# Risk-Aware Submodular Optimization for Multi-Robot Coordination

Lifeng Zhou, *Student 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 CVaR has recently been used in optimization of linear cost functions in robotics, we take the first step towards 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 matroidal 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, 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. In particular, for the mobility-on-demand study, we propose an online triggering assignment algorithm that triggers a new assignment only can potentially lead to reducing the waiting time at demand locations. We verify the performance of the Sequential Greedy Algorithm and the online triggering assignment algorithm through simulations.**

***Index Terms*—risk-aware decision making, conditional value at risk, submodular maximization, sequential greedy algorithm.**

## I. Introduction

Combinatorial optimization problems find a variety of applications in robotics. Typical examples include:

- *Sensor placement:* Where to place sensors to maximally cover the environment [1] or reduce the uncertainty in the environment [2]?
- *Task allocation:* How to allocate tasks to robots to maximize the overall utility gained by the robots [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]–[9], and informative path planning [10]. The underlying optimization problem in most cases can be written as:

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

L. Zhou is with the Department of Electrical and Computer Engineering, Virginia Tech, Blacksburg, USA. {lfzhou}@vt.edu.

P. Tokekar is with the Department of Computer Science, University of Maryland, College Park, USA. {tokekar}@umd.edu.

This material is based upon work supported by the NSF under grant number IIS-1637915 and ONR under grant number N00014-18-1-2829.

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 [11], [12]. Submodularity is the property of diminishing returns. Many information theoretic measures, such as mutual information [2], and geometric measures such as the visible area [13], are known to be submodular. $\mathcal{I}$ denotes a matroidal constraint [11], [12]. 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") [14]. The objective of this problem is to find a set $\mathcal{S}$ satisfying a matroidal 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 [11], [12].

In practice, sensors can fail or get compromised [15] or robots may not know the exact positions of the targets [16]. 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 Equation. 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 which 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 $\mathcal{S}$. Thus, the greedy algorithm still retains its constant-factor performance guarantee [11], [12]. Examples of this approach include influence maximization [17], moving target detection and tracking [16], and robot assignment with travel-time uncertainty [18].

While optimizing the expected utility has its uses, it also has its pitfalls too. Consider the example of mobility-on-demand where two vehicles, the red vehicle and the blue vehicle, are available to pick up the passengers at a demand location (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

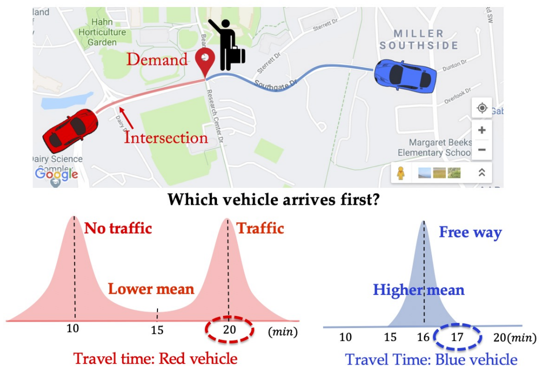


Fig. 1: Mobility on demand with travel time uncertainty of self-driving vehicles.

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., $17 \sim 20$ $min$ about half of the times. A more risk-aware passenger would choose the blue vehicle which has higher expected waiting time 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 *Conditional-Value-at-Risk* (CVaR) [19], [20]. CVaR is parameterized by a risk level $\alpha$. Informally, maximizing CVaR is equivalent to maximizing the expectated utility in the worst $\alpha$-tail scenarios.[1]

**Related work**. Several works have focused on optimizing CVaR. In their seminal work [20], Rockafellar and Uryasev 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 CVaR for general loss distributions in finance [21].

Another popular risk measure in finance is the Value-at-Risk (VaR) [22], which is generally used for formulating the chance-constrained optimization problems. Yang and Chakraborty studied a chance-constrained combinatorial optimization problem that takes into account the risk in multi-robot assignment [23]. They later extended this to knapsack problems [24]. They solved the problem by transforming it to a risk-aware problem with mean-variance measure [25]. However, Majumdar and Pavone argued that CVaR is a better measure to quantify risk than VaR or mean-variance based on six proposed axioms in the context of robotics [26].

When the utility is a discrete submodular set function, i.e., $f(\mathcal{S}, y)$, Maehara presented a negative result for maximizing CVaR [27]— 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 in [28] 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 considered a CVaR maximization of a continuous submodular function instead of the submodular set functions [29]. They gave a $(1-1/e)$–approximation algorithm for continuous submodular functions and also evaluated the algorithm for discrete submodular functions using portfolio optimization [28].

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

**Contributions**. In this paper, we first focus on the problem of selecting a single set, similar to [27], to optimize CVaR of a stochastic submodular set function. We propose an approximation algorithm with provable theoretical guarantees. Then, we present an online variant of this algorithm, specifically for the mobility-on-demand problem. We study how to trigger replanning by solving the CVaR optimization problem to avoid unnecessary vehicle assignments.

Our contributions are as follows:

- We propose the Sequential Greedy Algorithm (SGA) which uses the deterministic greedy algorithm [11], [12] as a subroutine to find the maximum value of CVaR (Algorithm 1).
- We prove that the solution found by SGA is within a constant factor of the optimal performance along with an additive term which depends on the optimal value and sampling error. We also prove that SGA runs in polynomial time (Theorem 1) and the performance improves as the running time increases.
- We propose the Online Triggering Assignment (OTA) for the mobility-on-demand application (Algorithm 2) which triggers SGA only when certain conditions are satisfied.
- We demonstrate the utility of the proposed CVaR maximization problem through two case studies (Section III-B). We evaluate the performance of SGA and OTA through simulations (Section VI).

