Risk-Aware Submodular Optimization for Multirobot Coordination

Lifeng Zhou, *Member, IEEE* and Pratap Tokekar, *Member, IEEE*

***Abstract*—We study the problem of incorporating risk while making combinatorial decisions under uncertainty. We formulate a discrete submodular maximization problem for selecting a set using conditional value at risk (CVaR), a risk metric commonly used in financial analysis. While the CVaR has recently been used in the optimization of linear cost functions in robotics, we take the first step toward extending this to discrete submodular optimization and provide several positive results. Specifically, we propose the sequential greedy algorithm that provides an approximation guarantee on finding the maxima of the CVaR cost function under a matroid constraint. The approximation guarantee shows that the solution produced by our algorithm is within a constant factor of the optimal and an additive term that depends on the optimal. Our analysis uses the curvature of the submodular set function and proves that the algorithm runs in polynomial time. This formulates a number of combinatorial optimization problems that appear in robotics. We use two such problems, i.e., vehicle assignment under uncertainty for mobility on demand and sensor selection with failures for environmental monitoring, as case studies to demonstrate the efficacy of our formulation. We also study the problem of adaptive risk-aware submodular maximization. We design a heuristic solution that triggers the replanning only when certain conditions are satisfied, to eliminate unnecessary planning. In particular, for the online mobility-on-demand study, we propose an adaptive triggering assignment algorithm that triggers a new assignment only when it can potentially reduce the waiting time at demand locations. We verify the performance of the proposed algorithms through simulations.**

***Index Terms*—Conditional value at risk (CVaR), risk-aware decision making, sequential greedy algorithm (SGA), submodular maximization.**

I. Introduction

COMBINATORIAL optimization problems find a variety of applications in robotics. Typical examples include the following.

1) *Sensor placement:* Where to place sensors to maximally cover the environment [1] or reduce the uncertainty in the environment [2]?
2) *Task allocation:* How to allocate tasks to robots to maximize the overall utility gained by the robots [3]?
3) *Combinatorial auction:* How to choose a combination of items for each player to maximize the total rewards [4]?

Algorithms for solving such problems find use in sensor placement for environment monitoring [1], [2], robot target assignment and tracking [5]–[10], and informative path planning [11], [12]. The underlying optimization problem in most cases can be written as

$$\max_{\mathcal{S}\in\mathcal{I},\mathcal{S}\subseteq\mathcal{X}} f(\mathcal{S}) \tag{1}$$

where $\mathcal{X}$ denotes a ground set from which a subset of elements $\mathcal{S}$ must be chosen. Typically, f is a monotone submodular utility function [13], [14]. Submodularity is the property of diminishing returns. Many information-theoretic measures, such as mutual information [2], and geometric measures, such as the visible area [15], are known to be submodular. $\mathcal{I}$ denotes a matroid constraint [13], [14]. Matroids are a powerful combinatorial tool that can represent constraints on the solution set, e.g., cardinality constraints ("place no more than k sensors") and connectivity constraints ("the communication graph of the robots must be connected") [16]. The objective of this problem is to find a set $\mathcal{S}$ satisfying a matroid constraint $\mathcal{I}$ and maximizing the utility $f(\mathcal{S})$. The general form of this problem is NP-complete. However, a greedy algorithm yields a constant-factor approximation guarantee [13], [14].

In practice, sensors can fail or get compromised [17] or robots may not know the exact positions of the targets [18]. Hence, the utility $f(\mathcal{S})$ is not necessarily deterministic but can have uncertainty. Our main contribution is to extend the traditional formulation given in (1) to also account for the uncertainty in the actual cost function. We model the uncertainty by assuming that the utility function is of the form $f(\mathcal{S}, y)$, where $\mathcal{S} \in \mathcal{X}$ is the decision variable and $y \in \mathcal{Y}$ represents a random variable that is independent of $\mathcal{S}$. We focus on the case where $f(\mathcal{S}, y)$ is monotone submodular in $\mathcal{S} \in \mathcal{X}$ and integrable in y.

The traditional way of stochastic optimization is to use the expected utility as the objective function

$$\max_{\mathcal{S}\in\mathcal{I},\mathcal{S}\in\mathcal{X}} \mathbb{E}_y[f(\mathcal{S}, y)]. \tag{2}$$

Since the sum of the monotone submodular functions is monotone submodular, $\mathbb{E}_y[f(\mathcal{S}, y)]$ is still monotone submodular in

Manuscript received 7 November 2021; accepted 17 February 2022. Date of publication 6 April 2022; date of current version 4 October 2022. This paper was recommended for publication by Associate Editor F. Amigoni and Editor P. Robuffo Giordano upon evaluation of the reviewers' comments. This work was supported in part by the National Science Foundation under Grant IIS-1637915 and in part by the Office of Naval Research under Grant N00014-18-1-2829. *(Corresponding author: Lifeng Zhou.)*

Lifeng Zhou was with the Department of Electrical and Computer Engineering, Virginia Tech, Blacksburg, VA 24061 USA. He is now with the General Robotics, Automation, Sensing and Perception Laboratory, University of Pennsylvania, Philadelphia, PA 19104 USA (e-mail: lfzhou@seas.upenn.edu).

Pratap Tokekar is with the Department of Computer Science, University of Maryland, College Park, MD 20742 USA (e-mail: tokekar@umd.edu).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TRO.2022.3158227.

Digital Object Identifier 10.1109/TRO.2022.3158227

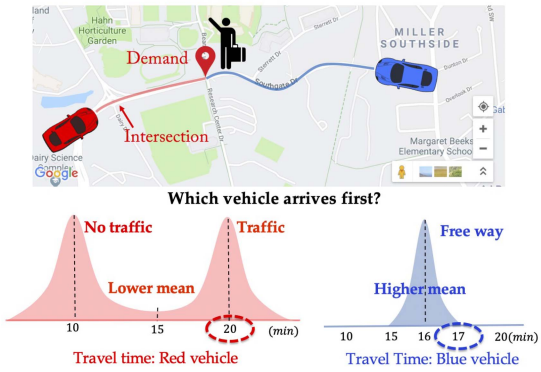


Fig. 1. Mobility on demand with travel time uncertainty of self-driving vehicles.

$\mathcal{S}$. Thus, the greedy algorithm still retains its constant-factor performance guarantee [13], [14]. Examples of this approach include influence maximization [19], moving target detection and tracking [18], and robot assignment with travel-time uncertainty [20].

While optimizing the expected utility has its uses, it also has its pitfalls. Consider the example of mobility on demand, where two vehicles, i.e., the red vehicle and the blue vehicle, are available to pick up the passengers at a demand location (see Fig. 1). The red vehicle is closer to the demand location, but it needs to cross an intersection where it may need to stop and wait. The blue vehicle is further from the demand location, but there is no intersection along the path. The travel time for the red vehicle follows a bimodal distribution (with and without traffic stop), whereas that for the blue vehicle follows a unimodal distribution with a higher mean but lower uncertainty. Clearly, if the passenger uses the expected travel time as the objective, they would choose the red vehicle. However, they will risk waiting a much longer time, i.e., 17–20 min about half of the times. A more risk-aware passenger would choose the blue vehicle, which has higher expected waiting time of 16 min but a lesser risk of waiting longer.

Thus, in these scenarios, it is natural to go beyond expectation and focus on a risk-aware measure. One popular coherent risk measure is the *conditional value at risk* (CVaR) [21], [22]. The CVaR is parameterized by a risk level α. Informally, maximizing CVaR is equivalent to maximizing the expected utility in the worst α-tail scenarios.[1]

Related work: Several works have focused on optimizing the CVaR. Rockafellar and Uryasev [22] presented an algorithm for CVaR minimization for reducing the risk in financial portfolio optimization with a large number of instruments. Note that, in portfolio optimization, we select a distribution over available decision variables, instead of selecting a single one. Later, they showed the advantage of optimizing the CVaR for general loss distributions in finance [23].

Another popular risk measure in finance is the value at risk (VaR) [24], which is generally used for formulating the chance-constrained optimization problems. Yang and Chakraborty [25] studied a chance-constrained combinatorial optimization problem that takes into account the risk in multirobot assignment. They later extended this to knapsack problems [26]. They solved the problem by transforming it to a risk-aware problem with mean–variance measure [27].

Compared to the VaR, Rockafellar and Uryasev [22] stated that the CVaR has more desirable mathematical characteristics, such as subadditivity, convexity, and coherence, and is easier to optimize when it is calculated from scenarios. In addition, Majumdar and Pavone [28] argued that the CVaR has better properties for quantifying risk than the VaR or mean–variance based on six proposed axioms in the context of robotics.

When the utility is a discrete submodular set function, i.e., $f(\mathcal{S}, y)$, Maehara [29] presented a negative result for maximizing the CVaR—there is no polynomial time multiplicative approximation algorithm for this problem under some reasonable assumptions in computational complexity. To avoid this difficulty, Ohsaka and Yoshida [30] used the same idea from portfolio optimization and proposed a method of selecting a distribution over available sets rather than selecting a single set and gave a provable guarantee. Following this line, Wilder [31] considered a CVaR maximization of a continuous submodular function instead of the submodular set functions. They gave a $(1 - 1/e)$-approximation algorithm for continuous submodular functions and also evaluated the algorithm for discrete submodular functions using portfolio optimization [30].

Contributions: In this article, we first focus on the problem of selecting a single set, similar to [29], to optimize the CVaR of a stochastic submodular set function. We propose an approximation algorithm with provable theoretical guarantees. Then, we study the problem of adaptive risk-aware submodular maximization. We design a triggering replanning strategy that solves the CVaR optimization problem only at specific time steps to avoid unnecessary planning costs.

Our contributions are as follows.

1) We propose the sequential greedy algorithm (SGA), which uses the deterministic greedy algorithm [13], [14] as a subroutine to find the maximum value of the CVaR (see Algorithm 1).
2) We prove that the solution found by the SGA is within a constant factor of the optimal performance along with an additive term, which depends on the optimal value and a sampling error. We also prove that SGA runs in polynomial time (see Theorem 1) and the performance improves as the running time increases.
3) We formulate an adaptive risk-aware submodular maximization problem for the online version (see Problem 2). We devise an event-triggered replanning strategy that replans by SGA only when certain conditions are reached. Particularly, we design the adaptive triggering assignment (ATA) for the online mobility-on-demand application (see Algorithm 2), which triggers reassignment only at some specific time steps to avoid unnecessary assignments.
4) We demonstrate the utility of the proposed CVaR maximization problem through two case studies (see Section III-B). We evaluate the performance of SGA and the ATA through simulations (see Section VI).

[1]We formally review the CVaR and other related concepts in Section II-C.

Organization of the rest of this article: We give the necessary background knowledge for the rest of this article in Section II. We formulate the CVaR submodular maximization problem with two case studies in Section III. We present SGA along with the analysis of its computational complexity and approximation ratio in Section IV. We present an adaptive risk-aware submodular maximization problem and design a triggering replanning strategy in Section V. We illustrate the performance of SGA to the two case studies and evaluate the performance of ATA with street networks in Section VI. We conclude this article in Section VII.

A preliminary version of this article was first presented in [32] without the analysis of the approximation error induced by sampling method (see Section IV), the formulation of the adaptive risk-aware submodular maximization problem and a triggering replanning strategy (see Section V), and the numerical evaluations of ATA on street networks (see Section VI-C).

II. Background and Preliminaries

We start by defining the conventions used in this article.

Calligraphic font denotes a set (e.g., $\mathcal{A}$). Given a set $\mathcal{A}$, $2^{\mathcal{A}}$ denotes its power set. $|\mathcal{A}|$ denotes the cardinality of $\mathcal{A}$. Given a set $\mathcal{B}$, $\mathcal{A} \setminus \mathcal{B}$ denotes the set of elements in $\mathcal{A}$ that are not in $\mathcal{B}$. $\Pr[\cdot]$ denotes the probability of an event and $\mathbb{E}[\cdot]$ denotes the expectation of a random variable. $\lceil x \rceil = \min\{n \in \mathbb{Z} | x \leq n\}$, where $\mathbb{Z}$ denotes the set of integers.

Next, we give the background on set functions, the greedy algorithm, and risk measures.

A. Background on Set functions: Monotonicity, Submodularity, Matroid, and Curvature

We begin by reviewing useful properties of a set function $f(\mathcal{S})$ defined for a finite ground set $\mathcal{X}$ and matroid constraints.

1) *Monotonicity [13]:* A set function $f : 2^{\mathcal{X}} \mapsto \mathbb{R}$ is monotone (nondecreasing) if and only if for any sets $\mathcal{S} \subseteq \mathcal{S}' \subseteq \mathcal{X}$, we have $f(\mathcal{S}) \leq f(\mathcal{S}')$.
2) *Normalized function [14]:* A set function $f : 2^{\mathcal{X}} \mapsto \mathbb{R}$ is called normalized if and only if $f(\emptyset) = 0$.
3) *Submodularity [13, Proposition 2.1]:* A set function $f : 2^{\mathcal{X}} \mapsto \mathbb{R}$ is submodular if and only if for any sets $\mathcal{S} \subseteq \mathcal{S}' \subseteq \mathcal{X}$, and any element $s \in \mathcal{X}$ and $s \notin \mathcal{S}'$, we have $f(\mathcal{S} \cup \{s\}) - f(\mathcal{S}) \geq f(\mathcal{S}' \cup \{s\}) - f(\mathcal{S}')$. Therefore, the marginal gain $f(\mathcal{S} \cup \{s\}) - f(\mathcal{S})$ is nonincreasing.
4) *Matroid [33, Sec. 39.1]:* Denote a nonempty collection of subsets of $\mathcal{X}$ as $\mathcal{I}$. The pair $(\mathcal{X}, \mathcal{I})$ is called a matroid if and only if the following conditions are satisfied.
   1) For any set $\mathcal{S} \subseteq \mathcal{X}$, it must hold that $\mathcal{S} \in \mathcal{I}$, and for any set $\mathcal{P} \subseteq \mathcal{S}$ it must hold that $\mathcal{P} \in \mathcal{I}$.
   2) For any sets $\mathcal{S}, \mathcal{P} \in \mathcal{I}$ and $|\mathcal{P}| \leq |\mathcal{S}|$, it must hold that there exists an element $s \in \mathcal{S} \backslash \mathcal{P}$ such that $\mathcal{P} \cup \{s\} \in \mathcal{I}$.

   We will use two specific forms of matroids that are reviewed next.
   1) *Uniform matroid:* A *uniform matroid* is a matroid $(\mathcal{X}, \mathcal{I})$ such that for a positive integer κ, $\{\mathcal{S} : \mathcal{S} \subseteq \mathcal{X}, |\mathcal{S}| \leq \kappa\}$. Thus, the uniform matroid only constrains the cardinality of the feasible sets in $\mathcal{I}$.
   2) *Partition matroid:* A *partition matroid* is a matroid $(\mathcal{X}, \mathcal{I})$ such that for a positive integer n, disjoint sets $\mathcal{X}_1, \ldots, \mathcal{X}_n$ and positive integers $\kappa_1, \ldots, \kappa_n$, $\mathcal{X} \equiv \mathcal{X}_1 \cup \cdots \mathcal{X}_n$ and $\mathcal{I} = \{\mathcal{S} : \mathcal{S} \subseteq \mathcal{X}, |\mathcal{S} \cap \mathcal{X}_i| \leq \kappa_i$ for all $i = 1, \ldots, n\}$.
5) *Curvature [34]:* Consider a matroid $\mathcal{I}$ for $\mathcal{X}$ and a nondecreasing submodular set function $f : 2^{\mathcal{X}} \mapsto \mathbb{R}$ such that (without loss of generality) for any element $s \in \mathcal{X}$, $f(s) \neq 0$. The curvature measures how far f is from submodularity or linearity. Define *curvature* of f over the matroid $\mathcal{I}$ as

$$c_f \triangleq 1 - \min_{s \in \mathcal{S}, \mathcal{S} \in \mathcal{I}} \frac{f(\mathcal{S}) - f(\mathcal{S} \setminus \{s\})}{f(s)}. \tag{3}$$

   Note that the definition of curvature c_f (3) implies that $0 \leq c_f \leq 1$. Specifically, if $c_f = 0$, it means for all the feasible sets $\mathcal{S} \in \mathcal{X}$, $f(\mathcal{S}) = \sum_{s \in \mathcal{S}} f(s)$. In this case, f is a modular function. In contrast, if $c_f = 1$, then there exist a feasible $\mathcal{S} \in \mathcal{I}$ and an element $s \in \mathcal{X}$ such that $f(\mathcal{S}) = f(\mathcal{S} \setminus \{s\})$. In this case, the element s is redundant for the contribution of the value of f given the set $\mathcal{S} \setminus \{s\}$.

B. Greedy Approximation Algorithm

In order to maximize a set function f, the greedy algorithm selects each element s of $\mathcal{X}$ based on the maximum marginal gain at each round, i.e.,

$$s = \operatorname*{argmax}_{s \in \mathcal{X}} f(\mathcal{P} \cup \{s\}) - f(\mathcal{P})$$

where $\mathcal{P} \subseteq \mathcal{S}$.

