol{\theta}$ in a supervised fashion, and then use the parameter $\boldsymbol{w} = g_{\theta}(\boldsymbol{z})$ to optimize $f(S, \boldsymbol{w})$.

By contrast, the proposed paradigm that integrates downstream optimization is given below.

Problem 1. *Given the training data $\mathcal{D} = \{(\boldsymbol{z}_1, \boldsymbol{w}_1), (\boldsymbol{z}_2, \boldsymbol{w}_2), \ldots, (\boldsymbol{z}_{|\mathcal{D}|}, \boldsymbol{w}_{|\mathcal{D}|})\}$, learn a function $g_{\boldsymbol{\theta}}$ parameterized by $\boldsymbol{\theta}$ such that the learning cost $L = \frac{1}{|\mathcal{D}|}\sum_{i=1}^{|\mathcal{D}|} \ell_i(\boldsymbol{w}_i, \hat{\boldsymbol{w}}_i)$ is minimized, where $\ell_i(\boldsymbol{w}_i, \hat{\boldsymbol{w}}_i)$ is defined through Eq.* (1) *to Eq.* (3)*:*

$$\hat{\boldsymbol{w}}_i := g_{\boldsymbol{\theta}}(\boldsymbol{z}_i) \tag{1}$$

$$\hat{\mathcal{S}} := \mathcal{S}^*(\hat{\boldsymbol{w}}_i) \quad \textit{by solving } (5) \textit{ with } \boldsymbol{w} = \hat{\boldsymbol{w}}_i \tag{2}$$

$$\ell_i(\hat{\boldsymbol{w}}_i, \boldsymbol{w}_i) := f(\mathcal{S}^*(\boldsymbol{w}_i), \boldsymbol{w}_i) - f(\hat{\mathcal{S}}, \boldsymbol{w}_i), \tag{3}$$

where $\mathcal{S}^*(\boldsymbol{w}_i)$ denotes the solution of (5) *returned by some approximation algorithms with $\boldsymbol{w} = \boldsymbol{w}_i$; $f(\mathcal{S}^*(\boldsymbol{w}_i), \boldsymbol{w}_i)$ denotes the decision quality when we use the ground truth parameter $\boldsymbol{w}_i$ for decisions; $f(\hat{\mathcal{S}}, \boldsymbol{w}_i)$ denotes the decision quality when we use the predicted parameter $\hat{\boldsymbol{w}}_i$ for decisions, i.e., use $\hat{\boldsymbol{w}}_i$ to obtain the decision $\hat{\mathcal{S}}$, but the decision is evaluated w.r.t. the true parameter $\boldsymbol{w}_i$.*

The intuition for Eq. (3) is that we want to minimize the gap between the decision quality of the true parameters and that of the predicted parameters. One challenge is when we use the chain rule to compute the gradient of the loss function, we need to differentiate through the optimization problem (the first term on the r.h.s. of Eq. (4)) as shown in the illustrative computational graph in Fig. 1.

$$\frac{\partial \ell_i}{\partial \boldsymbol{\theta}} = \frac{\partial \ell_i}{\partial \hat{\boldsymbol{w}}_i} \cdot \frac{\partial \hat{\boldsymbol{w}}_i}{\partial \boldsymbol{\theta}} \tag{4}$$

In the following sections, we will show how to approximately compute the first term on the r.h.s. of Eq. (4).

IV. Case Study

Suppose that there is a set of candidate routes, $\mathcal{T}$, each of which starts and terminates at the same depot. Our goal is to select a subset from $\mathcal{T}$ for UGVs to traverse and recharge the UAV along the way such that the total number of UAVs that UGVs will recharge is maximized.

Environment Model: As shown in Fig. 2, the working environment is described by an area $\mathcal{E} \subseteq \mathbb{R}^2$. There are n_a UAVs that are executing persistent monitoring. The energy consumption of UAVs will be affected by the wind. The wind field is represented as a tuple $(\boldsymbol{\omega}_s, \omega_o)$, where $\boldsymbol{\omega}_s$ and ω_o denote the description vectors for the speed and the orientation of the wind, respectively. There are n_g UGVs in $\mathcal{E}$, denoted by the set $\{1, \ldots, n_g\}$.