We consider maximizing a normalized monotone submodular set function f. For any matroid, the greedy algorithm gives a 1/2 approximation [14]. In particular, the greedy algorithm can give a $(1 - 1/e)$ approximation of the optimal solution under the uniform matroid [13]. If we know the curvature of the set function f, we have a $1/(1 + c_f)$ approximation for any matroid constraint [34, Th. 2.3]. That is,

$$\frac{f(\mathcal{S}^G)}{f^\star} \geq \frac{1}{1 + c_f}.$$

where $\mathcal{S}^G \in \mathcal{I}$ is the set selected by the greedy algorithm, $\mathcal{I}$ is the uniform matroid, and $f^\star$ is the function value with optimal solution. Note that, if $c_f = 0$, which means f is modular, then the greedy algorithm reaches the optimal. If $c_f = 1$, then we have the $1/2$ approximation.

C. Risk Measures

Let $f(\mathcal{S}, y)$ be a utility function with the decision set $\mathcal{S}$ and the random variable y. For each $\mathcal{S}$, the utility $f(\mathcal{S}, y)$ is also a random variable with a distribution induced by that of y. First, we define the VaR at risk level $\alpha \in (0, 1]$.

Value at risk:

$$\mathrm{VaR}_\alpha(\mathcal{S}) = \inf\{\tau \in \mathbb{R}, \Pr[f(\mathcal{S}, y) \leq \tau] \geq \alpha\}. \tag{4}$$

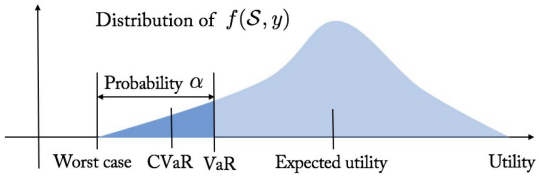


Fig. 2. Illustration of risk measures: VaR and CVaR.

Thus, $\text{VaR}_\alpha(\mathcal{S})$ denotes the left endpoint of the α-quantile(s) of the random variable $f(\mathcal{S}, y)$. The CVaR is the expectation of this set of α worst cases of $f(\mathcal{S}, y)$, defined as

Conditional value at risk:

$$\text{CVaR}_\alpha(\mathcal{S}) = \mathop{\mathbb{E}}_{y}[f(\mathcal{S}, y) | f(\mathcal{S}, y) \leq \text{VaR}_\alpha(\mathcal{S})]. \tag{5}$$

Fig. 2 shows an illustration of $\text{VaR}_\alpha(\mathcal{S})$ and $\text{CVaR}_\alpha(\mathcal{S})$. $\text{CVaR}_\alpha(\mathcal{S})$ is more popular than $\text{VaR}_\alpha(\mathcal{S})$ since it has better properties [22], such as *coherence* [35].

When optimizing $\text{CVaR}_\alpha(\mathcal{S})$, we usually resort to an auxiliary function

$$H(\mathcal{S}, \tau) = \tau - \frac{1}{\alpha}\mathbb{E}[(\tau - f(\mathcal{S}, y))_+]$$

where $(z)_+ = \max(z, 0)$. We know that optimizing $\text{CVaR}_\alpha(\mathcal{S})$ over $\mathcal{S}$ is equivalent to optimizing the auxiliary function $H(\mathcal{S}, \tau)$ over $\mathcal{S}$ and τ [22]. The following lemmas give useful properties of the auxiliary function $H(\mathcal{S}, \tau)$.

Lemma 1: If $f(\mathcal{S}, y)$ is normalized, monotone increasing, and submodular in set $\mathcal{S}$ for any realization of y, the auxiliary function $H(\mathcal{S}, \tau)$ is monotone increasing and submodular, but not necessarily normalized in set $\mathcal{S}$ for any given τ.

We provide the proofs for all the lemmas and the theorem in the Appendix.

Lemma 2: The auxiliary function $H(\mathcal{S}, \tau)$ is concave in τ for any given set $\mathcal{S}$.

Lemma 3: For any given set $\mathcal{S}$, the gradient of the auxiliary function $H(\mathcal{S}, \tau)$ with respect to τ fulfills $-(\frac{1}{\alpha} - 1) \leq \frac{\partial H(\mathcal{S}, \tau)}{\partial \tau} \leq 1$.

III. Problem Formulation and Case Studies

We first formulate the CVaR submodular maximization problem and then present two applications that we use as case studies.

A. Problem Formulation

We consider the problem of maximizing $\text{CVaR}_\alpha(\mathcal{S})$ over a decision set $\mathcal{S} \subseteq \mathcal{X}$ under a matroid constraint $\mathcal{S} \in \mathcal{I}$. We know that maximizing $\text{CVaR}_\alpha(\mathcal{S})$ over $\mathcal{S}$ is equivalent to maximizing the auxiliary function $H(\mathcal{S}, \tau)$ over $\mathcal{S}$ and τ [22]. Thus, we propose the maximization problem as follows.

Problem 1 (CVaR submodular maximization):

$$\begin{aligned} &max\, \tau - \frac{1}{\alpha}\mathbb{E}[(\tau - f(\mathcal{S}, y))_+] \\ &\text{s.t. } \mathcal{S} \in \mathcal{I}, \tau \in [0, +\infty). \end{aligned} \tag{6}$$

Problem 1 gives a risk-aware version of maximizing submodular set functions.

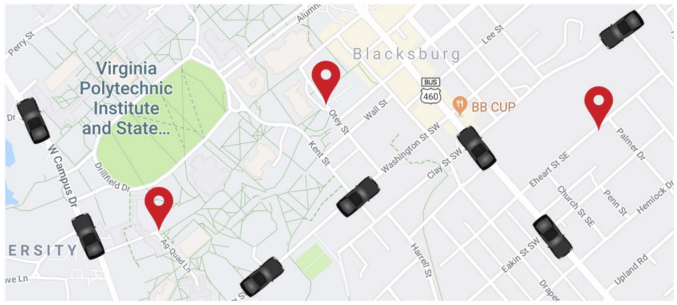


Fig. 3. Mobility on demand with multiple demands and multiple self-driving vehicles.

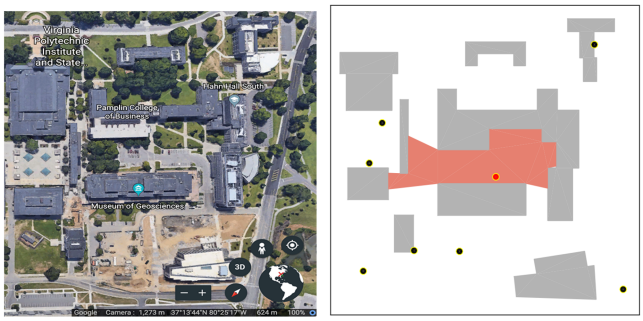
B. Case Studies

The risk-aware submodular maximization (see Problem 1) has many applications in robotics. In the following, we describe two specific applications, which we will use in the simulations.

1) Resilient Mobility on Demand: Consider a mobility-on-demand problem, where we assign R vehicles to N demand locations under arrival time uncertainty. An example is shown in Fig. 3, where seven self-driving vehicles must be assigned to three demand locations to pick up passengers. We follow the same constraint setting in [20]—each vehicle can be assigned to at most one demand, but multiple vehicles can be assigned to the same demand. Only the vehicle that arrives first is chosen for picking up the passengers. Note that the advantage of the redundant assignment to each demand is that it counters the effect of uncertainty and reduces the waiting time at demand locations [20]. This may be too conservative for consumer mobility-on-demand services but can be crucial for urgent and time-critical tasks such as delivering medical supplies [36].

Assume that the arrival time for the vehicle to arrive at a demand location is a random variable. Its randomness can depend on the mean arrival time. For example, it is possible to have a shorter path that passes through many intersections, which leads to uncertainty on arrival time, while a longer road (possibly a highway) has a lower arrival time uncertainty. Note that for each demand location, there is a set of vehicles assigned to it. The vehicle selected at the demand location is the one that arrives first. Then, this problem becomes a minimization one since we would like to minimize the arrival time at all demand locations. We convert it into a maximization one by taking the reciprocal of the arrival time. Specifically, we use the *arrival utility*, which is the reciprocal of arrival time. Instead of selecting the vehicle at the demand location with minimum arrival time, we select the vehicle with maximum arrival utility. The arrival utility is also a random variable, and its randomness depends on the mean arrival utility. Denote the arrival utility for vehicle $j \in \{1, \ldots, R\}$ arriving at demand location $i \in \{1, \ldots, N\}$ as e_{ij}. Denote the subset of robots assigned to demand location i as $\mathcal{S}_i$. Then, the arrival utility for a demand location i is computed as $\max_{j \in \mathcal{S}_i} e_{ij}$. Furthermore, we denote the assignment utility as the sum of arrival utilities at all locations, i.e.,

$$f(\mathcal{S}, y) = \sum_{i \in N} \max\nolimits_{j \in \mathcal{S}_i} e_{ij} \tag{7}$$



(a) (b)

Fig. 4. Campus monitoring by using a set of sensors with visibility regions. (a) Part of Virginia Tech campus from Google Earth. (b) Top view of part of a campus and ground sensor's visibility region.

with $\bigcup_{i=1}^{N} \mathcal{S}_i = \mathcal{S}$ and $\mathcal{S}_i \cap \mathcal{S}_k = \emptyset, i, k \in \{1, \ldots, N\}$. $\mathcal{S}_i \cap \mathcal{S}_k = \emptyset$ indicates that the selected set $\mathcal{S}$ satisfies a partition matroid constraint, $\mathcal{S} \in \mathcal{I}$, which represents that each vehicle can be assigned to at most one demand. The assignment utility $f(\mathcal{S}, y)$ is monotone submodular in $\mathcal{S}$ due to the "max" function. $f(\mathcal{S}, y)$ is normalized since $f(\emptyset, y) = 0$. Here, we regard uncertainty as a risk. Our risk-aware assignment problem is a tradeoff between assignment utility and uncertainty. Our goal is to maximize the total utilities at the demand locations while considering the risk from uncertainty.

2) Robust Environment Monitoring: Consider an environment monitoring problem, where we monitor part of a campus with a group of ground sensors (see Fig. 4). Given a set of N candidate positions $\mathcal{X}$, we would like to choose a subset of M positions $\mathcal{S} \subseteq \mathcal{X}, M \leq N$, to place visibility-based sensors to maximally cover the environment. The visibility regions of the ground sensors are obstructed by the buildings in the environment [see Fig. 4(b)]. Consider a scenario where the probability of failure of a sensor depends on the area it can cover. That is, a sensor covering a larger area has a larger risk of failure associated with it. This may be due to the fact that the same number of pixels is used to cover a larger area, and therefore, each pixel covers proportionally a smaller footprint. As a result, the sensor risks missing out on detecting small objects.

Denote the probability of success and the visibility region for each sensor $i, i \in \{1, \ldots, M\}$ as p_i and v_i, respectively. Thus, the polygon each sensor i monitors is also a random variable. Denote this random polygon as A_i and denote the selection utility as the joint coverage area of a set of sensors, $\mathcal{S}$, i.e.,

$$f(\mathcal{S}, y) = \text{area}\left(\bigcup_{i=1:M} A_i\right), \quad i \in \mathcal{S}, \mathcal{S} \subseteq \mathcal{I}. \tag{8}$$

The selection utility $f(\mathcal{S}, y)$ is monotone submodular in $\mathcal{S}$ due to the overlapping area. $f(\mathcal{S}, y)$ is normalized since $f(\emptyset, y) = 0$. Here, we regard the sensor failure as a risk. Our robust environment monitoring problem is a tradeoff between area coverage and sensor failure. Our goal is to maximize the joint area covered while considering the risk from sensor failure.

IV. Algorithm and Analysis

We present SGA for solving Problem 1 by leveraging the useful properties of the auxiliary function $H(\mathcal{S}, \tau)$. The pseudocode is given in Algorithm 1. The SGA mainly consists of searching for the appropriate value of τ by solving a subproblem for a fixed τ under a matroid constraint. Even for a fixed τ, the subproblem of optimizing the auxiliary function is NP-complete. Nevertheless, we can employ the greedy algorithm for the subproblem and sequentially apply it for searching over all τ. We explain each stage in detail next.

A. Sequential Greedy Algorithm

Following are the four stages in SGA.

1) Initialization (see Algorithm 1, line 1): Algorithm 1 defines a storage set $\mathcal{M}$ and initializes it to be the empty set. Note that, for each specific τ, we can use the greedy algorithm to obtain a near-optimal solution $\mathcal{S}^G$ based on the monotonicity and submodularity of the auxiliary function $H(\mathcal{S}, \tau)$. $\mathcal{M}$ stores all the $(\mathcal{S}^G, \tau)$ pairs when searching all the possible values of τ.

*2) Searching for τ (**for** loop in Algorithm 1, lines 2–10):* We use a user-defined separation Δ (see Algorithm 1, line 3) to sequentially search for all possible values of τ within $[0, \Gamma]$. We use Γ as a searching upper bound on τ. That is because, when τ is greater than the maximal value of $f(\mathcal{S}, y))$, the auxiliary function $H(\mathcal{S}, \tau) = (1 - \frac{1}{\alpha})\tau + \frac{1}{\alpha} f(\mathcal{S}, y)$, which is nonincreasing in τ given the risk level $\alpha \in (0, 1]$. Thus, in order to maximize $H(\mathcal{S}, \tau)$, we do not need to search for the cases when τ is larger than the maximum value of $f(\mathcal{S}, y)$. Therefore, we set the value of Γ as the maximum value of $f(\mathcal{S}, y)$. Even though $f(\mathcal{S}, y)$ is a random variable, its maximum value can be set by the user based on the specific problems at hand. For example, in the environment monitoring scenario (see Section III-B), Γ can be computed as the total area of the environment because this is the largest area that robots can jointly cover. In the mobility-on-demand scenario (see Section III-B), Γ can be the highest total utilities for reaching all demand locations (e.g., when there is no traffic congestion and vehicles travel with maximum legal speeds). We show how to find Γ for these specific cases in Section VI.

3) Greedy Algorithm (see Algorithm 1, lines 4–8): For a specific τ, say τ_i, we use the greedy approach to choose set $\mathcal{S}_i^G$. We first initialize set $\mathcal{S}_i^G$ to be the empty set (see Algorithm 1, line 4). Under a matroid constraint, $\mathcal{S}_i^G \in \mathcal{I}$ (see Algorithm 1, line 5), we add a new element s which gives the maximum marginal gain of $\hat{H}(\mathcal{S}_i^G, \tau_i)$ (see Algorithm 1, line 6) into set $\mathcal{S}_i^G$ (see Algorithm 1, line 7) in each round.

4) Find the Best Pair (see Algorithm 1, line 11): Based on the collection of pairs $\mathcal{M}$ (see Algorithm 1, line 9), we pick the pair $(\mathcal{S}_i^G, \tau_i) \in \mathcal{M}$ that maximizes $\hat{H}(\mathcal{S}_i^G, \tau_i)$ as the final solution S_i^G. We denote this value of τ by τ^G. Notably, $\tau^G \in (0, \Gamma]$.

Designing an oracle: Note that an oracle $\mathcal{O}$ is used to calculate the value of $H(\mathcal{S}, \tau)$. We can use a sampling-based method, as an oracle, to approximate $H(\mathcal{S}, \tau)$. Specifically, we sample n_s realizations $\tilde{y}(s)$ from the distribution of y and approximate $H(\mathcal{S}, \tau)$ as $\hat{H}(\mathcal{S}, \tau) = \tau - \frac{1}{n_s \alpha} \sum_{\tilde{y}} [(\tau - f(\mathcal{S}, \tilde{y}))_+]$. According to [30, Lemmas 4.1–4.5], if the number of samples is

Algorithm 1: Sequential Greedy Algorithm.

Input: • Ground set $\mathcal{X}$ and matroid $\mathcal{I}$
• User-defined risk level $\alpha \in (0,1]$
• Range of the parameter $\tau \in [0,\Gamma]$ and discretization stage $\Delta \in (0,\Gamma]$
• An oracle $\mathcal{O}$ that approximates $H(\mathcal{S},\tau)$ as $\hat{H}(\mathcal{S},\tau)$
Output: • Selected set $\mathcal{S}^G$ and corresponding parameter τ^G

$\mathcal{M} \leftarrow \emptyset$
for $i = \{0,1,\ldots,\lceil \frac{\Gamma}{\Delta} \rceil\}$ **do**
  $\tau_i = i\Delta$
  $\mathcal{S}_i^G \leftarrow \emptyset$
  while $\mathcal{S}_i^G \in \mathcal{I}$ **do**
    $s = \underset{s \in \mathcal{X}\setminus\mathcal{S}_i^G, \mathcal{S}_i^G \cup \{s\} \in \mathcal{I}}{\operatorname{argmax}} \hat{H}((\mathcal{S}_i^G \cup \{s\}),\tau_i) - \hat{H}(\mathcal{S}_i^G,\tau_i)$
    $\mathcal{S}_i^G \leftarrow \mathcal{S}_i^G \cup \{s\}$
  end while
  $\mathcal{M} = \mathcal{M} \cup \{(\mathcal{S}_i^G,\tau_i)\}$
end for
$(\mathcal{S}^G,\tau^G) = \underset{(\mathcal{S}_i^G,\tau_i)\in\mathcal{M}}{\operatorname{argmax}} \hat{H}(\mathcal{S}_i^G,\tau_i)$

$n_s = O(\frac{\Gamma^2}{\epsilon^2}\log\frac{1}{\delta}), \delta, \epsilon \in (0,1)$, the CVaR approximation error is less than ϵ with the probability at least $1-\delta$. We provide a brief proof for the number of samples in the Appendix.