UAV Behavior: There are three components defining the behavior of each UAV. The first one is the task route, which is defined as a sequence of ordered locations projected on the ground, and the UAV will persistently monitor these locations. The UAV will fly at a fixed speed v_a between two task locations and its energy consumption will be affected by the wind. The second component is the recharging strategy dealing with the depletion of the battery. The third component is the energy consumption model. We use the same model as that in [25].

Context Observation: Each observation $\boldsymbol{z}$ consists of two components: the task routes of all UAVs; and the wind field $(\boldsymbol{\omega_s}, \omega_o)$ of the working area.

If such submodular objective f is known, the problem boils down to a submodular maximization problem with a matroid constraint: let $\mathcal{T}$ be set of all candidate routes, the problem is to select a subset from $\mathcal{T}$ to maximize the objective, i.e.,

$$\max_{\mathcal{S}\subseteq\mathcal{T},\ |\mathcal{S}|\leq n_g} f(\mathcal{S}, \boldsymbol{w}), \tag{5}$$

where $\boldsymbol{w}$ denotes the parameters in the objective function.

Parameterization of the Objective Function In general, such applications have no closed-form expression of the objective function. In this paper, we consider the case where the objective function f is the linear combination of a set of basis functions. Such parameterization techniques are commonly used in the literature on learning submodular functions [3], [14], [26]. Similar to [14], we assume without loss of generality that each basis function is characterized by a subset $W_i \subseteq V$. That is, for any W_i, let $\psi_i(\mathcal{S})$ be a count of how many vertices of the tours $\mathcal{S}$ lie in W_i, then f_i is a functional of $\psi_i(\mathcal{S})$.

The overall objective function is:

$$f(\mathcal{S}, \boldsymbol{w}) = \sum_{i=1}^{n}\sum_{j=1}^{|\Gamma|} w_{i,j} f_{i,j}(\mathcal{S}), \tag{6}$$

where $\boldsymbol{w} = [w_{i,j}]$ denotes the matrix of unknown parameters of the function; basis functions are defined as $f_{i,j}(\mathcal{S}) = \sum_{\alpha=1}^{\psi_i(\mathcal{S})} \gamma_j^{(\alpha-1)}$; and $\gamma_j \in (0, 1]$ comes from a known set Γ.

V. Learning Algorithm

In this section, we describe the stochastic techniques to smoothe the greedy algorithm for submodular maximization and how can we apply the result to our framework. The key idea is: by introducing proper stochastic perturbances into combinatorial optimization, the expected output as a function of its parameters can be smoothed and differentiable.

A. Smoothed Greedy Algorithm

The Smoothed Greedy (SG) algorithm is given in Algorithm 1, which was first proposed in [24]. For a given $\boldsymbol{w} \in \mathcal{W}$, In each iteration step, we compute marginal gain $f_{\mathcal{S}}(u, \boldsymbol{w})$ for each candidate element $u \in U_k$ (line 3); we define $n_k := |U_k|$. Let $\boldsymbol{m}_k(\boldsymbol{w}) = (m_k(u_1, \boldsymbol{w}), m_k(u_2, \boldsymbol{w}), \ldots, m_k(u_{n_k}, \boldsymbol{w})) \in \mathbb{R}^{n_k}$ denote the marginal gain vector. The probability vector, $\boldsymbol{p}_k(\boldsymbol{w}) = (p_k(u_1, \boldsymbol{w}), \ldots, p_k(u_{n_k}, \boldsymbol{w}))$, is computed as:

$$\boldsymbol{p}_k(\boldsymbol{w}) = \operatorname*{argmax}_{\boldsymbol{p}\in\Delta^{n_k}}\{\langle \boldsymbol{m}_k(\boldsymbol{w}),\ \boldsymbol{p}\rangle - \Omega_k(\boldsymbol{p})\}, \tag{7}$$