Observability of y: SGA uses an oracle $\mathcal{O}$ to sample from y for the approximation of $H(\mathcal{S},\tau)$. Notably, y is partially known and its realizations (samples) can be taken from some historical data. For example, in the vehicle assignment problem (see Section III-B), the samples of vehicles' travel times can be drawn from some historical traffic dataset. Even though the travel time is not exactly known beforehand, when the vehicle traverses a street (or path), its travel time can be observed and updated, as shown in the online version of the mobility on demand (see Section V).

B. Performance Analysis of SGA

Let $\mathcal{S}^G \in \mathcal{I}$ and $\tau^G \in (0,\Gamma]$ be the set and the scalar chosen by SGA, and let $\mathcal{S}^\star$ and $\tau^\star$ be the set and the scalar chosen by the optimal solution. Our analysis uses the curvature $c_f \in [0,1]$ of function $H(\mathcal{S},\tau)$ that is submodular in set $\mathcal{S}$. Then, we get our main result.

Theorem 1: Algorithm 1 yields a bounded approximation for Problem 1. Specifically

$$H(\mathcal{S}^G,\tau^G) \geq \frac{1}{1+c_f}(H(\mathcal{S}^\star,\tau^\star)-\Delta) - \frac{c_f}{1+c_f}\tau^G\left(\frac{1}{\alpha}-1\right) - \epsilon \tag{9}$$

with probability at least $1-\delta$, when the number of samples $n_s = O(\frac{\Gamma^2}{\epsilon^2}\log\frac{1}{\delta}), \delta, \epsilon \in (0,1)$. The computational time is $O(\lceil\frac{\Gamma}{\Delta}\rceil|\mathcal{X}|^2 n_s)$, where Γ and Δ are the searching upper bound on τ and searching separation parameter, and $|\mathcal{X}|$ is the cardinality of the ground set $\mathcal{X}$.

The SGA gives $1/(1+c_f)$ approximation of the optimal with three additional approximation errors. One approximation error comes from the searching separation Δ. We can make this error arbitrarily small by setting Δ to be close to zero with the cost of increasing the number of search operations $\lceil\frac{\Gamma}{\Delta}\rceil$. The second one is the sampling error ϵ from the oracle. Evidently, ϵ can be reduced by increasing the number of samples n_s. The third one comes from the additive term

$$H^{\text{add}} = \frac{c_f}{1+c_f}\tau^G\left(\frac{1}{\alpha}-1\right) \tag{10}$$

which depends on the curvature c_f and the risk level α. When the risk level α is very small, this error is very large, which means that SGA may not give a good performance guarantee of the optimal. However, if the function $H(\mathcal{S},\tau)$ is close to modular in $\mathcal{S}$ ($c_f \to 0$), this error is close to zero. Notably, when $c_f \to 0$ and $\Delta,\epsilon \to 0$, SGA gives a near-optimal solution, i.e., $H(\mathcal{S}^G,\tau^G) \to H(\mathcal{S}^\star,\tau^\star)$.

Next, we briefly describe the proof of Theorem 1. We start with the proof of approximation ratio and then go to the analysis of the computational time. We need to prove on some structural lemmas beforehand. Let $\mathcal{S}_i^\star$ be the optimal set for a specific τ_i that maximizes $H(\mathcal{S},\tau=\tau_i)$, and let $H(\mathcal{S}^\star,\tau^\star)$ be the maximum value of $H(\mathcal{S},\tau)$. We sequentially search for $\tau \in [0,\Gamma]$ with a searching separation Δ. We first describe the relationship between the maximum $H(\mathcal{S},\tau)$ founded by searching τ with a searching separation Δ and $H(\mathcal{S}^\star,\tau^\star)$.

Lemma 4:

$$\max_{i\in\{0,1,\ldots,\lceil\frac{\Gamma}{\Delta}\rceil\}} H(\mathcal{S}_i^\star,\tau_i) \geq H(\mathcal{S}^\star,\tau^\star)-\Delta. \tag{11}$$

Next, we describe the relationship between $H(\mathcal{S}_i^G,\tau)$ with set $\mathcal{S}_i^G$ selected by our greedy approach and $H(\mathcal{S}_i^\star,\tau)$ for each τ_i.

Lemma 5:

$$H(\mathcal{S}_i^G,\tau_i) \geq \frac{1}{1+c_f}H(\mathcal{S}_i^\star,\tau_i) - \frac{c_f}{1+c_f}\tau_i\left(\frac{1}{\alpha}-1\right) \tag{12}$$

where c_f is the curvature of the function $H(\mathcal{S},\tau)$ in $\mathcal{S}$ with a matroid constraint $\mathcal{I}$.

Finally, using Lemmas 4 and 5 and considering the sampling error, we can find relationship between $H(\mathcal{S}^G,\tau^G)$ (our greedy value) and $H(\mathcal{S}^\star,\tau^\star)$ (the optimal value). We summarize this relationship in Theorem 1 with a detailed proof in the Appendix.

V. Adaptive Risk-Aware Submodular Maximization

In this section, we study an adaptive version of the risk-aware submdoular maximization described in Problem 1. This is motivated by the real-time (or online) applications, where either the external environment or the internal robot state is dynamic and changing. For example, the stochastic travel times of the vehicles are changing due to real-time traffic conditions in the mobility on demand, and the sensors can fail at some point for environment monitoring. In these dynamic scenarios, if the planning is only scheduled at the initial time step and kept fixed (i.e., *one-step* planning), the robots (e.g., vehicles or sensors) may not be able

to fully utilize the real-time information for achieving a better team performance.

Instead, the robots may need to replan to adapt to real-time situations. For instance, in the mobility-on-demand scenario, as the vehicles move, they can use real-time traffic conditions to update the assignment for achieving shorter arrival time (i.e., higher utility) at demand locations as the actual travel time information is revealed.

One intuitive online strategy is to replan at all time steps. Clearly, the *all-step* planning can achieve a better performance than the *one-step* planning, since it fully utilizes the real-time information at all time steps. However, planning at all time steps can be costly (e.g., in terms of the computing resources or energy). Crucially, planning at all time steps may not be necessary.

Thus, instead of either *one-step* planning or *all-step* planning, we seek replanning strategies that can make risk-aware decisions with as fewer replanning efforts. Formally, our bicriteria optimization problem is stated as follows.

Problem 2 (Adaptive risk-aware submodular maximization): Given a risk level $\alpha \in (0, 1]$, find a risk-aware sequential replanning strategy that decides when to replan $\{k^{\mathrm{rp}}\}$ and how to replan $\{\mathcal{S}(k^{\mathrm{rp}})\}$ to

$$\begin{aligned} &max(f(\{\mathcal{S}(k^{\mathrm{rp}})\}), -|\{k^{\mathrm{rp}}\}|) \\ &\text{s.t. } \mathcal{S}(k^{\mathrm{rp}}) \in \mathcal{I} \\ &|\{k^{\mathrm{rp}}\}| \geq 1 \\ &k^{\mathrm{rp}} \in [1, T) \end{aligned} \tag{13}$$

where $[1, T)$ denotes the time span of the planning process, k^{rp} denotes a replanning time step and $\mathcal{S}(k^{\mathrm{rp}})$ denotes the replanning strategy (e.g., replacement or reassignment) at step k^{rp}, $\{\mathcal{S}(k^{\mathrm{rp}})\}$ denotes a sequence of replanning strategies corresponding to a sequence of replanning time steps $\{k^{\mathrm{rp}}\}$ during $[1, T)$, $f(\{\mathcal{S}(k^{\mathrm{rp}})\})$ denotes the utility obtained by executing the sequence of replanning strategies $\{\mathcal{S}(k^{\mathrm{rp}})\}$, $|\{k^{\mathrm{rp}}\}|$ denotes the number of times for replanning, and $|\{k^{\mathrm{rp}}\}| \geq 1$ indicates there must exist an initial planning to start the process at time step 1.

In Problem 2, we aim to simultaneously maximize the process utility $f(\{\mathcal{S}(k^{\mathrm{rp}})\})$ and minimize the number of replanning times $|\{k^{\mathrm{rp}}\}|$. We seek a sequential replanning strategy $(\{k^{\mathrm{rp}}\}^\star, \{\mathcal{S}(k^{\mathrm{rp}})\}^\star)$ that is nondominated or Pareto optimal for Problem 2. Here, the Pareto optimality depicts a situation where no solution other than $(\{k^{\mathrm{rp}}\}^\star, \{\mathcal{S}(k^{\mathrm{rp}})\}^\star)$ could improve the process utility or reduce the number of replanning times without reducing the former or raising the latter, or there is no scope for either improving the process utility or reducing the number of replanning times.

Particularly, the process utility function $f(\{\mathcal{S}(k^{\mathrm{rp}})\})$ captures the performance of the sequence of the replanning strategies $\{\mathcal{S}(k^{\mathrm{rp}})\}$ during $[1, T)$. For example, in the mobility on demand, it can be the total arrival utilities of all demand locations when these demand locations are reached by vehicles. Similarly, for environmental monitoring, it can be the joint area covered at the end of the execution horizon. The process utility is *deterministic* since it measures the performance when the process is done (i.e., at the final step T). Essentially, the optimal process utility would be for a clairvoyant strategy that will know exactly how the scenario will unfold *a priori*. However, to compute the replanning strategy $\mathcal{S}(k)$ at some intermediate step $k \in [1, T)$, the robots need to optimize a stochastic utility $f(\mathcal{S}(k), y)$, as the robots can only use the information (e.g., stochastic travel time or sensor working status) revealed so far and, thus, have uncertain knowledge of the future.

The optimal solution to Problem 2 can be found if we assume that the robots know all the future information exactly, e.g., what the travel times will be or which sensors will fail. However, this is not the realistic scenario where the robots have no knowledge of what will exactly happen in the future. Thus, finding the optimal replanning strategy for Problem 2 is extremely difficult or even impossible. Notably, even with *all-step* planning, there is no guarantee on obtaining the optimal value of the process utility. That is because, using the information unveiled so far, the planning is myopic and could mislead the planning in the future steps. In addition, minimizing the number of replanning times simultaneously makes Problem 2 even more challenging.

Instead, we relax the hardness of Problem 2 by fixing the strategy to decide *how to replan* $\mathcal{S}(k^{\mathrm{rp}})$ at every replanning step k^{rp} and build on it to present a heuristic solution that is twofold. Specifically, we leverage SGA (see Algorithm 1) to compute $\mathcal{S}(k^{\mathrm{rp}})$. That is because, at each replanning step $k^{\mathrm{rp}} \in [1, T)$, we aim to design a risk-aware strategy to optimize the stochastic utility $f(\mathcal{S}(k^{\mathrm{rp}}), y)$, with y representing the uncertainty of the environment in the future steps (i.e., from the current step k^{rp} to the end step T). That is,

$$\begin{aligned} &\max \tau - \frac{1}{\alpha}\mathbb{E}[(\tau - f(\mathcal{S}(k^{\mathrm{rp}}), y))_+] \\ &\text{s.t. } \mathcal{S}(k^{\mathrm{rp}}) \in \mathcal{I}, \tau \in [0, +\infty) \end{aligned}$$

which is exactly an instance of Problem 1 and SGA gives a bounded approximation (see Theorem 1).

Then, the question left is to decide *when to replan* to minimize $|\{k^{\mathrm{rp}}\}|$. To this end, we exploit an event-driven mechanism, where the planning is updated or recomputed only when certain conditions are met. For environmental monitoring, there is an obvious solution that the (well-working) sensors update the replacement by SGA only when some sensor fails. In this way, they can maintain a good monitoring performance without replacing themselves at all time steps. Such a strategy is not obvious for the mobility-on-demand case. Therefore, we explicitly describe the online version of the mobility on demand and design a triggering reassignment strategy to determine when to replan.

A. Online Mobility on Demand and Triggering Vehicle Reassignment

We study an online version of the resilient mobility on demand described in Section III-B. Similarly, we assign R vehicles to serve N demand locations. The assignment process starts at time step 1 when vehicles start traveling toward their assigned demand locations by the initial vehicle–demand assignment. The process ends at time step T when all the demand locations are reached by the vehicles. We denote the stochastic

Algorithm 2: Adaptive Triggering Assignment.

1: $\mathcal{S} = \text{SGA}(\{e_{ij}(1), \alpha)$
2: **while** not all N demand locations are reached **do**
3: Each vehicle j travels toward assigned demand i
4: Each vehicle j updates $t_{ij}(k)$, $e_{ij}(k)$
5: Check for all demand locations $d_i, i \in \{1, \ldots, N\}$
6: **if** $\exists \bar{t}_{ij}(k) \leq \gamma \bar{t}_{ij'}(k)$ and $\texttt{var}(t_{ij}(k)) \leq \texttt{var}(t_{ij'})(k), j, j' \in \mathcal{S}_i(k)$ **then**
7: $\mathcal{S} = \text{SGA}(\{e_{ij}(k)\}), \alpha)$
8: **end if**
9: **end while**

arrival time for each vehicle j to each demand location i at step $k, k \in [1, T)$ as $t_{ij}(k)$ with the mean $\bar{t}_{ij}(k)$ and variance $\texttt{var}(t_{ij})(k)$. Then, the corresponding arrival utility at step k is computed as $e_{ij}(k) = 1/t_{ij}(k)$. The arrival time $t_{ij}(k)$ and arrival utility $e_{ij}(k)$ can be updated based on the real-time vehicles' positions and traffic conditions (e.g., real-time waiting times at intersections). Similarly, for each demand location, we assign a set of vehicles to it and only the vehicle that arrives first at the demand location is chosen. The objective is to develop a replanning strategy that maximizes the total arrival utilities (or equivalently, minimizes the total arrival times) of all demand locations when all these demand locations are reached by vehicles while having the minimum number of times for replanning, as captured by Problem 2.

Toward this end, we design an ATA strategy in Algorithm 2. Specifically, for deciding *how to reassign* at each reassignment step $k^{\texttt{rp}}$, we use SGA to update the vehicle–demand assignment by using the vehicles' real-time arrival utilities $e_{ij}(k)$. This is based on the fact that, at each intermediate step $k \in [1, T)$, the problem turns out to be maximizing the CVaR of the stochastic utility function

$$f(\mathcal{S}(k), y) = \sum_{i \in N} \max_{j \in \mathcal{S}_i(k)} e_{ij}(k)$$

with $\mathcal{S}(k)$ and $\mathcal{S}_i(k)$ denoting the vehicle–demand assignment and the set of vehicles assigned to demand location i at step k. We know optimizing the CVaR of $f(\mathcal{S}(k), y)$ is an instance of Problem 1 and SGA yields a solution with a bounded approximation (see Theorem 1). Before that, we need to decide *when to reassign* (i.e., the reassignment step $k^{\texttt{rp}}$), which is tackled by ATA (see Algorithm 2), where an event-driven mechanism is adopted to update the assignment only when certain conditions are met.

A general version of ATA is described in Algorithm 2. Find a specific implementation of ATA for the mobility on demand in street networks in the Appendix (see Algorithm 3).[2] ATA mainly contains two assignment steps—initial assignment (see Algorithm 2, line 1) and triggering assignment (see Algorithm 2, line 7). Once the initial assignment (see Algorithm 2, line 1) is done, each vehicle i travels toward its (first) assigned demand location j (see Algorithm 2, line 3) and updates its arrival time $t_{ij}(k)$ and corresponding arrival utility $e_{ij}(k)$ at each time step k (see Algorithm 2, line 4).

The reassignment is triggered by checking the arrival times of vehicles assigned to each demand location d_i (see Algorithm 2, line 5). Specifically, among the set of vehicles $\mathcal{S}_i(k)$ assigned to demand location i, if the arrival time of one vehicle $j \in \mathcal{S}_i(k)$ is less than that of one other vehicle $j' \in \mathcal{S}_i(k)$ on average and has smaller uncertainty (see Algorithm 2, line 6), a new assignment is triggered (see Algorithm 2, line 7), because, in this case, there is no need to continue assigning vehicle j' to demand location i. Instead, it would be helpful to trigger a new assignment to assign vehicle j' to other demand locations. Here, $\gamma \in (0, 1)$ is the triggering ratio. It regulates the triggering frequency, e.g., a larger γ triggers more assignments. The ATA terminates when all demand locations are reached by vehicles (see Algorithm 2, line 2).

It is intuitive that ATA can (probably) achieve a shorter arrival time than one-step assignment since the real-time information (e.g., vehicles' positions and traffic conditions) can be utilized for updating assignments. One simple example could be that if a demand location is reached by some vehicles, the other vehicles assigned to it and yet have not arrived should be reassigned to other demand locations. However, scheduling reassignment at every step, as in *all-step* reassignment, may not be necessary or even detrimental in some cases. That is because the reassignment based on the information revealed so far (before the final step) is myopic and could be misleading for the future assignments. Clearly, *all-step* reassignment with the most number of reassignments increases the chance of misleading. In the next section, we illustrate the comparisons of these three assignment algorithms and the effect of the triggering ratio γ.

VI. Simulations

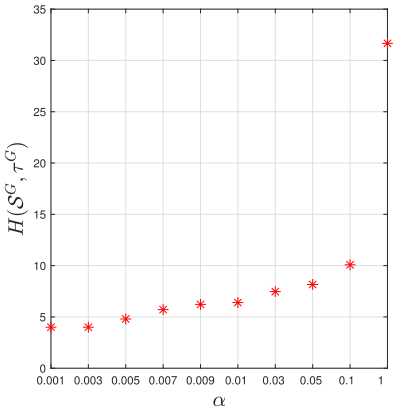
We perform numerical simulations to verify the performance of SGA in risk-aware mobility-on-demand and robust environment monitoring and the performance of ATA in online risk-aware mobility on demand. Our code is available online.[3]

A. Resilient Mobility on Demand Under Arrival Time Uncertainty

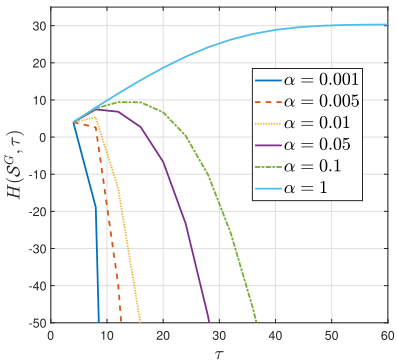
We consider assigning $R = 6$ supply vehicles to $N = 4$ demand locations in a 2-D environment. The positions of the demand locations and the supply vehicles are randomly generated within a square environment of ten-unit side length. Denote the Euclidean distance between demand location $i \in \{1, \ldots, N\}$ and vehicle position $j \in \{1, \ldots, R\}$ as d_{ij}. Based on the distribution discussion of the arrival utility distribution in Section III-B, we assume that each arrival utility e_{ij} has a uniform distribution with its mean proportional to the reciprocal of the distance between demand i and vehicle j. Furthermore, the uncertainty is higher if the mean utility is higher. Note that the algorithm can handle other, more complex, distributions for arrival times. We use a uniform distribution for ease of exposition. Specifically, denote the mean of e_{ij} as $\bar{e}_{ij}$ and set $\bar{e}_{ij} = 10/d_{ij}$. We model the

[2]For clarity, we drop the time indices k of notations for online mobility on demand in Algorithm 3 and the corresponding simulations in Section VI-C.

[3][Online]. Available: https://github.com/raaslab/risk_averse_submodular_selection



(a)



(b)

Fig. 5. Value of $H(\mathcal{S}, \tau)$ by SGA with respect to different risk levels. (a) Value of $H(\mathcal{S}^G, \tau^G)$ with respect to several risk levels α. (b) Function $H(\mathcal{S}^G, \tau)$ with respect to several risk levels α.

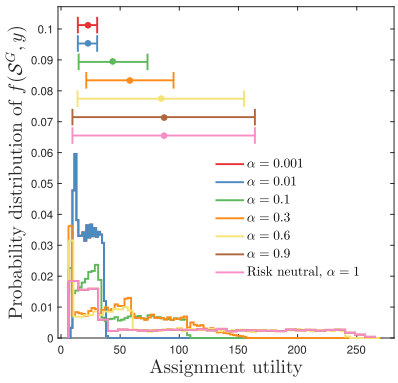


Fig. 6. Distribution of the assignment utility $f(\mathcal{S}^G, y)$ by SGA.

arrival utility distribution to be a uniform distribution as follows:

$$e_{ij} = [\bar{e}_{ij} - \bar{e}_{ij}^{2.5}/\max\{\bar{e}_{ij}\}, \bar{e}_{ij} + \bar{e}_{ij}^{2.5}/\max\{\bar{e}_{ij}\}]$$

where $\max\{\bar{e}_{ij}\} = \max_{i,j} e_{ij}, i \in \{1, \ldots, N\}, j \in \{1, \ldots, R\}$.

From the assignment utility function (7), for any realization of y, say $\tilde{y}$, we have

$$f(\mathcal{S}, \tilde{y}) := \sum_{i \in N} \max_{j \in \mathcal{S}_i} \tilde{e}_{ij}$$

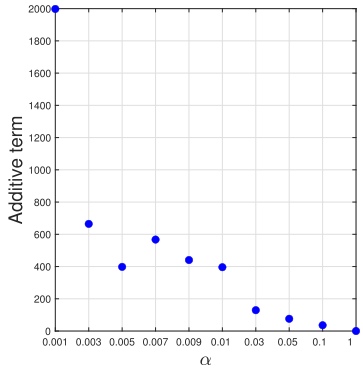


Fig. 7. Additive term (10) in the approximation ratio with respect to risk level α.

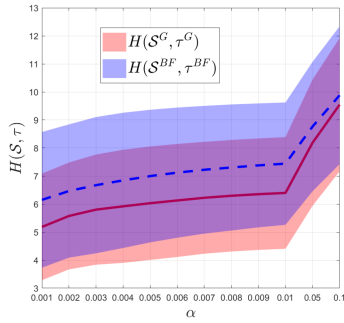


Fig. 8. Comparison of the values of $H(\mathcal{S}, \tau)$ obtained by SGA (i.e., $H(\mathcal{S}^G, \tau^G)$) and obtained by the sequential brute-force algorithm with respect to small risk levels $\alpha \in [0.001, 0.1]$ in 30 trials.

where $\tilde{e}_{ij}$ indicates one realization of e_{ij}. If all vehicle–demand pairs are independent from each other, y models a multi-independent uniform distribution. We sample n_s times from underlying multi-independent uniform distribution of y and approximate the auxiliary function $H(\mathcal{S}, \tau)$ as

$$\hat{H}(\mathcal{S}, \tau) = \tau - \frac{1}{n_s \alpha} \sum_{\tilde{y}} \left[\left(\tau - \sum_{i \in N} \max_{j \in \mathcal{S}_i} \tilde{e}_{ij} \right)_+ \right].$$

We set the upper bound of the parameter τ as $\Gamma = N \max(\tilde{e}_{ij}), i = \{1, \ldots N\}, j = \{1, \ldots, R\}$, to make sure $\Gamma - f(\mathcal{S}, y) \geq 0$. We set the searching separation for τ as $\Delta = 1$.

After receiving the pair $(\mathcal{S}^G, \tau^G)$ from SGA, we plot the value of $H(\mathcal{S}^G, \tau^G)$ and $H(\mathcal{S}^G, \tau)$ with respect to different risk levels α in Fig. 5. Fig. 5(a) shows that $H(\mathcal{S}^G, \tau^G)$ increases when α increases. This suggests that SGA correctly maximizes $H(\mathcal{S}, \tau)$. Fig. 5(b) shows that $H(\mathcal{S}^G, \tau)$ is concave or piecewise concave, which is consistent with the property of $H(\mathcal{S}, \tau)$.

We plot the distribution of assignment utility

$$f(\mathcal{S}^G, y) = \sum_{i \in N} \max_{j \in \mathcal{S}_i^G} e_{ij}$$

in Fig. 6 by sampling $n_s = 1000$ times from the underlying distribution of y. $\mathcal{S}_i^G$ is a set of vehicles assigned to demand location i by SGA. $\mathcal{S}^G = \bigcup_{i=1}^N \mathcal{S}_i^G$. When the risk level α is small, vehicle–demand pairs with low utilities (equivalently, low uncertainties) are selected. This is because a small risk level

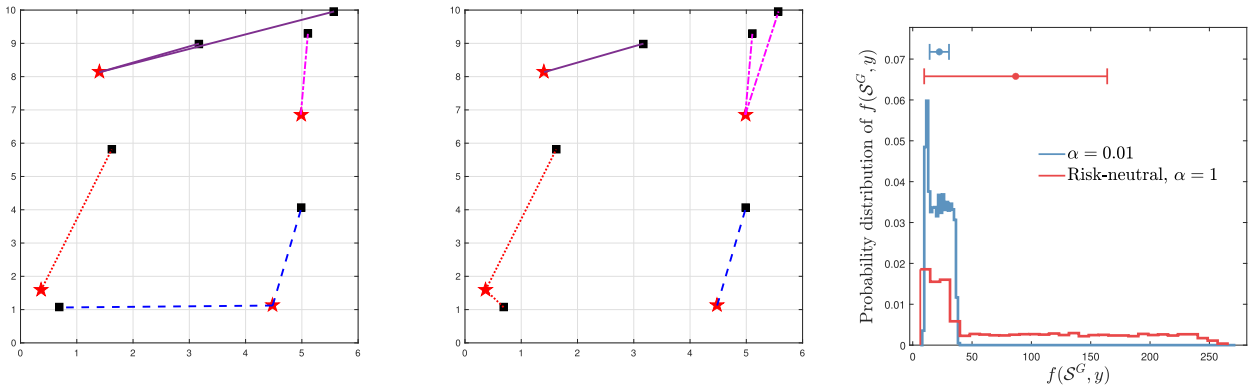


Fig. 9. Assignments and utility distributions by SGA with two extreme risk level values. The red solid star represents the demand location. The black solid square represents the vehicle position. The line between the vehicle and the demand represents an assignment. (a) Assignment when $\alpha = 0.01$. (b) Assignment when $\alpha = 1$ (risk-neutral). (c) Assignment utility distributions at $\alpha = 0.01$ and $\alpha = 1$.

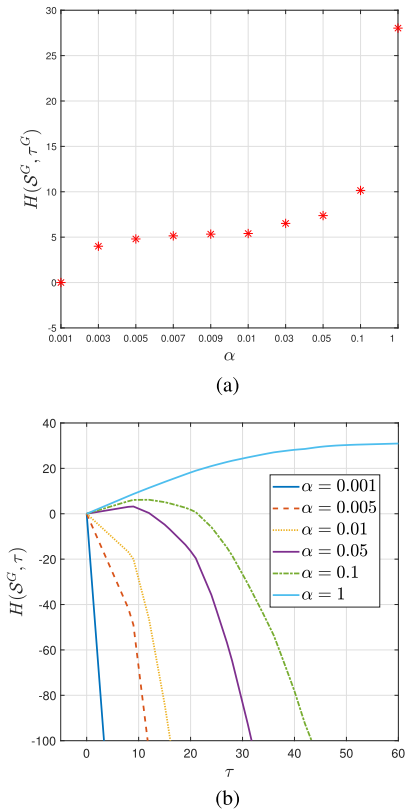


Fig. 10. Value of $H(\mathcal{S}, \tau)$ by SGA with respect to different risk levels. (a) Value of $H(\mathcal{S}^G, \tau^G)$ with respect to different risk levels α. (b) Function $H(\mathcal{S}^G, \tau)$ with respect to several risk levels α.

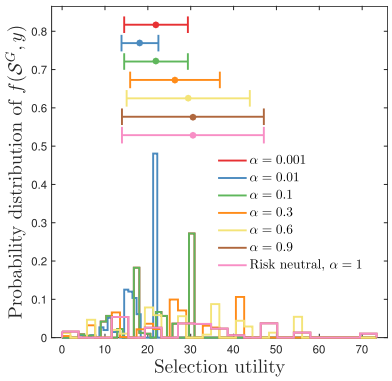


Fig. 11. Distribution of the selection utility $f(\mathcal{S}^G, y)$ by SGA.

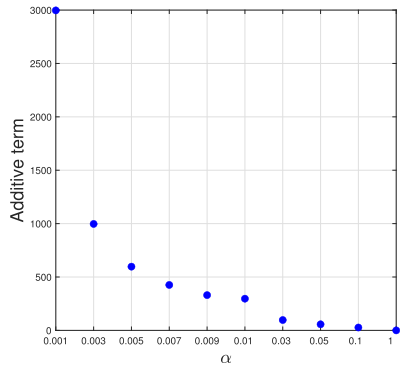


Fig. 12. Additive term (10) in the approximation ratio with respect to risk level α.

indicates that the assignment is conservative and only willing to take a little risk. Thus, lower utility with lower uncertainty is assigned to avoid the risk induced by the uncertainty. In contrast, when α is large, the assignment is allowed to take more risk to gain more assignment utility. Vehicle–demand pairs with high utilities (equivalently, high uncertainties) are selected in such a case.

Fig. 7 depicts the additive approximation error (10) of SGA with respect to the risk level α. It shows that when α is close to zero, SGA has a large approximation error and the error vanishes quickly when α increases (especially when $\alpha \geq 0.1$). Since SGA may not give a good solution with a large additive approximation error, it is worth evaluating how well SGA does when the risk level α is very small.

To this end, for $\alpha \in [0.001, 0.1]$, we compare SGA with a *sequential brute-force algorithm* that uses the brute-force search

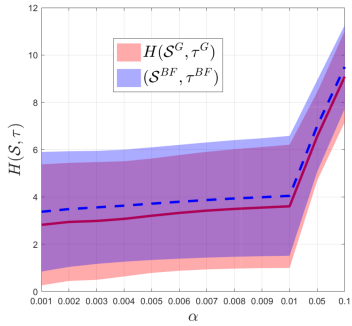


Fig. 13. Comparison of the values of $H(\mathcal{S}, \tau)$ obtained by SGA (i.e., $H(\mathcal{S}^G, \tau^G)$) and obtained by the sequential brute-force algorithm with respect to small risk levels $\alpha \in [0.001, 0.1]$ in 30 trials.

as a subroutine instead of the greedy algorithm (lines 5–8) in Algorithm 1, as shown in Fig. 8. That is, for each specific τ_i, we use the brute-force search to enumerate all possible sets $\mathcal{S}$ to find the optimal set $\mathcal{S}_i^{BF}$ that maximizes $H(\mathcal{S}, \tau_i)$. For example, in this mobility-on-demand case with six vehicles assigned to four demand locations, the total number of feasible assignments can be computed as the total number of ways to partition six vehicles into four disjoint sets (i.e., 65; see [37]) multiplying by the total number of permutations of four demand locations (i.e., 24), that is, $65 \times 24 = 1560$. Over all these 1560 possible assignments, we select out the assignment set $\mathcal{S}_i^{BF}$ that maximizes $H(\mathcal{S}, \tau_i)$ at each τ_i. Then, we choose the best pair $(\mathcal{S}^{BF}, \tau^{BF})$ that gives the maximum value of $H(\mathcal{S}, \tau)$, denoted as $H(\mathcal{S}^{BF}, \tau^{BF})$, over all $(\mathcal{S}_i^{BF}, \tau_i)$ pairs (collected when searching for all possible values of τ). By referring to the proof of the approximation ratio of SGA (Proof of Theorem 1 in the Appendix), it is easy to verify that the sequential brute-force algorithm gives the following guarantee for maximizing $H(\mathcal{S}, \tau)$:

$$H(\mathcal{S}^{BF}, \tau^{BF}) \geq H(\mathcal{S}^\star, \tau^\star) - \Delta - \epsilon \tag{14}$$

with probability at least $1 - \delta$, when the number of samples, $n_s = O(\frac{\Gamma^2}{\epsilon^2} \log \frac{1}{\delta}), \delta, \epsilon \in (0, 1)$. Different from SGA, the sequential brute-force algorithm does not have the additive approximation error.

We compare the performance of SGA and the sequential brute-force algorithm using the same positions of six vehicles and four demand locations randomly generated in 30 trials. Fig. 8 shows that SGA works comparatively with the sequential brute-force algorithm for maximizing $H(\mathcal{S}, \tau)$ on average when α is close to zero. This suggests SGA can still perform relatively well when it has a large additive approximation error due to very small risk levels.

We also compare SGA (using CVaR measure) with the greedy algorithm (using the expectation, i.e., risk-neutral measure [20]) in Fig. 9. Note that risk-neutral measure is a special case of CVaR_α measure when $\alpha = 1$. We give an illustrative example of the assignment by SGA for two extreme risk levels, $\alpha = 0.01$ and $\alpha = 1$. When α is small ($\alpha = 0.01$), the assignment is conservative, and thus, farther vehicles (with lower utility and lower uncertainty) are assigned to each demand [see Fig. 9(a)]. In contrast, when $\alpha = 1$, nearby vehicles (with higher utility and higher uncertainty) are selected for the demands [see Fig. 9(b)]. Even though the mean value of the assignment utility distribution is larger at $\alpha = 1$ than at $\alpha = 0.01$, it is exposed to the risk of receiving lower utility since the mean-std bar at $\alpha = 1$ has smaller left endpoint than the mean-std bar at $\alpha = 0.01$ [see Fig. 9(c)].