Algorithm 1: Smoothed Greedy

```
    Input  :  $f(\mathcal{S}, \boldsymbol{w})$ and independent set $\mathcal{I}$
    Output: Set $\mathcal{S}$ of tours for each robot
1   $\mathcal{S} \leftarrow \emptyset$
2   for $k \leftarrow 1$ to $N$ do
        // find all addable elements in the current round
3       $U_k = \{u_1, \ldots, u_{n_k}\} \leftarrow \{T \notin \mathcal{S} \mid \mathcal{S} \cup \{T\} \in \mathcal{I}\}$
        // marginal gain for all addable elements
4       $\boldsymbol{m}_k(\boldsymbol{w}) \leftarrow (f_{\mathcal{S}}(u_1, \boldsymbol{w}), \ldots, f_{\mathcal{S}}(u_{n_k}, \boldsymbol{w}))$
        // compute a probability distribution
5       $\boldsymbol{p}_k(\boldsymbol{w}) \leftarrow \mathrm{argmax}_{\boldsymbol{p}\in\Delta^{n_k}}\{\langle \boldsymbol{m}_k(\boldsymbol{w}),\ \boldsymbol{p}\rangle - \Omega_k(\boldsymbol{p})\}$
6       $s_k \leftarrow$ sample $u \in U_k$ with probability $p_k(u, \boldsymbol{w})$
7       $\mathcal{S} \leftarrow \mathcal{S} \cup \{s_k\}$
8   end
9   return $\mathcal{S}$
```

where $\Delta^{n_k} := \{\boldsymbol{p} \in \mathbb{R}^{n_k} \mid \boldsymbol{p} \geq \boldsymbol{0}_{n_k}, \langle \boldsymbol{p}, \boldsymbol{1}_{n_k}\rangle = 1\}$ is the $(n_k - 1)$-dimensional probability simplex; $\Omega_k : \mathbb{R}^{n_k} \to \mathbb{R}$ is a strictly convex function and is a regularization function.

Next, we will show the theoretical results for Algorithm 1. Detailed explanations and proofs can be found in [24]. Let $\delta \geq 0$ be a constant that satisfies $\delta \geq \Omega_k(\boldsymbol{p}) - \Omega_k(\boldsymbol{q})$ for all $k = 1, \ldots, |\mathcal{S}|$ and $\boldsymbol{p}, \boldsymbol{q} \in \Delta^{n_k}$. We will use δ to quantify the performance of SG.

As shown in Theorem 1 in [24], in expectation, the output of SG satisfies that $\mathbb{E}[f(\mathcal{S}, \boldsymbol{w})] \geq (1 - \frac{1}{e})f(OPT, \boldsymbol{w}) - \delta n_g$, where OPT denotes the optimal solution. This result suggests that the SG algorithm in expectation almost preserves the performance of the deterministic greedy algorithm, whose approximation factor is $(1 - \frac{1}{e})$, with one extra term δn_g, which is the price for differentiability. It should be noted that by using SG, the output is stochastic and we focus on the expected result of the output. The regularization functions Ω_k are chosen to guarantee the expected outputs of SG differentiable. Examples for Ω_k will be discussed in the Sec. VI.

B. Gradient Estimation

Let $\mathcal{O}_{\mathcal{I}}$ be the set of all possible solutions returned by SG. Let $p(\mathcal{S}, \boldsymbol{w}) \in [0, 1]$ be the probability for $\mathcal{S} \in \mathcal{O}_{\mathcal{I}}$. Specifically, for a returned sequence $\mathcal{S} = \{s_1, \ldots, s_{|\mathcal{S}|}\} \in \mathcal{O}_{\mathcal{I}}$, the associated probability can be computed as $p(\mathcal{S}, \boldsymbol{w}) = \prod_{k=1}^{|\mathcal{S}|} p_k(s_k, \boldsymbol{w})$, where $p_k(s_k, \boldsymbol{w})$ is the element of $\boldsymbol{p}_k(\boldsymbol{w})$ defined Eq. (7) corresponding to $s_k \in U_k$.

Next, we will show how to construct a gradient estimator based on the output distribution. Let $Q(\mathcal{S})$ be any scalar- or vector-valued function. We want to compute $\nabla_{\boldsymbol{w}}\mathbb{E}_{\mathcal{S}\sim p(\boldsymbol{w})}[Q(\mathcal{S})] = \sum_{\mathcal{S}\in\mathcal{O}_{\mathcal{I}}} Q(\mathcal{S})\nabla_{\boldsymbol{w}} p(\mathcal{S}, \boldsymbol{w})$. Since the size of the independent set will increase exponentially w.r.t. the size of the ground set, it is computationally expensive to compute this gradient exactly. Instead, we will use the following unbiased estimator for the gradient in training.

As shown in Proposition 1 in [24], let $\mathcal{S}_j = (s_1, \ldots, s_{|\mathcal{S}_j|}) \sim p(\boldsymbol{w})(j = 1, \ldots, N)$ be outputs of SG.

Then,

$$\frac{1}{N}\sum_{j=1}^{N} Q(\mathcal{S}_j)\otimes \nabla_{\boldsymbol{w}} \ln p(\mathcal{S}_j,\boldsymbol{w}) \tag{8}$$

is an unbiased estimator of $\nabla_{\boldsymbol{w}}\mathbb{E}_{\mathcal{S}\sim p(\boldsymbol{w})}\left[Q(\mathcal{S})\right]$, where $\otimes$ denotes the outer product.

C. Differentiable Submodular Maximization for DOL

For the i-th training sample $(\boldsymbol{z}_i,\boldsymbol{w}_i)$, the associated cost w.r.t. $\boldsymbol{\theta}$ is redefined for SG as:

$$\ell_i(\hat{\boldsymbol{w}}_i,\boldsymbol{w}_i) := f(\mathcal{S}^*(\boldsymbol{w}_i),\boldsymbol{w}_i) - \mathbb{E}_{\hat{\mathcal{S}}\sim p(\hat{\boldsymbol{w}}_i)}\left[f(\hat{\mathcal{S}},\boldsymbol{w}_i)\right], \tag{9}$$

where $\hat{\boldsymbol{w}}_i = g(\boldsymbol{z}_i,\boldsymbol{\theta})$.

For a training set with batch size M, we are interested in minimizing the empirical objective function $\frac{1}{M}\sum_{i=1}^{M}\ell_i$, where $p(\hat{\boldsymbol{w}})$ is the output distribution of SG and ℓ_i is defined in Eq. (9).

We compute the gradient using the chain rule:

$$\frac{\partial \ell_i}{\partial \boldsymbol{\theta}} = \frac{\partial \ell_i}{\partial \hat{\boldsymbol{w}}_i}\cdot\frac{\partial \hat{\boldsymbol{w}}_i}{\partial \boldsymbol{\theta}} = -\frac{\partial \mathbb{E}_{\hat{\mathcal{S}}\sim p(\hat{\boldsymbol{w}}_i)}\left[f(\hat{\mathcal{S}},\boldsymbol{w}_i)\right]}{\partial \hat{\boldsymbol{w}}_i}\cdot\frac{\partial \hat{\boldsymbol{w}}_i}{\partial \boldsymbol{\theta}}, \tag{10}$$

where $\hat{\boldsymbol{w}} = g(\boldsymbol{z}_i,\boldsymbol{\theta})$.

Suppose we take N trials of SG, by setting $Q(\hat{\mathcal{S}}) = f(\hat{\mathcal{S}},\boldsymbol{w}_i)$ in Eq. (8), we have:

$$\begin{aligned}&\text{r.h.s. of (10)}\\ &= \frac{-1}{N}\sum_{j=1}^{N} f(\hat{\mathcal{S}}_j,\boldsymbol{w}_i)\otimes\nabla_{\hat{\boldsymbol{w}}_i}\ln p(\hat{\mathcal{S}}_j,\hat{\boldsymbol{w}}_i)\cdot\frac{\partial \hat{\boldsymbol{w}}_i}{\partial \boldsymbol{\theta}},\end{aligned} \tag{11}$$

where $\hat{\boldsymbol{w}}_i = g(\boldsymbol{z}_i,\boldsymbol{\theta})$.

Then, the remaining problem is how to compute $\nabla_{\hat{\boldsymbol{w}}_i}\ln p(\hat{\mathcal{S}}_j,\hat{\boldsymbol{w}}_i)$ for a given sample $\hat{\mathcal{S}}_j = (s_1,\ldots,s_{|\hat{\mathcal{S}}_j|})$ returned by SG. It should be noticed that $\nabla_{\hat{\boldsymbol{w}}_i}\ln p(\hat{\mathcal{S}}_j,\hat{\boldsymbol{w}}_i) = \nabla_{\hat{\boldsymbol{w}}_i}\ln\prod_{k=1}^{|\hat{\mathcal{S}}_j|}p(s_k,\hat{\boldsymbol{w}}_i) = \sum_{k=1}^{|\hat{\mathcal{S}}_j|}\frac{1}{p(s_k,\hat{\boldsymbol{w}}_i)}\nabla_{\hat{\boldsymbol{w}}_i}p(s_k,\hat{\boldsymbol{w}}_i)$, where $p(s_k,\hat{\boldsymbol{w}}_i)$ can be obtained when we run the SG algorithm. Therefore, we just need to compute $\nabla_{\hat{\boldsymbol{w}}_i}p(s_k,\hat{\boldsymbol{w}}_i)$. Let $\boldsymbol{p}_k(\hat{\boldsymbol{w}}_i)$ be the probability returned by Eq. (7) at the step k corresponding to the sample $\hat{\mathcal{S}}_j$. By using the chain rule,