B. Robust Environment Monitoring

We consider selecting $M = 4$ locations from $N = 8$ candidate locations to place sensors in the environment (see Fig. 4). Denote the area of the free space as v^{free}. The positions of N candidate locations are randomly generated within the free space v^{free}. We calculate the visibility region for each sensor v_i by using the VisiLibity library [38]. Based on the sensor model discussed in Section III-B, we set the probability of success for each sensor i as

$$p_i = 1 - v_i / v^{\text{free}}$$

and model the working of each sensor as a Bernoulli distribution with p_i probability of success and $1 - p_i$ probability of failure. Thus, the random polygon monitored by each sensor A_i follows the distribution

$$\begin{cases} \Pr[A_i = v_i] = p_i \\ \Pr[A_i = 0] = 1 - p_i \end{cases}. \tag{15}$$

From the selection utility function (8), for any realization of y, say $\tilde{y}$, we have

$$f(\mathcal{S}, y) = \text{area}\left(\bigcup_{i=1:M} \tilde{A}_i \right)$$

where $\tilde{A}_i$ indicates one realization of A_i by sampling y. If all sensors are independent of each other, we can model the working of a set of sensors as a multi-independent Bernoulli distribution. We sample $n_s = 1000$ times from the underlying multi-independent Bernoulli distribution of y and approximate the auxiliary function $H(\mathcal{S}, \tau)$ as

$$\hat{H}(\mathcal{S}, \tau) = \tau - \frac{1}{n_s \alpha} \sum_{\tilde{y}} \left[\left(\tau - \bigcup_{i=1:M} \tilde{A}_i \right)_+ \right].$$

We set the upper bound for the parameter τ as the area of all the free space v^{free} and set the searching separation for τ as $\Delta = 1$.

We use SGA to find the pair $(\mathcal{S}^G, \tau^G)$ with respect to several risk levels α. We plot the value of $H(\mathcal{S}^G, \tau^G)$ for several risk levels in Fig. 10(a). A larger risk level gives a larger $H(\mathcal{S}^G, \tau^G)$, which means that the pair $(\mathcal{S}^G, \tau^G)$ found by SGA correctly maximizes $H(\mathcal{S}, \tau)$ with respect to the risk level α. Moreover, we plot functions $H(\mathcal{S}^G, \tau)$ for several risk levels α in Fig. 10(b). Note that $\mathcal{S}^G$ is computed by SGA at each τ. For each α, $H(\mathcal{S}^G, \tau)$ shows the concavity or piecewise concavity of function $H(\mathcal{S}, \tau)$.

Based on the $\mathcal{S}^G$ calculated by SGA, we sample $n_s = 1000$ times from the underlying distribution of y and plot the distribution of the selection utility

$$f(\mathcal{S}^G, y) = \bigcup_{i=1:M} A_i, i \in \mathcal{S}^G$$

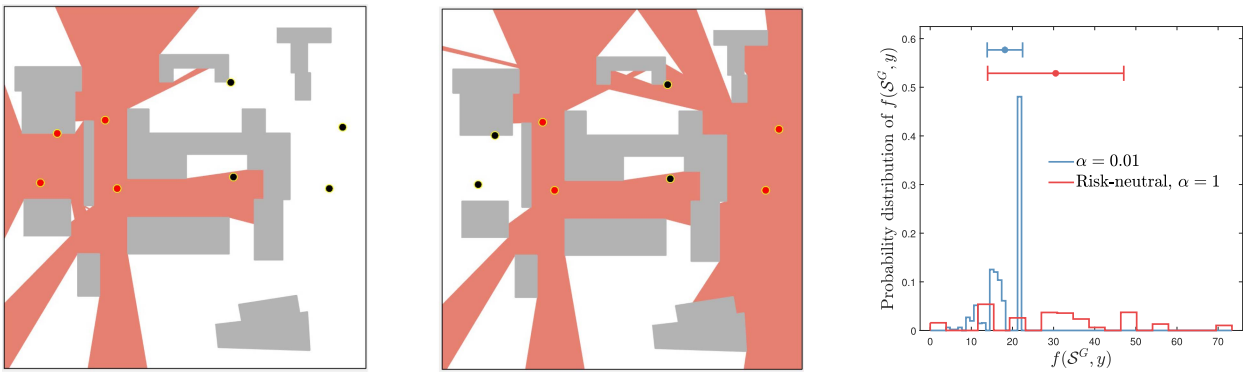


Fig. 14. Sensor selection and utility distributions by SGA with two extreme risk level values. The red solid circle represents the sensor selected by SGA. (a) Selection when $\alpha = 0.01$. (b) Selection when $\alpha = 1$ (risk-neutral). (c) Selection utility distributions at $\alpha = 0.01$ and $\alpha = 1$.

Fig. 15. Street network created by OSMnx in Blacksburg, VA, USA, with a shortest path (green path) from node v_i to node v_j (lighter blue dots).

in Fig. 11. Note that when the risk level α is small, the sensors with smaller visibility region and a higher probability of success should be selected. Lower risk level suggests a conservative selection. Sensors with a higher probability of success are selected to avoid the risk induced by the sensor failure. In contrast, when α is large, the selection would like to take more risk to gain more monitoring utility. The sensors with a larger visibility region and a lower probability of success should be selected. Fig. 11 demonstrates this behavior except when $\alpha = 0.001$. This is because when α is very small, the approximation error (10) is very large as shown in Fig. 12, and thus, SGA may not give a good solution.

Therefore, it is necessary to see how well SGA performs for small risk levels. Similar to the mobility-on-demand case, we compare SGA with a sequential brute-force algorithm that uses the brute-force search as a subroutine to enumerate all possible feasible placements for finding the optimal sensor placement $\mathcal{S}^{BF}$ for each specific τ_i. For instance, in this environment monitoring scenario with selecting four placement locations from eight candidate locations, there will be $\binom{8}{4} = 70$ placements in total. From these 70 possible placements, we choose the placement $\mathcal{S}_i^{BF}$ that maximizes $H(\mathcal{S}, \tau_i)$ at each τ_i. Then, we pick out the best pair $(\mathcal{S}^{BF}, \tau^{BF})$ that has the maximum value of $H(\mathcal{S}, \tau)$, denoted by $H(\mathcal{S}^{BF}, \tau^{BF})$, from all $(\mathcal{S}_i^{BF}, \tau_i)$ pairs (collected when searching for all possible values of τ). Note that the sequential brute-force algorithm has no additive error for maximizing $H(\mathcal{S}, \tau)$ (14).

With the same set of eight candidate locations randomly generated in 30 trials, we compare SGA with the sequential brute-force algorithm for small risk levels $\alpha \in [0.001, 0.1]$ in Fig. 13. The result shows that, on average, SGA works comparatively with the sequential brute-force algorithm (especially when $\alpha \geq 0.01$) for maximizing $H(\mathcal{S}, \tau)$. This implies that SGA can still work relatively well on average when the risk level is close to zero.

We also compare SGA by using CVaR measure with the greedy algorithm by using the expectation, i.e., risk-neutral measure (mentioned in [2, Sec. 6.1]) in Fig. 14. In fact, the risk-neutral measure is equivalent to the case of CVaR_α when $\alpha = 1$. We give an illustrative example of the sensor selection by SGA for two extreme risk levels, $\alpha = 0.01$ and $\alpha = 1$. When risk level α is small ($\alpha = 0.01$), the selection is conservative, and thus, the sensors with small visibility region are selected [see Fig. 14(a)]. In contrast, when $\alpha = 1$, the risk is neutral and the selection is more adventurous, and thus, sensors with large visibility region are selected [see Fig. 14(b)]. The mean-std bars of the selection utility distributions in Fig. 14(c) show that the selection utility at the expectation ($\alpha = 1$) has a larger mean value than the selection at $\alpha = 0.01$. However, the selection at $\alpha = 1$ has the risk of gaining lower utility since the left endpoint of mean-std bar at $\alpha = 1$ is smaller than the left endpoint of mean-std bar at $\alpha = 0.01$.

C. Online Resilient Mobility on Demand

In this section, we show the effectiveness of ATA (see Algorithms 2 and 3) by using the online mobility on demand with street networks. A detailed description of the settings is provided in the Appendix. We compare the performance of ATA with the other three assignment strategies, *offline*, *one-step*, and *all-step* assignments, in terms of the arrival time, the number of assignments, and the running time. The offline assignment is a clairvoyant strategy that knows the vehicles' arrival times (i.e., knows how the stochastic scenario will play out) exactly and uses

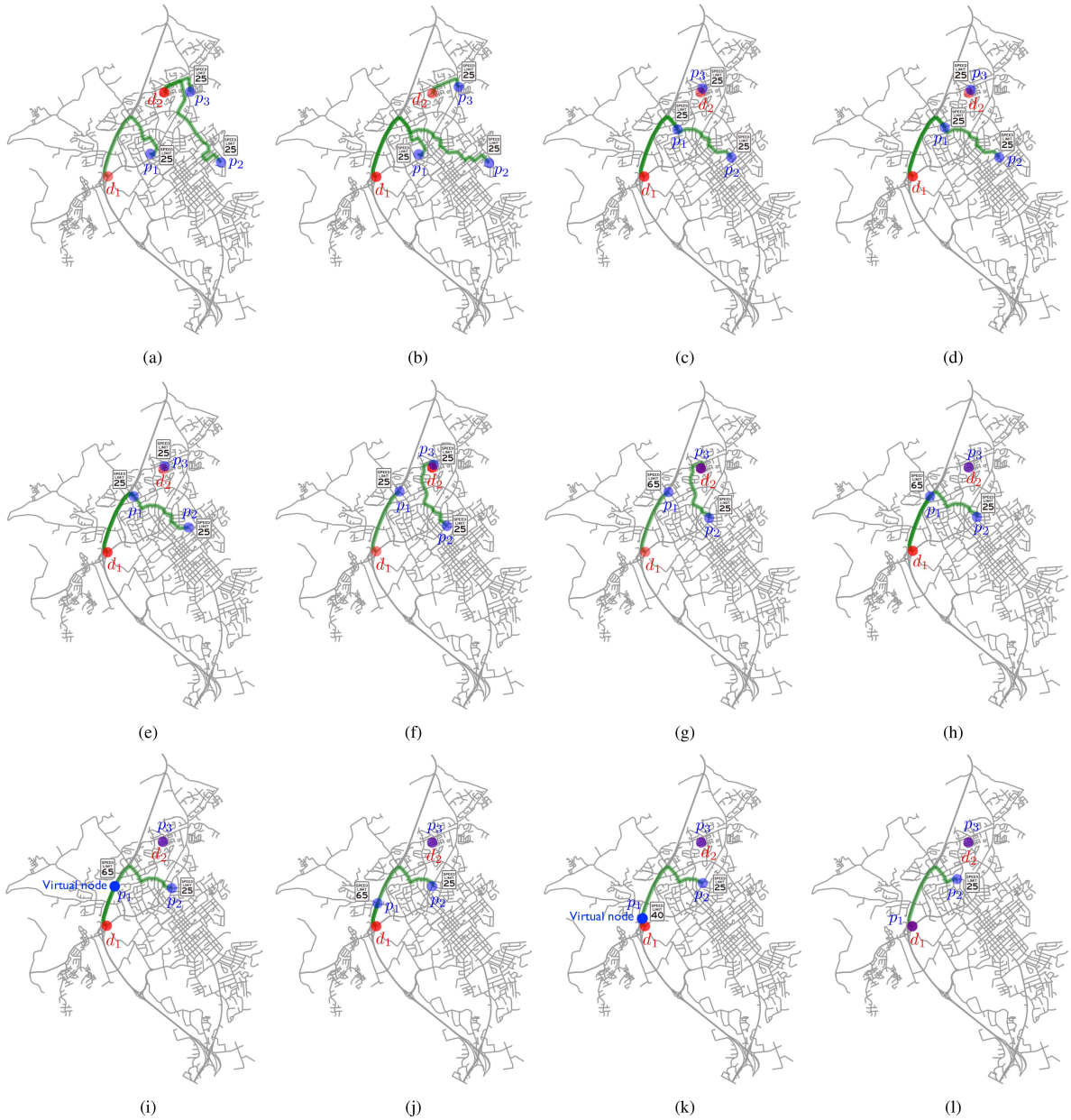


Fig. 16. One trial of ATA (see Algorithm 3) in action with three vehicles and two demand locations. The blue dots show the positions of the vehicles (p_1, p_2, p_3). The red dots show the demand locations (d_1, d_2). The speed sign shows the maximum speed on the street. (a) Step 1. (b) Step 2. (c) Step 19. (d) Step 20. (e) Step 22. (f) Step 23. (g) Step 28. (h) Step 29. (i) Step 30. (j) Step 31. (k) Step 32. (l) Step 34.

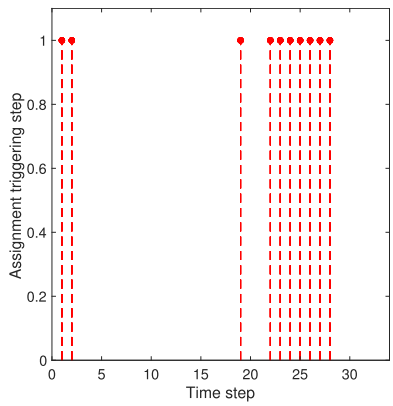


Fig. 17. Triggering time steps in one trial of ATA (see Algorithm 3) in action.

the standard greedy algorithm [14] to compute the assignment.[4] Thus, it can be set as a baseline. The difference of the three online assignment strategies, ATA, *one-step assignment*, and *all-step assignment*, comes from how often the assignment is scheduled. In the *one-step* assignment, the assignment is calculated at the initial time step and is fixed thereafter. In the *all-step* assignment, the assignment is computed at each time step. In ATA, the assignment is triggered only at specific time steps. In both the *one-step* and *all-step* assignments, all vehicles apply the "per-step online travel" rule (see Algorithm 3, lines 16–25) to go toward the assigned demand locations, as in the case of

[4]In the simulation, this is done by first storing the realistic arrival times of the vehicles at the final step and then using the greedy algorithm [14] to compute an offline assignment.

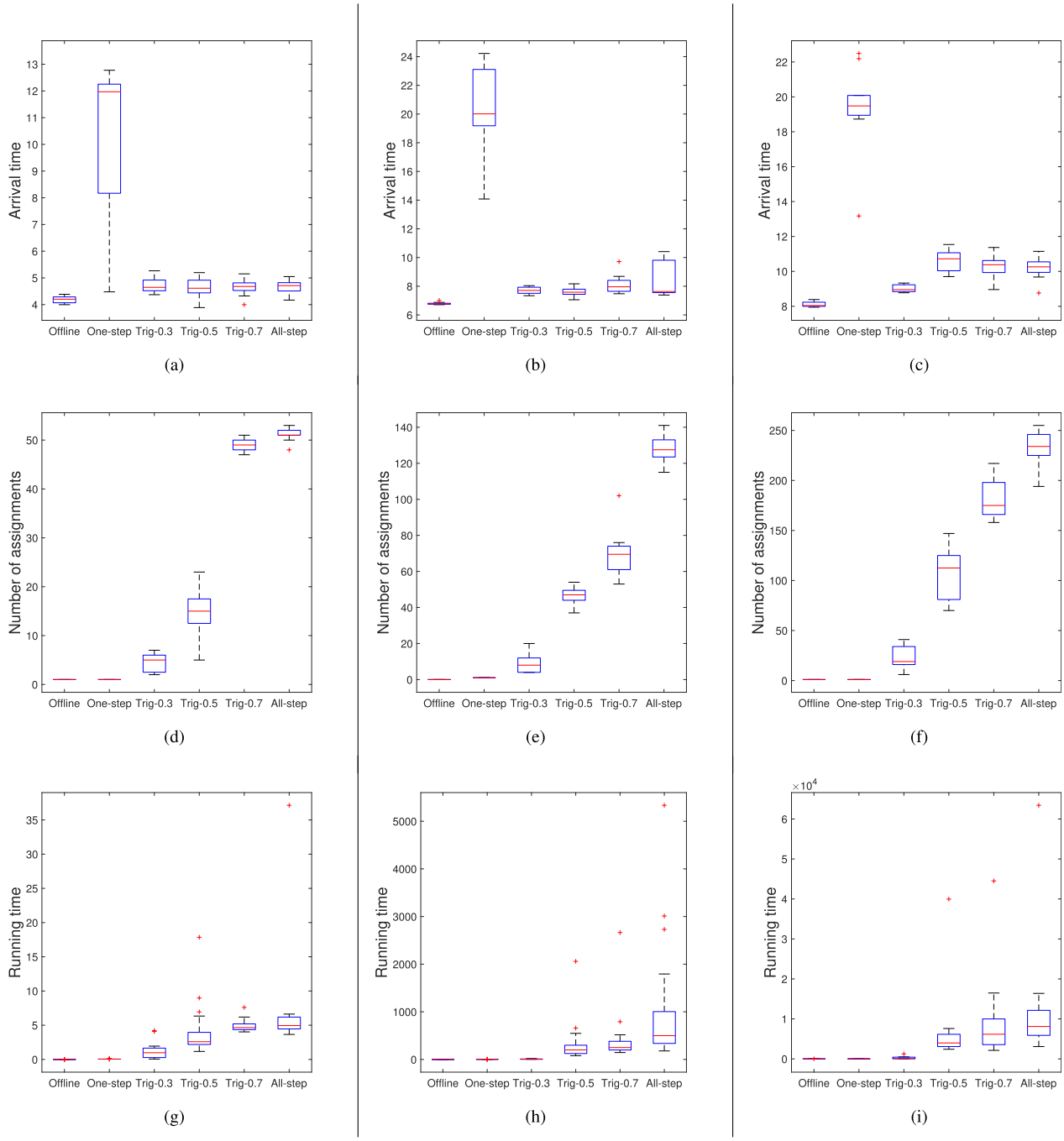
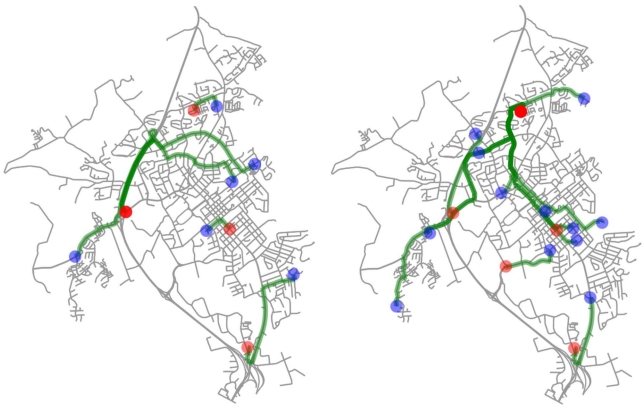


Fig. 18. Comparison of the number of assignments, arrival time, and running time by the *offline* assignment, *one-step* assignment, ATA (see Algorithm 3) with three triggering ratios, $\gamma = 0.3, 0.5, 0.7$, and *all-step* assignment. (a) Arrival time; three vehicles and two demands. (b) Arrival time; six vehicles and four demands. (c) Arrival time; 12 vehicles and five demands. (d) Number of assignments; three vehicles and two demands. (e) Number of assignments; six vehicles and four demands. (f) Number of assignments; 12 vehicles and five demands. (g) Running time; three vehicles and two demands. (h) Running time; six vehicles and four demands. (i) Running time, 12 vehicles and five demands.