$$\nabla_{\hat{\boldsymbol{w}}_i}\boldsymbol{p}_k(\hat{\boldsymbol{w}}_i) = \nabla_{\boldsymbol{m}_k}\boldsymbol{p}_k(\boldsymbol{m}_k)\cdot\nabla_{\hat{\boldsymbol{w}}_i}\boldsymbol{m}_k(\hat{\boldsymbol{w}}_i), \tag{12}$$

the row in $\nabla_{\hat{\boldsymbol{w}}_i}\boldsymbol{p}_k(\hat{\boldsymbol{w}}_i)$ corresponding to s_k will give $\nabla_{\hat{\boldsymbol{w}}_i}p(s_k,\hat{\boldsymbol{w}}_i)$. The first term on the r.h.s. of Eq. (12) can be computed using auto-differentiation tools [16] (the objective in (7) is strictly concave) and the second term can be computed by differentiating the parameterized submodular objective. We can then use the above result to compute the gradient of a batch and update the parameters $\boldsymbol{\theta}$ using the stochastic gradient descent method.

VI. Experiments

1) Simulation Setup:

Environment Model: There are $n_a = 10$ UAVs and $n_g = 3$ UGVs. The global wind ω is represented as $[a,b,\omega_o]$, where a and b are the shape and scale parameters of Weibull distribution, respectively, and ω_o is the wind direction.

UAV and UGV Behavior: In this case study, we consider the case that the task route is defined as a sequence of ordered locations uniformly sampled from a circle whose center is $[C_x,C_y]$ and radius is r. Using this geometric information, each route can be represented as a vector C_x,C_y,r. As for the recharging strategy, we use a simple strategy in the simulation: whenever the state of charge drops below 30%, fly to the nearest recharging location waiting for the UGVs. We use the same energy model as that in [25]. We generate UGV routes by first randomly selecting a set of nodes and then solving a Traveling Salesman Problem to get a route.

Context Input $\boldsymbol{z}$ and mapping $g_{\boldsymbol{\theta}}(\boldsymbol{z})$: As shown in Fig. 1, each $\boldsymbol{z}$ consists of two components: the task routes of all UAVs and the wind field vector $[a,b,\omega_o]$ of the working area. Since the route of the UAV can be parameterized by a circle (C_x,C_y,r), z can be represented as $[C_x^1,C_y^1,r^1,C_x^2,C_y^2,r^2,\ldots,C_x^{n_a},C_y^{n_a},r^{n_a},a,b,\omega_o]$. $g_{\boldsymbol{\theta}}(\boldsymbol{z})$ is instantiated using neural networks with one hidden layer (of size 64) with ReLu activation function.

Regularization and Basis Function We choose the entropy function for experiments. Specifically, when $\Omega_k(\boldsymbol{p}) = \epsilon\sum_{i=1}^{n_k}p(u_i)\ln p(u_i)$, where $p(u_i)$ is the $i-$th entry of $\boldsymbol{p}\in[0,1]^{n_k}$ and $\epsilon > 0$ is an arbitrary constant, δ can be set to $\epsilon\ln n_k$. For the road graph $G = (V,E)$, we use graph partition algorithms to generate nine sets of nodes, i.e., $\{W_1,\ldots,W_9\}$. For each partition $W_i\subset V$, we define three basis functions for decay parameters $\gamma\in\{0.001,0.5,1\}$.

2) Generate Training Data:

Raw Data: Based on the UAV UGV behavior models, we build a simulator for the routing problems. Given simulation parameters (e.g., UAV routes, and wind conditions), it will simulate UAVs' execution of persistent tasks. If we provide several selected routes to the simulator, it will return the number of UAVs recharged if UGV follows these routes. Based on this simulator, for each context observation $\boldsymbol{z}$, we test multiple possible route selections and obtain a set of the actual number of UAVs that UGV recharged. We will use this raw data to obtain the training data.

To train the decision-oriented framework proposed in Sec. III, we need the i-th data point to be in the form $(\boldsymbol{z}_i,\boldsymbol{w}_i)$, where $\boldsymbol{z}_i$ denotes the context input and $\boldsymbol{w}_i$ is the corresponding parameter vector of the objective function. However, $\boldsymbol{w}_i$ is not directly available and we need to do some pre-processing of the raw data. Specifically, for each $\boldsymbol{z}$, we have a set of values for different selections, i.e., $\mathcal{F} = \{f(\mathcal{S}_1),\ldots,f(\mathcal{S}_{|\mathcal{F}|})\}$. We find the corresponding $\boldsymbol{w}$ by solving the following regularized least square optimization problem [26], i.e., $\min_{\boldsymbol{w}\geq 0}\sum_{i=1}^{|\mathcal{F}|}\|f(\mathcal{S}_i,\boldsymbol{w}) - f(\mathcal{S}_i)\|_2^2 + \xi\|\boldsymbol{w}\|_2^2$, where ξ is a user-specified regularization parameter.

3) Results:

Fig. 4 shows the learning curves over epochs. In each epoch, we compute the gradient by sampling a batch size of 40 in each iteration. We can see that as the training epoch increases, the loss will gradually decrease to a steady value. It should be noticed that the loss here represents the solution quality gap between the solution obtained using ground truth parameters and the solution obtained using predicted param-

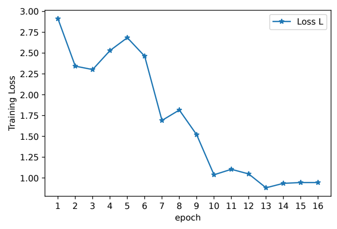


Fig. 4. Per epoch loss curve of DOL.

TABLE I

TEST RESULTS FOR LEARNED MODELS.

Avg # of UAV recharged	DOL	two-stage	random
$n_a = 6, n_g = 3$	5.3 ± 0.5	4.6 ± 0.4	2.1 ± 1.4
$n_a = 10, n_g = 3$	9.2 ± 0.7	8.5 ± 0.6	4.5 ± 1.3
$n_a = 15, n_g = 4$	14.1 ± 0.8	13.2 ± 0.7	7.5 ± 1.6

eters as defined in Problem 1. Therefore, such a decrease suggests that the decision quality is improving.

After training, we test the performance of the learned models using the simulator. We generate a set of context observations $\{\boldsymbol{z}_i\}_{\text{test}}$ and compute the corresponding predicted weights $\{\hat{\boldsymbol{w}}_i\}_{\text{test}}$. Then, we use $\{\hat{\boldsymbol{w}}_i\}_{\text{test}}$ to select routes and feed the route to the simulator to obtain the actual number of UAVs recharged. The result is shown in Table VI-.2. We compare three approaches: DOL (our), two-stage (classic supervise learning with MSE loss), and random (select routes randomly without any learning). As shown in Table VI-.2, our approach on average can result in better route selection and recharge more UAVs.

VII. Conclusion

We propose a decision-oriented learning framework for mobile charging routing problems. We first show how to formulate the learning problem in the context of mobile charging routing. Then, we show how to make submodular maximization a differentiable layer by using stochastic smoothing techniques. The proposed framework and formulation are validated through several case studies.

References

[1] A. Krause, A. Singh, and C. Guestrin, "Near-optimal sensor placements in gaussian processes: Theory, efficient algorithms and empirical studies." *Journal of Machine Learning Research*, vol. 9, no. 2, 2008.

[2] L. Zhou, V. Tzoumas, G. J. Pappas, and P. Tokekar, "Resilient active target tracking with multiple robots," *IEEE Robotics and Automation Letters*, vol. 4, no. 1, pp. 129–136, 2018.

[3] K.-S. Tseng and B. Mettler, "Near-optimal probabilistic search via submodularity and sparse regression," *Autonomous Robots*, vol. 41, no. 1, pp. 205–229, 2017.

[4] G. L. Nemhauser, L. A. Wolsey, and M. L. Fisher, "An analysis of approximations for maximizing submodular set functions—I," *Mathematical programming*, vol. 14, pp. 265–294, 1978.

[5] D. Mitchell, N. Chakraborty, K. Sycara, and N. Michael, "Multi-robot persistent coverage with stochastic task costs," in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2015, pp. 3401–3406.

[6] J. Derenick, N. Michael, and V. Kumar, "Energy-aware coverage control with docking for robot teams," in *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2011, pp. 3667–3672.