ATA. Particularly, in these three online assignment strategies, we compute the arrival time as the summation of all per-step time intervals [$T^{\texttt{step}}$ in (20)] from time of the first step to the final step when all demand locations are reached.

We consider assigning R supply vehicles to N demand locations on a street network, $\mathcal{G}$, created using OSMnx [39]. OSMnx is a python package that allows for easily constructing, projecting, visualizing, and analyzing complex street networks [39]. We give an example showing a network of drivable public streets created by OSMnx in Blacksburg, VA, USA, in Fig. 15. OSMnx also allows for finding a shortest path between two nodes (intersections) of the networks by Dijkstra's algorithm if there exists one (see Fig. 15). Thus, we use OSMnx to compute a shortest path from each vehicle's initial position to



(a) (b)

Fig. 19. Initial assignment in (a) six-vehicle four-demand system and (b) 12-vehicle five-demand system.

each demand location, if there exists one. We can also extract the intersection degree and edge length of the street network directly by using OSMnx.

We model the waiting time at intersection v_i following a truncated normal distribution

$$t(v_i) \sim \mathcal{N}^{\texttt{Truc}}(0, \beta_1 \texttt{deg}(v_i)), t(v_i) \in [0, T(v_i)] \tag{16}$$

where $T(v_i)$ denotes the maximum waiting time at the intersection v_i. We set the variance of the waiting time to be proportional to the degree of the intersection, i.e., $\texttt{var}(t(v_i)) = \beta_1 \texttt{deg}(v_i)$ with $\beta_1 \in (0, +\infty)$. A larger degree means more incoming and outgoing edges, and thus, the traffic at this intersection is more likely to be congested, which increases the probability of higher waiting time. Notably, the truncated Gaussian distribution is just one possible way to capture the randomness of the waiting time. In practice, instead of assuming a particular distribution, one would use historic traffic data to model the distribution at each intersection and along every edge. Our algorithm works for any model that can be sampled.

We set $\beta_1 = 1$ and model the maximum waiting time as $T(v_i) = 5\texttt{deg}(v_i)$. The model assumes that a larger degree of an intersection implies a larger maximum waiting time there. We model the real-time waiting time at an intersection v_i for each vehicle i as a sample from the truncated normal distribution (16). In practice, the real-time waiting time can be acquired when the vehicle reaches the intersection.

We set $\beta_2 = 1$ and use the speed limits of the street network in Blacksburg from Google Maps to compute the edge travel time [see (17)]. If a demand location is not reachable from a vehicle's position, we set the path travel time (18) from the vehicle to the demand as infinity. We set the risk level $\alpha = 0.1$ in SGA (see Algorithm 1).

1) Qualitative Results: We first show one trial of ATA in action in Fig. 16, where we assign $R = 3$ vehicles to $N = 2$ demand locations with triggering ratio $\gamma = 0.5$. Since we model the real-time waiting time at each intersection as a random sample from its distribution, ATA can end up with different results in different trials, even with the same vehicles' initial positions and demand locations.

In Fig. 16, ATA starts at time step 1 [see Fig. 16(a)] and ends at time step 34 [see Fig. 16(l)] when the two demand locations, d_1 and d_2, are reached. Within these 34 time steps, the assignment is only triggered at ten time steps (steps 1, 2, 19, 22, 23, 24, 25, 26, 27, and 28), as shown in Fig. 17. At these ten steps, the triggering conditions with $\gamma = 0.5$ [see (21) and (22)] are satisfied. For example, at step 1 [see Fig. 16(a)], the path length from p_3 to d_2 is less than half of that from p_2 to d_2 (21) and the path degree from p_3 to d_2 is less than that from p_2 to d_2 (22). In this case, there is no need to continue assigning vehicle 2 to d_2 since it is likely that vehicle 3 will reach d_2 earlier than vehicle 2. Therefore, triggering the reassignment to assign vehicle 2 to d_1 could be helpful in reducing the arrival time at d_1 [step 2; see Fig. 16(b)]. Similar assignment flip happens from step 22 [see Fig. 16(e)] to step 23 [see Fig. 16(f)], where vehicle 2 is switched to be assigned to d_2. However, the assignment does not flip at all triggering time steps. For example, even though the assignment is triggered at step 19 (see Fig. 17), the assignment remains the same at step 20 [see Fig. 16(c) and (d)]. One possible reason could be that assigning vehicle 2 to d_1 is still more helpful, since vehicle 3 is already very close to d_2 [see Fig. 16(c) and (d)]. The ATA also guarantees that once a demand is reached, e.g., in step 28, d_2 is reached by vehicle 3 [see Fig. 16(g)], no vehicle will be assigned to it later [see Fig. 16(h)–(l)]. Particularly, in this three-vehicle two-demand system, there is not even reassignment triggered afterward (see Fig. 17). In Fig. 16(i) and (k), we create virtual nodes (darker blue dots) to denote the intermediate positions of the vehicle on the highway.

2) Quantitative Results: We then compare the arrival time, number of assignments, and running time of these three assignment strategies in Fig. 18. We execute all three assignment strategies in ten trials with the same initial vehicles' positions and demand locations. In particular, we consider three scales of vehicle–demand system, e.g., $R = 3, N = 2$ (see Fig. 16), $R = 6, N = 4$ [see Fig. 19(a)], and $R = 12, N = 5$ [see Fig. 19(b)]. For each scale of the vehicle–demand system, we evaluate three different triggering ratios, $\gamma = \{0.3, 0.5, 0.7\}$, in each trial.

Fig. 18(a)–(c) shows that the *offline assignment* performs the best in terms of the arrival time, which is intuitive given that it knows the arrival times exactly, while the other three online assignments have the uncertain knowledge of the arrival times only. We observe that ATA with some suitable triggering ratios [e.g., $\gamma = 0.3, 0.5, 0.7$ in Fig. 18(a), $\gamma = 0.3, 0.5$ in Fig. 18(b), and $\gamma = 0.3$ in Fig. 18(c)] performs close to the offline strategy, which shows the effectiveness of ATA. In addition, note that ATA and the *all-step* assignment reach all demand locations earlier than the *one-step* assignment. This is expected since in ATA and the *all-step* assignment, the vehicles are reassigned to reduce the arrival time using the real-time information, while the *one-step* assignment only assigns the vehicles at the initial time step. For the same reason, the *one-step* assignment has less running time and fewer assignments, as shown in Fig. 18(d)–(i).

Fig. 18(d)–(i) shows that ATA schedules fewer assignments and uses less running time than that of the *all-step* assignment.

In particular, Fig. 18(d)–(f) shows that ATA with $\gamma = 0.5$ avoids at least half of the assignments compared to the *all-step* assignment. Even with fewer assignments, ATA performs comparably with or in some cases even better than *all-step* assignment in terms of the arrival time, as shown in Fig. 18(a)–(c). Particularly, with respect to the average of arrival time, ATA with triggering ratio $\gamma = 0.5$, $\gamma = 0.5$, and $\gamma = 0.3$ performs better than the *all-step* assignment in three-vehicle two-demand [see Fig. 18(a)], six-vehicle four-demand [see Fig. 18(b)], and 12-vehicle five-demand [see Fig. 18(c)] systems, respectively. This result echoes the analysis at the end of Section V that more assignments may not always help because the assignment using the traffic conditions revealed so far is nearsighted.

We also observe from Fig. 18(d)–(i) that a larger triggering ratio γ leads to more assignments and more running time of ATA. This is because, with a larger triggering ratio, the assignment will be triggered more frequently, as shown in Algorithm 2, line 6. Particularly, for three triggering ratios $\gamma = 0.3, 0.5, 0.7$, ATA with $\gamma = 0.3$ schedules fewer assignments [see Fig. 18(d)–(f)] and spends less running time [see Fig. 18(g)–(i)], while achieving a comparable [see Fig. 18(a) and (b)] or even better [see Fig. 18(c)] performance than that of ATA with $\gamma = 0.5, 0.7$. This again verifies the fact that more assignments during the online mobility-on-demand process may be unhelpful and even misleading.

VII. Conclusion

In this article, we studied a risk-aware discrete submodular maximization problem. We provided the first positive results for discrete CVaR submodular maximization for selecting a set under matroid constraints. In particular, we proposed SGA and analyzed its approximation ratio and the running time. We also studied the problem of the adaptive risk-aware submodular problem for the online version of the mobility-on-demand and environmental monitoring cases. We presented a triggering replanning strategy for this problem. In particular, for online mobility on demand, we designed an ATA strategy (see Algorithm 2) to reduce unnecessary computation. We demonstrated the two practical use cases of the CVaR submodular maximization problem and verified the effectiveness of ATA approach (see Algorithm 2).

Notably, our SGA works for any matroid constraint. In particular, the multiplicative approximation ratio can be improved to $1/c_f(1 - e^{-c_f})$ if we know that the constraint is a uniform matroid [34, Th. 5.4].

The additive term in our analysis depends on α. This term can be large when the risk level α is very small. Even though, in practice, SGA works relatively well on average with small risk levels (see Figs. 8 and 13), our ongoing work is to remove this dependence on α, perhaps by designing another algorithm specifically for low risk levels. We note that if we use an optimal algorithm instead of the greedy algorithm as a subroutine, then the additive term disappears from the approximation guarantee (e.g., the sequential brute-force algorithm). The algorithm also requires knowing Γ. We showed how to find Γ (or an upper bound for it) for the two case studies considered in this article. Devising a general strategy for finding Γ is part of our ongoing work.

Note that, in ATA, increasing the triggering ratio γ results in more assignments but may not always improve the performance (i.e., achieving a shorter arrival time), as shown in Fig. 18. Thus, our second future research direction is to design a generic approach for deciding the best triggering ratio γ.

Our third line of ongoing work focuses on applying the risk-aware strategy to multivehicle routing, patrolling, and informative path planning in dangerous environments [40] and mobility on demand with real-world datasets (2014 NYC Taxicab Factbook).[5]

In the mobility-on-demand application, we define the objective function as the sum of arrival utilities at all demand locations (7). The "sum" metric captures total assignment utility and is commonly used in vehicle assignment problems (see [20] and [41]). However, in the mobility-on-demand scenario, one typically expects all demand locations to be reached by vehicles within the least time. Hence, a more desirable objective would be to maximize the *minimum arrival utility* (or equivalently, to minimize the *maximum arrival time*) to each demand location [42]. However, the minimum arrival utility (or maximum arrival time) is not submodular (or supermodular) in the decision set $\mathcal{S}$, so that Algorithm 1 may not be directly utilized or have a bounded approximation guarantee to optimize it. Therefore, our fourth future research avenue is to revisit the mobility-on-demand application with the objective of maximizing minimum arrival utility (or minimizing maximum arrival time) and, building on Algorithm 1, design corresponding risk-aware algorithms with hard guarantees.

As a final comment, we discuss the reason for choosing the CVaR over other risk-aware measures such as mean–variance [27] and VaR [25]. In general, this is a design choice and the CVaR may not be the best choice in some cases. Generally speaking, for the utility with Gaussian distribution, it is more convenient to use the mean–variance measure given that it has a closed-form expression [43] and is equivalent to the CVaR with appropriate risk parameters selected. However, since it needs to know the mean and variance of the distribution, it does not work for arbitrary distributions (without a closed-form expression). The VaR measures how much a low utility we might get with a given probability. If the distribution of the utility is smooth (or stable), the VaR is sufficient for measuring the risks. However, the more fluctuant the utility, the greater the chance that VaR will not quantify the risks well, as it only cares about a single cutoff point. The CVaR is a more suitable measure for this scenario by considering all the bad tailed cases. The CVaR is also more desirable for the cases where the bad low probability (tailed) events have large value or impact and can cause huge losses since it is more conservative than the VaR (see Fig. 2). Furthermore, the CVaR has nicer properties than mean–variance and VaR for capturing risks in robotics [28] and is easier for optimization when it is computed by drawing samples [22].

[5][Online]. Available: http://www.nyc.gov/html/tlc/downloads/pdf/2014_taxicab_fact_book.pdf

APPENDIX

We start by describing the settings for the online mobility-on-demand with street networks. We then introduce the corresponding OTA in Algorithm 3 based on these settings. After that, we provide proofs for the lemmas and the theorem in this article.

A. Online Mobility on Demand with Street Networks

We consider the vehicles travel on a street network $\mathcal{G} = \{\mathcal{V}, \mathcal{W}\}$ with the intersections as nodes $\mathcal{V} = \{v_1, v_2, \ldots, v_V\}$ and the driveway connecting two adjacent intersections, v_i and v_j, as edges $w_{v_i,v_j} \in \mathcal{W} \subseteq \mathcal{V} \times \mathcal{V}$. Notably, the street network $\mathcal{G}$ is a directed graph because of the driving direction on the street. We use $\texttt{deg}(v_i)$ to denote the degree of a node, which is the number of the incoming and outgoing edges of the node.

For an edge w_{v_i,v_j}, we denote its length by $\texttt{len}(w_{v_i,v_j})$. Denote a path P_{v_i,v_k} from node v_i to node v_k as a series of orderly connected edges on the graph with the following form: $\{(v_i, v_j), (v_j, v_m), \ldots, (v_n, v_k)\}$. We compute the length $\texttt{len}(P_{v_i,v_k})$ and the degree $\texttt{deg}(P_{v_i,v_k})$ of path P_{v_i,v_k} by the summation of the lengths of all edges and the summation of the degrees of all nodes on the path, respectively. Denote the set of all nodes on path P_{v_i,v_k} as $\mathcal{V}(P_{v_i,v_k})$.

Owing to the unpredictable traffic condition at the intersection (node), we assume that the waiting time at an unseen intersection v_i is a random variable, denoted as $t(v_i)$. We assume that the variance of the waiting time, $\texttt{var}(t(v_i))$, is proportional to the degree of the intersection, because, for an intersection, a larger degree means more incoming and outgoing edges, and thus, the traffic condition at this intersection can have larger uncertainty. However, when vehicle j arrives at an intersection, it can acquire the real-time traffic condition and the waiting time $t_j^{\texttt{wait}}$ at the intersection.