[7] L. Liu and N. Michael, "Energy-aware aerial vehicle deployment via bipartite graph matching," in *2014 International Conference on Unmanned Aircraft Systems (ICUAS)*. IEEE, 2014, pp. 189–194.

[8] K. Yu, J. M. O'Kane, and P. Tokekar, "Coverage of an environment using energy-constrained unmanned aerial vehicles," in *2019 international conference on robotics and automation (ICRA)*. IEEE, 2019, pp. 3259–3265.

[9] G. Shi, N. Karapetyan, A. B. Asghar, J.-P. Reddinger, J. Dotterweich, J. Humann, and P. Tokekar, "Risk-aware uav-ugv rendezvous with chance-constrained markov decision process," *The 61th IEEE Conference on Decision and Control (CDC)*, 2022.

[10] A. B. Asghar, G. Shi, N. Karapetyan, J. Humann, J.-P. Reddinger, J. Dotterweich, and P. Tokekar, "Risk-aware resource allocation for multiple uavs-ugvs recharging rendezvous," *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2023.

[11] A. N. Elmachtoub and P. Grigas, "Smart "predict, then optimize"," *Management Science*, vol. 68, no. 1, pp. 9–26, 2022.

[12] B. Wilder, B. Dilkina, and M. Tambe, "Melding the data-decisions pipeline: Decision-focused learning for combinatorial optimization," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, no. 01, 2019, pp. 1658–1665.

[13] J. Mandi, P. J. Stuckey, T. Guns *et al.*, "Smart predict-and-optimize for hard combinatorial optimization problems," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 02, 2020, pp. 1603–1610.

[14] N. Wilde, A. Sadeghi, and S. L. Smith, "Learning submodular objectives for team environmental monitoring," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 960–967, 2021.

[15] B. Amos and J. Z. Kolter, "Optnet: Differentiable optimization as a layer in neural networks," in *International Conference on Machine Learning*. PMLR, 2017, pp. 136–145.

[16] A. Agrawal, B. Amos, S. Barratt, S. Boyd, S. Diamond, and J. Z. Kolter, "Differentiable convex optimization layers," *Advances in neural information processing systems*, vol. 32, 2019.

[17] S. Muntwiler, K. P. Wabersich, and M. N. Zeilinger, "Learning-based moving horizon estimation through differentiable convex optimization layers," in *Learning for Dynamics and Control Conference*. PMLR, 2022, pp. 153–165.

[18] B. Amos, I. Jimenez, J. Sacks, B. Boots, and J. Z. Kolter, "Differentiable mpc for end-to-end planning and control," *Advances in neural information processing systems*, vol. 31, 2018.

[19] S. Chen, K. Saulnier, N. Atanasov, D. D. Lee, V. Kumar, G. J. Pappas, and M. Morari, "Approximating explicit model predictive control using constrained neural networks," in *2018 Annual American control conference (ACC)*. IEEE, 2018, pp. 1520–1527.

[20] M. Bhardwaj, B. Boots, and M. Mukadam, "Differentiable gaussian process motion planning," in *2020 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2020, pp. 10 598–10 604.

[21] A. Ferber, B. Wilder, B. Dilkina, and M. Tambe, "Mipaal: Mixed integer program as a layer," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 02, 2020, pp. 1504–1511.

[22] M. V. Pogančić, A. Paulus, V. Musil, G. Martius, and M. Rolinek, "Differentiation of blackbox combinatorial solvers," in *International Conference on Learning Representations*, 2019.

[23] J. Djolonga and A. Krause, "Differentiable learning of submodular models," *Advances in Neural Information Processing Systems*, vol. 30, 2017.

[24] S. Sakaue, "Differentiable greedy algorithm for monotone submodular maximization: Guarantees, gradient estimators, and applications," in *International Conference on Artificial Intelligence and Statistics*. PMLR, 2021, pp. 28–36.

[25] F. B. Sorbelli, F. Corò, S. K. Das, and C. M. Pinotti, "Energy-constrained delivery of goods with drones under varying wind conditions," *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, no. 9, pp. 6048–6060, 2020.

[26] S. Tschiatschek, R. K. Iyer, H. Wei, and J. A. Bilmes, "Learning mixtures of submodular functions for image collection summarization," *Advances in neural information processing systems*, vol. 27, 2014.