We compute the time to traverse an edge w_{v_i,v_j} by

$$t(w_{v_i,v_j}) = \beta_2 \texttt{len}(w_{v_i,v_j})/\texttt{maxv}(w_{v_i,v_j}) \tag{17}$$

where $\texttt{maxv}(w_{v_i,v_j})$ denotes the limited speed on edge w_{v_i,v_j} and $\beta_2 \in [1, +\infty)$. Here, we do not consider the uncertain traffic condition when the vehicle is traveling on the edges. Thus, as shown in (17), the edge travel time can be obtained beforehand. Then, the time to traverse a path P_{v_i,v_j} combines the waiting time at all the intersections and the time to travel through all the edges on the path. That is,

$$t(P_{v_i,v_j}) = \sum_{v \in \mathcal{V}(P_{v_i,v_j})} t(v) + \beta_2 \sum_{w \in P_{v_i,v_j}} t(w). \tag{18}$$

Notably, the path travel time $t(P_{v_i,v_j})$ is a random variable with its randomness induced by the waiting time. Similarly, we compute the path travel time by using a sampling method—taking an equal number of samples of the waiting time at each intersection and then using (18) with appropriate dimension manipulation. Clearly, the arrival utility to v_j from v_i through traveling along path P_{v_i,v_j} can be computed as

$$e(P_{v_i,v_j}) = 1/t(P_{v_i,v_j}). \tag{19}$$

We denote the time that each vehicle j needs to arrive at the next intersection as $t_j^{\texttt{next}}$. Then, the per-step time interval is computed as

$$T^{\texttt{step}} = \min_{j \in \{1,\ldots,R\}} t_j^{\texttt{next}}. \tag{20}$$

Notably, the per-step time interval $T^{\texttt{step}}$ depends on $t_j^{\texttt{next}}$ from all the vehicles and can be changing at each time step. In addition, when computing $T^{\texttt{step}}$ at each step, some vehicles may wait at the intersections, whereas the others may be traveling on the edges.

For simplicity, we assume that the vehicles' initial positions and the demand locations are some nodes (intersections) of graph $\mathcal{G}$.[6]

B. Online Triggering Assignment on Street Networks

In this section, we explicitly introduce our online triggering assignment strategy on street networks in Algorithm 3.

Initial assignment (see Algorithm 3, lines 1–3): We first find a shortest path from each vehicle's initial position to each demand location by using Dijkstra's algorithm (see Algorithm 3, line 1). Then, we use (18) and (19) to compute the arrival utility of these shortest paths (see Algorithm 3, line 2). Finally, we select a risk level α and use SGA (see Algorithm 1) to get an initial vehicle-to-demand assignment $\mathcal{S}$ with $\mathcal{S} = \bigcup_{i=1}^{N} \mathcal{S}_i$ (see Algorithm 3, line 3).

After the initial assignment, each vehicle j knows its assigned demand location d_i with the corresponding shortest path, $P^{\star}_{v(p_j),v(d_i)}$ (computed in Algorithm 3, line 1). Then, each vehicle travels on its shortest path towards its assigned demand location. Notably, the demand location and the corresponding (shortest) path assigned to each vehicle can be changing because of the online reassignment later on.

Our terminated condition is to guarantee that all demand locations are reached (see Algorithm 3, line 5), which needs an underlying assumption that the number of vehicles is larger than the number of demand locations, i.e., $R \geq N$.

Per-step time interval update (see Algorithm 3, lines 4 and 6–15): To update the per-step time interval $T^{\texttt{step}}$ at each time step, each vehicle j stores the previous intersection it was in, $v_j^{\texttt{prev}}$, which is initialized by an empty set (see Algorithm 3, line 4).

1) If vehicle j arrives at a new intersection $v(p_j)$ (see Algorithm 3, line 7), it updates its previous intersection by the current new intersection (see Algorithm 3, line 8) and acquires the real-time waiting time at the new intersection $v(p_j)$ (see Algorithm 3, line 9). In addition, it computes the time to spend for arriving at the next intersection $v(p_j) + 1$ as $t_j^{\texttt{next}}$ (see Algorithm 3, line 10).
2) If vehicle j has not reached a new intersection (see Algorithm 3, line 11), it reduces the time to reach the new intersection by the per-step time interval (at the last time step) $T^{\texttt{step}}$ (see Algorithm 3, line 12).
3) After all vehicles computing $t_j^{\texttt{next}}$, they select the minimum $t_j^{\texttt{next}}$ as the current per-step time interval (see Algorithm 3, line 15).

[6]Indeed, if the positions of some vehicles are not at the intersections, we can easily create corresponding virtual intersections for them and add the virtual intersections on the graph.

Algorithm 3: Triggering Assignment on Street Networks.

Input: • Graph $\mathcal{G}$ with intersections $\mathcal{V}$ and edges $\mathcal{W}$
• N demand locations d_i with corresponding intersections, $v(d_i), i \in \{1, \ldots, N\}$. R vehicles' initial positions p_j with corresponding intersections, $v(p_j), j \in \{1, \ldots, R\}$

1: Find a shortest path $P^\star_{v(p_j),v(d_i)}$ from each $v(p_j)$ to each $v(d_i)$. Collect these shortest paths as $\{P^\star_{v(p_j),v(d_i)}\}$
2: Compute the arrival utility of these shortest paths as $e(\{P^\star_{v(p_j),v(d_i)}\})$ by (18) and (19)
3: $\mathcal{S} = \text{SGA}(e(\{P^\star_{v(p_j),v(d_i)}\}), \alpha)$
4: Set $v_j^{\texttt{prev}} = \emptyset, t_j^{\texttt{next}} = 0, j \in \{1, \ldots, R\}$ and $T^{\texttt{step}} = 0$
5: **while** not all N demand locations are reached **do**
6: **for** each vehicle $j \in \{1, \ldots, R\}$ **do**
7: **if** $v(p_j) \neq v_j^{\texttt{prev}}$ **then**
8: updates $v_j^{\texttt{prev}} \leftarrow v(p_j)$
9: acquires real-time $t_i^{\texttt{wait}}$ at intersection $v(p_j)$
10: computes $t_j^{\texttt{next}} = t_j^{\texttt{wait}} + t(e_{v(p_j),v(p_j)+1})$ (17)
11: **else**
12: $t_j^{\texttt{next}} = t_j^{\texttt{next}} - T^{\texttt{step}}$
13: **end if**
14: **end for**
15: update $T^{\texttt{step}} = \min_{j\in\{1,\ldots,R\}} t_j^{\texttt{next}}$ (20)
16: **for** each vehicle $j \in \{1, \ldots, R\}$ **do**
17: **if** $t_j^{\texttt{next}} == T^{\texttt{step}}$ **then**
18: updates $v(p_j) \leftarrow v(p_j) + 1$
19: **else if** $t_j^{\texttt{wait}} >= T^{\texttt{step}}$ **then**
20: updates $t_j^{\texttt{wait}} = t_j^{\texttt{wait}} - T^{\texttt{step}}$
21: **else if** $t_j^{\texttt{wait}} < T^{\texttt{step}}$ **then**
22: updates $t_j^{\texttt{wait}} = 0$
23: travels $\texttt{maxv}(e_{v(p_j),v(p_j)+1})(t_j^{\texttt{next}} - T^{\texttt{step}})/\beta_2$ distance on the current edge $e_{v(p_j),v(p_j)+1}$
24: **end if**
25: **end for**
26: Check for all demand locations $d_i, i \in \{1, \ldots, N\}$
27: **if** $\exists \texttt{len}(P^\star_{v(p_j),v(d_i)}) \leq \gamma \texttt{len}(P^\star_{v(p'_j),v(d_i)})$ and $\texttt{deg}(P^\star_{v(p_j),v(d_i)}) \leq \texttt{deg}(P^\star_{v(p'_j),v(d_i)})$ **then**
28: Find a shortest path $P^\star_{v(p_j),v(d_i)}$ from each $v(p_j)$ to each $v(d_i)$. Collect these shortest paths as $\{P^\star_{v(p_j),v(d_i)}\}$
29: Compute the arrival utility of these shortest paths as $E(\{P^\star_{v(p_j),v(d_i)}\})$ by (18) and (19)
30: $\mathcal{S} = \texttt{SGA}(e(\{P^\star_{v(p_j),v(d_i)}\}), \alpha)$,
31: **end if**
32: **end while**

Per-step online travel (see Algorithm 3, lines 16–25): Based on the current per-step time interval $T^{\texttt{step}}$, each vehicle j can decide its motion at the current time step.

1) If vehicle j can reach the next intersection $v(p_j) + 1$ within per-step time $T^{\texttt{step}}$ (see Algorithm 3, line 17), it moves to $v(p_j) + 1$ and updates its current intersection $v(p_j)$ by the next intersection $v(p_j) + 1$ (see Algorithm 3, line 18).
2) If vehicle j's waiting time $t_j^{\texttt{wait}}$ at the current intersection $v(p_j)$ is larger than per-step time $T^{\texttt{step}}$ (see Algorithm 3, line 19), it still needs to wait at $v(p_j)$ after per-step time $T^{\texttt{step}}$, but its waiting time is reduced by $T^{\texttt{step}}$ (see Algorithm 3, line 20).
3) If vehicle j's waiting time $t_j^{\texttt{wait}}$ at the current intersection $v(p_i)$ is less than per-step time $T^{\texttt{step}}$ (see Algorithm 3, line 21), it will leave the current intersection $v(p_j)$ or it is already traveling on the current edge $e_{v(p_j),v(p_j)+1}$, i.e., $t_j^{\texttt{wait}} = 0$. In both cases, even though vehicle j cannot reach the next intersection $v(p_j) + 1$ (since $t_j^{\texttt{next}} > T^{\texttt{step}}$), it will traverse some distance on the current edge (see Algorithm 3, line 23), which is computed by leveraging on (17) as

$$\begin{aligned}
&\texttt{len}(e_{v(p_j),v(p_j)+1}) \frac{t_j^{\texttt{next}} - T^{\texttt{step}}}{t(e_{v(p_j),v(p_j)+1})} \\
&= \texttt{len}(e_{v(p_j),v(p_j)+1}) \frac{t_j^{\texttt{next}} - T^{\texttt{step}}}{\beta_2 \frac{\texttt{len}(e_{v(p_j),v(p_j)+1})}{\texttt{maxv}(e_{v(p_j),v(p_j)+1})}} \\
&= \texttt{maxv}(e_{v(p_j),v(p_j)+1})(t_j^{\texttt{next}} - T^{\texttt{step}})/\beta_2.
\end{aligned}$$

Triggering reassignment (see Algorithm 3, lines 26–31): We trigger the reassignment when some condition happens. Recall that, for each demand location d_i, it has a set of vehicles $\mathcal{S}_i$ assigned to it. Thus, in total, there are $|\mathcal{S}_i|$ shortest paths to demand d_i. We check for all demand locations (see Algorithm 3, line 26) and trigger the reassignment as long as there exists a demand location i with

$$\texttt{len}(P^\star_{v(p_j),v(d_i)}) \leq \gamma \texttt{len}(P^\star_{v(p'_j),v(d_i)}) \tag{21}$$

$$\texttt{deg}(P^\star_{v(p_j),v(d_i)}) \leq \texttt{deg}(P^\star_{v(p'_j),v(d_i)}) \tag{22}$$

where $\gamma \in (0, 1)$ denotes the triggering ratio (see Algorithm 3, line 27). Because, in this case, there exists a path $P^\star_{v(p_j),v(d_i)}$ that has a shorter travel length and smaller travel time uncertainty (remember we use the degree to capture the waiting time uncertainty), than that of the other path $P^\star_{v(p'_j),v(d_i)}$. Thus, there is no need to continue assigning the vehicle with path $P^\star_{v(p'_j),v(d_i)}$ to the demand i. We, therefore, trigger the reassignment and hopefully assign this vehicle to other demand location to reduce the total travel time (see Algorithm 3, lines 28–30). Notably, the triggering ratio regulates the frequency of reassignment, i.e., a larger γ triggers more reassignments.

C. Proofs

Proof of Lemma 1: Since $f(\mathcal{S}, y)$ is monotone increasing and submodular in $\mathcal{S}$, $\max\{\tau - f(\mathcal{S}, y), 0\}$ is monotone decreasing and supermodular in $\mathcal{S}$, and its expectation is also monotone decreasing and supermodular in $\mathcal{S}$. Then, $H(\mathcal{S}, \tau)$ is monotone increasing and submodular in $\mathcal{S}$.

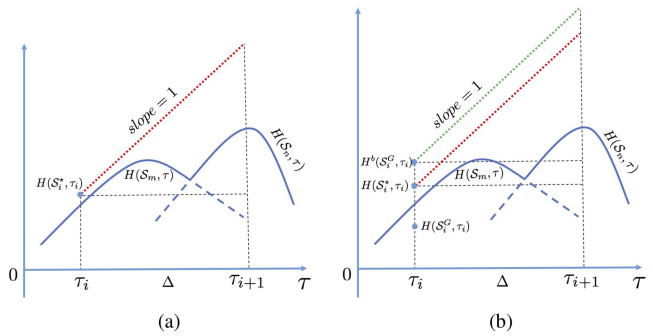


Fig. 20. Illustration of $H(\mathcal{S},\tau)$ within $\tau\in[\tau_i,\tau_{i+1}]$. (a) $H(\mathcal{S},\tau)$ is under the red-dotted line. (b) $H(\mathcal{S},\tau)$ is under the red-dotted line and green-dotted line.

$H(\emptyset,\tau)=\tau(1-\frac{1}{\alpha})$ given $f(\mathcal{S},y)$ is normalized ($f(\emptyset,y)=0$). Thus, $H(\mathcal{S},\tau)$ is not necessarily normalized since τ is not necessarily zero. See a similar proof in [22] and [29]. ■

Proof of Lemma 2: Since $\max(\tau-f(\mathcal{S},y),0)$ is convex in τ, its expectation is also convex in τ. Then $-\frac{1}{\alpha}\mathbb{E}[\max(\tau-f(\mathcal{S},y),0)]$ is concave in τ and $H(\mathcal{S},\tau)$ is concave in τ.

Proof of Lemma 3: By using the result in [22, Lemma 1 and Proof of Theorem 1], we know that $H(\mathcal{S},\tau)$ is concave and continuously differentiable with derivative given by

$$\frac{\partial H(\mathcal{S},\tau)}{\partial\tau}=1-\frac{1}{\alpha}(1-\Phi(f(\mathcal{S},y)))$$

where $\Phi(f(\mathcal{S},y))$ is the cumulative distribution function of $f(\mathcal{S},y)$. Thus, $0\le\Phi(f(\mathcal{S},y)))\le 1$, which proves the lemma.

Proof of the number of samples n_s: We reach the statement by using the proof by Ohsaka and Yoshida [30, Lemmas 4.1–4.5]. We have via taking $c=\Gamma$ (the upper bound of the search separation τ) in [30, Lemma 4.4]. Then, in [30, Lemma 4.5], to make

$$|\text{CVaR}_\alpha(X)-\text{CVaR}_\alpha(\hat{X})|\le\epsilon \tag{23}$$

we need to set

$$\sup_{x\in\mathbb{R}}|F_X(x)-F_{\hat{X}}(x)|\le\frac{\epsilon}{\Gamma}. \tag{24}$$

By the Dvoretzky–Kiefer–Wolfowitz inequality, inequality (24) fulfills with probability at least $1-2\exp(-2n_s\frac{\epsilon^2}{\Gamma^2})$. Thus, by setting $n_s=O(\frac{\Gamma^2}{\epsilon^2}\log\frac{1}{\delta}),\delta,\epsilon\in(0,1)$, we have $1-2\exp(-2n_s\frac{\epsilon^2}{\Gamma^2})=1-\delta$.

Proof of Lemma 4: Denote $H_i^\star=\max H(\mathcal{S},\tau)$ with $\tau\in[i\Delta,(i+1)\Delta),\mathcal{S}\in\mathcal{I}$. From Lemmas 2 and 3, we know $H(\mathcal{S},\tau)$ is concave in τ and $\frac{\partial H(\mathcal{S},\tau)}{\partial\tau}<1$. The properties of concavity and bound on the gradient give

$$H_i^\star-H(\mathcal{S}_i^\star,\tau_i)\le\Delta.$$

We illustrate this claim by using Fig. 20(a). Since $\mathcal{S}_i^\star$ is the optimal set at τ_i for maximizing $H(\mathcal{S},\tau)$, the value of $H(\mathcal{S},\tau)$ with any other set $\mathcal{S}\in\mathcal{I}$ at τ_i is at most $H(\mathcal{S}_i^\star,\tau_i)$. That is, $H(\mathcal{S},\tau_i)\le H(\mathcal{S}_i^\star,\tau_i)$. Since $H(\mathcal{S},\tau)$ is a concave function of τ for any specific $\mathcal{S}$, $H(\mathcal{S},\tau)$ can be a single concave function, i.e., $H(\mathcal{S}_m,\tau)$ or $H(\mathcal{S}_n,\tau)$ or a piecewise concave function by a combination of several concave functions, i.e., a combination of $H(\mathcal{S}_m,\tau)$ and $H(\mathcal{S}_n,\tau)$ during $\tau\in[\tau_i,\tau_{i+1}]$ [see Fig. 20(a)]. In either case, $H(\mathcal{S},\tau)$ is below the line starting at $H(\mathcal{S}_i^\star,\tau_i)$ with $slope=1$ during $\tau\in[\tau_i,\tau_{i+1}]$ (the red-dotted line in Fig. 20(a)). Since $H(\mathcal{S},\tau_i)\le H(\mathcal{S}_i^\star,\tau_i)$ and $H(\mathcal{S},\tau)$ has a bounded gradient $\frac{\partial H(\mathcal{S},\tau)}{\partial\tau}\le 1$. Thus, $H_i^\star-H(\mathcal{S}_i^\star,\tau_i)\le\frac{\partial H(\mathcal{S},\tau)}{\partial\tau}\Delta=\Delta,\forall i\in\{0,1,\dots,\lceil\frac{\Gamma}{\Delta}\rceil\}$.

Then, we have $H_i^\star-\max_i H(\mathcal{S}_i^\star,\tau_i)\le\Delta,\forall i\in\{0,1,\dots,\lceil\frac{\Gamma}{\Delta}\rceil\}$. Note that $H_i^\star$ is the maximum value of $H(\mathcal{S},\tau)$ at each interval $\tau\in[i\Delta,(i+1)\Delta)$. The maximum value of $H(\mathcal{S},\tau),H(\mathcal{S}^\star,\tau^\star)$ is equal to one of $H_i^\star,i\in\{0,1,\dots,\lceil\frac{\Gamma}{\Delta}\rceil\}$. Thus, we reach the claim in Lemma 4. ■

Proof of Lemma 5: We use the previous result [34, Th. 2.3] for the proof of this claim. We know that for any given τ, $H(\mathcal{S},\tau)$ is a nonnormalized monotone submdoular function in $\mathcal{S}$ (see Lemma 1). For maximizing normalized monotone submodular set functions, the greedy approach can give a $1+1/c_f$ approximation of the optimal performance with any matroid constraint [34, Th. 2.3]. After normalizing $H(\mathcal{S},\tau)$ by $H(\mathcal{S},\tau)-H(\emptyset,\tau)$, we have

$$\frac{H(\mathcal{S}_i^G,\tau_i)-H(\emptyset,\tau_i)}{H(\mathcal{S}_i^\star,\tau_i)-H(\emptyset,\tau_i)}\ge\frac{1}{1+c_f} \tag{25}$$

with any matroid constraint. Given $0\le c_f\le 1$ and $H(\emptyset,\tau)=-\tau(\frac{1}{\alpha}-1)$, we transform (25) into

$$H(\mathcal{S}_i^G,\tau_i)\ge\frac{1}{1+c_f}H(\mathcal{S}_i^\star,\tau_i)-\frac{c_f}{1+c_f}\tau_i\left(\frac{1}{\alpha}-1\right). \tag{26}$$

Thus, we prove Lemma 5.

Proof of Theorem 1: Approximation bound: From (12) in Lemma 5, we have that $H(\mathcal{S}_i^\star,\tau_i)$ is bounded by

$$H(\mathcal{S}_i^\star,\tau_i)\le(1+c_f)H(\mathcal{S}_i^G,\tau_i)+c_f\tau_i\left(\frac{1}{\alpha}-1\right). \tag{27}$$

Denote this upper bound as

$$H^b(\mathcal{S}_i^G,\tau_i):=(1+c_f)H(\mathcal{S}_i^G,\tau_i)+c_f\tau_i\left(\frac{1}{\alpha}-1\right).$$

We know $H(\mathcal{S},\tau)$ is below the line starting at $H(\mathcal{S}_i^\star,\tau_i)$ with $slope=1$ during $\tau\in[\tau_i,\tau_{i+1}]$ [the red-dotted line in Fig. 20(a) and (b)] (see Lemma 4). $H(\mathcal{S},\tau)$ must also be below the line starting at $H^b(\mathcal{S}_i^G,\tau_i)$ with $slope=1$ during $\tau\in[\tau_i,\tau_{i+1}]$ [the green-dotted line in Fig. 20(b)]. Similar to the proof in Lemma 4, we have $H_i^\star-H^b(\mathcal{S}_i^G,\tau_i)\le\Delta$ and

$$\max_{i\in\{0,1,\dots,\lceil\frac{\Gamma}{\Delta}\rceil\}}H^b(\mathcal{S}_i^G,\tau_i)\ge H(\mathcal{S}^\star,\tau^\star)-\Delta. \tag{28}$$

SGA selects the pair $(\mathcal{S}^G,\tau^G)$ as the pair $(\mathcal{S}_i^G,\tau_i)$ with $\max_i H(\mathcal{S}_i^G,\tau_i)$. Then, by inequalities (27) and (28), we have

$$(1+c_f)H(\mathcal{S}^G,\tau^G)+c_f\tau^G\left(\frac{1}{\alpha}-1\right)\ge H(\mathcal{S}^\star,\tau^\star)-\Delta. \tag{29}$$

By rearranging the terms, we obtain

$$H(\mathcal{S}^G,\tau^G)\ge\frac{1}{1+c_f}(H(\mathcal{S}^\star,\tau^\star)-\Delta)-\frac{c_f}{1+c_f}\tau^G\left(\frac{1}{\alpha}-1\right). \tag{30}$$

Note that we use an oracle $\mathcal{O}$ to approximate $H(\mathcal{S},\tau)$ with error ϵ by the sampling method (23). By considering the sampling error ϵ, we reach the statement of the approximation bound in Theorem 1.

Computational time: Next, we give the proof of the computational time of SGA in Theorem 1. We verify the computational time of SGA by following the stages of the pseudo code in Algorithm 1. First, from Algorithm 1, lines 2–10, we use a "for" loop for searching τ, which takes $\lceil \frac{\Gamma}{\Delta} \rceil$ evaluations. Second, within the "for" loop, we use the greedy algorithm to solve the subproblem (see Algorithm 1, lines 4–8). In order to select a subset $\mathcal{S}$ with size $|\mathcal{S}|$ from a ground set $\mathcal{X}$ with size $|\mathcal{X}|$, the greedy algorithm takes $|\mathcal{S}|$ rounds (see Algorithm 1, line 5) and calculates the marginal gain of the remaining elements in $\mathcal{X}$ at each round (see Algorithm 1, line 6). Thus, the greedy algorithm takes $\sum_{i=1}^{|\mathcal{S}|} |\mathcal{X}| - i$ evaluations. Third, by calculating the marginal gain for each element, the oracle $\mathcal{O}$ samples n_s times for computing $H(\mathcal{S},\tau)$. Thus, overall, the "for" loop containing the greedy algorithm with the oracle sampling takes $\lceil \frac{\Gamma}{\Delta} \rceil (\sum_{i=1}^{|\mathcal{S}|} |\mathcal{X}| - i) n_s$ evaluations. Finally, finding the best pair from storage set $\mathcal{M}$ (see Algorithm 1, line 11) takes $O(\lceil \frac{\Gamma}{\Delta} \rceil)$ time. Therefore, the computational complexity for SGA is

$$\left\lceil \frac{\Gamma}{\Delta} \right\rceil \left(\sum_{i=1}^{|\mathcal{S}|} |\mathcal{X}| - i \right) n_s + O\left(\left\lceil \frac{\Gamma}{\Delta} \right\rceil \right) = O\left(\left\lceil \frac{\Gamma}{\Delta} \right\rceil |\mathcal{X}|^2 n_s \right)$$

given $|\mathcal{S}| \leq |\mathcal{X}|$.

REFERENCES

[1] J. O'Rourke, *Art Gallery Theorems and Algorithms*, vol. 57. London, U.K.: Oxford Univ. Press, 1987.

[2] A. Krause, A. Singh, and C. Guestrin, "Near-optimal sensor placements in Gaussian processes: Theory, efficient algorithms and empirical studies," *J. Mach. Learn. Res.*, vol. 9, pp. 235–284, 2008.

[3] B. P. Gerkey and M. J. Matarić, "A formal analysis and taxonomy of task allocation in multi-robot systems," *Int. J. Robot. Res.*, vol. 23, no. 9, pp. 939–954, 2004.

[4] J. Vondrák, "Optimal approximation for the submodular welfare problem in the value oracle model," in *Proc. 40th Annu. ACM Symp. Theory Comput.*, 2008, pp. 67–74.

[5] J. R. Spletzer and C. J. Taylor, "Dynamic sensor planning and control for optimally tracking targets," *Int. J. Robot. Res.*, vol. 22, no. 1, pp. 7–20, 2003.

[6] O. Tekdas and V. Isler, "Sensor placement for triangulation-based localization," *IEEE Trans. Autom. Sci. Eng.*, vol. 7, no. 3, pp. 681–685, Jul. 2010.

[7] P. Tokekar, V. Isler, and A. Franchi, "Multi-target visual tracking with aerial robots," in *Proc. IEEE/RSJ Int. Conf. Intell. Robot. Syst.*, 2014, pp. 3067–3072.

[8] L. Zhou, V. Tzoumas, G. J. Pappas, and P. Tokekar, "Resilient active target tracking with multiple robots," *IEEE Robot. Autom. Lett.*, vol. 4, no. 1, pp. 129–136, Jan. 2018.

[9] L. Zhou and P. Tokekar, "Sensor assignment algorithms to improve observability while tracking targets," *IEEE Trans. Robot.*, vol. 35, no. 5, pp. 1206–1219, Oct. 2019.

[10] L. Zhou, V. Tzoumas, G. J. Pappas, and P. Tokekar, "Distributed attack-robust submodular maximization for multi-robot planning," in *Proc. IEEE Int. Conf. Robot. Autom.*. 2020, pp. 2479–2485.

[11] A. Singh, A. Krause, C. Guestrin, and W. J. Kaiser, "Efficient informative sensing using multiple robots," *J. Artif. Intell. Res.*, vol. 34, pp. 707–755, 2009.

[12] G. Shi, L. Zhou, and P. Tokekar, "Robust multiple-path orienteering problem: Securing against adversarial attacks," in *Proc. Robot. Sci. Syst. Conf.*, 2020. [Online]. Available: https://roboticsconference.org/2020/program/papers/95.html

[13] G. L. Nemhauser, L. A. Wolsey, and M. L. Fisher, "An analysis of approximations for maximizing submodular set functions—I," *Math. Program.*, vol. 14, no. 1, pp. 265–294, 1978.

[14] M. L. Fisher, G. L. Nemhauser, and L. A. Wolsey, "An analysis of approximations for maximizing submodular set functions—II," in *Polyhedral Combinatorics*. Berlin, Germany: Springer, 1978, pp. 73–87.

[15] H. Ding and D. Castanón, "Multi-agent discrete search with limited visibility," in *Proc. IEEE 56th Annu. Conf. Decis. Control*, 2017, pp. 108–113.

[16] R. K. Williams, A. Gasparri, and G. Ulivi, "Decentralized matroid optimization for topology constraints in multi-robot allocation problems," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2017, pp. 293–300.

[17] A. D. Wood and J. A. Stankovic, "Denial of service in sensor networks," *Computer*, vol. 35, no. 10, pp. 54–62, 2002.

[18] P. Dames, P. Tokekar, and V. Kumar, "Detecting, localizing, and tracking an unknown number of moving targets using a team of mobile robots," *Int. J. Robot. Res.*, vol. 36, nos. 13/14, pp. 1540–1553, 2017.

[19] D. Kempe, J. Kleinberg, and É. Tardos, "Maximizing the spread of influence through a social network," in *Proc. 9th ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2003, pp. 137–146.

[20] A. Prorok, "Redundant robot assignment on graphs with uncertain edge costs," in *Distributed Autonomous Robotic Systems*. Berlin, Germany: Springer, 2019, pp. 313–327.

[21] G. C. Pflug, "Some remarks on the value-at-risk and the conditional value-at-risk," in *Probabilistic Constrained Optimization*. Berlin, Germany: Springer, 2000, pp. 272–281.

[22] R. T. Rockafellar and S. Uryasev, "Optimization of conditional value-at-risk," *J. Risk*, vol. 2, pp. 21–42, 2000.

[23] R. T. Rockafellar and S. Uryasev, "Conditional value-at-risk for general loss distributions," *J. Bank. Finance*, vol. 26, no. 7, pp. 1443–1471, 2002.

[24] J. Morgan, *Riskmetrics Technical Document*. New York, NY, USA: Morgan Guaranty Trust Company, 1996.

[25] F. Yang and N. Chakraborty, "Algorithm for optimal chance constrained linear assignment," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2017, pp. 801–808.

[26] F. Yang and N. Chakraborty, "Algorithm for optimal chance constrained knapsack problem with applications to multi-robot teaming," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2018, pp. 1043–1049.

[27] H. Markowitz, "Portfolio selection," *J. Finance*, vol. 7, no. 1, pp. 77–91, 1952.

[28] A. Majumdar and M. Pavone, "How should a robot assess risk? Towards an axiomatic theory of risk in robotics," in *Robotics Research*. Berlin, Germany: Springer, 2020, pp. 75–84.

[29] T. Maehara, "Risk averse submodular utility maximization," *Oper. Res. Lett.*, vol. 43, no. 5, pp. 526–529, 2015.

[30] N. Ohsaka and Y. Yoshida, "Portfolio optimization for influence spread," in *Proc. 26th Int. Conf. World Wide Web*, 2017, pp. 977–985.

[31] B. Wilder, "Risk-sensitive submodular optimization," in *Proc. 32nd AAAI Conf. Artif. Intell.*, 2018, pp. 6451–6458.

[32] L. Zhou and P. Tokekar, "An approximation algorithm for risk-averse submodular optimization," in *Proc. Int. Workshop Algorithmic Found. Robot.*, 2018, pp. 144–159.

[33] A. Schrijver, *Combinatorial Optimization: Polyhedra and Efficiency*, vol. 24. Berlin, Germany: Springer, 2003.

[34] M. Conforti and G. Cornuéjols, "Submodular set functions, matroids and the greedy algorithm: Tight worst-case bounds and some generalizations of the Rado-Edmonds theorem," *Discrete Appl. Math.*, vol. 7, no. 3, pp. 251–274, 1984.

[35] P. Artzner, F. Delbaen, J.-M. Eber, and D. Heath, "Coherent measures of risk," *Math. Finance*, vol. 9, no. 3, pp. 203–228, 1999.

[36] E. Ackerman and E. Strickland, "Medical delivery drones take flight in East Africa," *IEEE Spectr.*, vol. 55, no. 1, pp. 34–35, 2018.

[37] *Count Number of Ways to Partition a Set Into k Subsets*, GeeksforGeeks, Sep. 2020. Accessed: Dec. 2, 2020. [Online]. Available: https://www.geeksforgeeks.org/count-number-of-ways-to-partition-a-set-into-k-subsets/

[38] K. J. Obermeyer *et al.*, "VisiLibity: A c++ library for visibility computations in planar polygonal environments," 2008. [Online]. Available: http://www.VisiLibity.org

[39] G. Boeing, "OSMnx: New methods for acquiring, constructing, analyzing, and visualizing complex street networks," *Comput., Environ. Urban Syst.*, vol. 65, pp. 126–139, 2017.

[40] S. Jorgensen, R. H. Chen, M. B. Milam, and M. Pavone, "The team surviving orienteers problem: Routing teams of robots in uncertain environments with survival constraints," *Auton. Robots*, vol. 42, no. 4, pp. 927–952, 2018.

[41] A. Prorok, "Robust assignment using redundant robots on transport networks with uncertain travel time," *IEEE Trans. Autom. Sci. Eng.*, vol. 17, no. 4, pp. 2025–2037, Oct. 2020.
[42] M. Malencia, V. Kumar, G. Pappas, and A. Prorok, "Fair robust assignment using redundancy," *IEEE Robot. Autom. Lett.*, vol. 6, no. 2, pp. 4217–4224, Apr. 2021.
[43] J. J. Chung, A. J. Smith, R. Skeele, and G. A. Hollinger, "Risk-aware graph search with dynamic edge cost discovery," *Int. J. Robot. Res.*, vol. 38, nos. 2/3, pp. 182–195, 2019.

Lifeng Zhou (Member, IEEE) received the bachelor's degree in automation from Huazhong University of Science and Technology, Wuhan, China, in 2013, the master's degree in automation from Shanghai Jiao Tong University, Shanghai, China, in 2016, and the Ph.D. degree in electrical and computer engineering from Virginia Tech, Blacksburg, VA, USA, in 2020.

He is currently a Postdoctoral Researcher with the General Robotics, Automation, Sensing and Perception Laboratory, University of Pennsylvania, Philadelphia, PA, USA. His research interests include multirobot coordination, approximation algorithms, combinatorial optimization, model-predictive control, graph neural networks, and resilient risk-aware decision making.

Pratap Tokekar (Member, IEEE) received the B.Tech. degree in electronics and telecommunication from the College of Engineering Pune, Pune, India, in 2008, and the Ph.D. degree in computer science from the University of Minnesota, Minneapolis, MN, USA, in 2014.

He is currently an Assistant Professor with the Department of Computer Science, University of Maryland, College Park, MD, USA. Previously, he was a Postdoctoral Researcher with the General Robotics, Automation, Sensing and Perception Laboratory, University of Pennsylvania, Philadelphia, PA, USA. From 2015 to 2019, he was an Assistant Professor with the Department of Electrical and Computer Engineering, Virginia Tech, Blacksburg, VA, USA.

Dr. Tokekar is a recipient of the National Science Foundation Directorate for Computer and Information Science and Engineering Research Initiation Initiative Award. He is an Associate Editor for IEEE ROBOTICS AND AUTOMATION LETTERS and IEEE TRANSACTIONS OF AUTOMATION SCIENCE AND ENGINEERING.