**Organization of rest of the paper**. We give the necessary background knowledge for the rest of the paper 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 OTA for the mobility-on-demand problem in Section V. We illustrate the performance of SGA to the two case studies and evaluate the performance of OTA with street networks in Section VI. We conclude the paper in Section VII.

A preliminary version of this paper was first presented in [30] without the analysis of the approximation error induced by sampling method (see Section IV) and OTA (see Section V) with its numerical evaluations (see Section VI-C).

## II. Background and Preliminaries

We start by defining the conventions used in the paper.

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 $\mathcal{B}$, $\mathcal{A}\backslash\mathcal{B}$ 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.

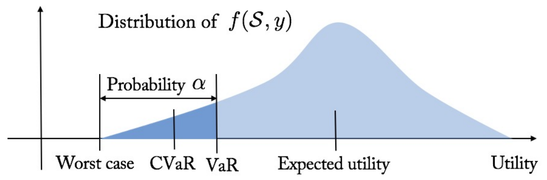


Fig. 2: An illustration of risk measures: VaR and CVaR.

### 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.

**Monotonicity [11]:** A set function $f : 2^{\mathcal{X}} \mapsto \mathbb{R}$ is monotone (non-decreasing) if and only if for any sets $\mathcal{S} \subseteq \mathcal{S}' \subseteq \mathcal{X}$, we have $f(\mathcal{S}) \leq f(\mathcal{S}')$.

**Normalized Function [12]:** A set function $f : 2^{\mathcal{X}} \mapsto \mathbb{R}$ is called normalized if and only if $f(\emptyset) = 0$.

**Submodularity [11, 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 non-increasing.

**Matroid [31, Section 39.1]:** Denote a non-empty 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:

- 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 $\mathcal{P} \in \mathcal{I}$.
- for any sets $\mathcal{S}, \mathcal{P} \subseteq \mathcal{X}$ 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.

**Uniform Matroid:** A *uniform matroid* is a matroid $(\mathcal{X}, \mathcal{I})$ such that for a positive integer $\kappa$, $\{\mathcal{S} : \mathcal{S} \subseteq \mathcal{X}, |\mathcal{S}| \leq \kappa\}$. Thus, the uniform matroid only constrains the cardinality of the feasible sets in $\mathcal{I}$.

**Partition Matroid:** A *partition matroid* is a matroid $(\mathcal{X}, \mathcal{I})$ such that for a positive integer $n$, disjoint sets $\mathcal{X}_1, ..., \mathcal{X}_n$ and positive integers $\kappa_1, ..., \kappa_n$, $\mathcal{X} \equiv \mathcal{X}_1 \cup \cdots \mathcal{X}_n$ and $\mathcal{I} = \{\mathcal{S} : \mathcal{S} \subseteq \mathcal{X}, |\mathcal{S} \cap \mathcal{X}_i| \leq \kappa_i$ for all $i = 1, ..., n\}$.

**Curvature [32]:** Consider a matroid $\mathcal{I}$ for $\mathcal{X}$, and a non-decreasing 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:

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

Note that the definition of curvature $k_f$ (Eq. 3) implies that $0 \leq k_f \leq 1$. Specifically, if $k_f = 0$, 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 $k_f = 1$, 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, that is,

$$s = \underset{s \in \mathcal{X}}{\text{argmax}}\ f(\mathcal{P} \cup \{s\}) - f(\mathcal{P})$$

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 [12]. In particular, the greedy algorithm can give a $(1 - 1/e)$–approximation of the optimal solution under the uniform matroid [11]. If we know the curvature of the set function $f$, we have a $1/(1 + k_f)$ approximation for any matroid constraint [32, Theorem 2.3]. That is,

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

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 $k_f = 0$, which means $f$ is modular, then the greedy algorithm reaches the optimal. If $k_f = 1$, then we have the $1/2$–approximation.

### C. *Risk measures*

Let $f(\mathcal{S}, y)$ be a utility function with 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 Value-at-Risk at risk level $\alpha \in (0, 1]$.

**Value at Risk:**

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

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

**Conditional Value at Risk:**

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

Figure 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 [20], such as *coherence* [33].

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

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

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 $\tau$ [20]. 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 $\tau$.

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

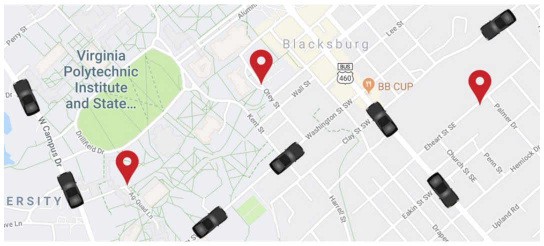


Fig. 3: Mobility-on-demand with multiple demands and multiple self-driving vehicles.

*Lemma 2:* The auxiliary function $H(\mathcal{S},\tau)$ is concave in $\tau$ 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 $\tau$ 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

**CVaR Submodular Maximization**: 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 $\tau$ [20]. Thus, we propose the maximization problem as:

*Problem 1:*

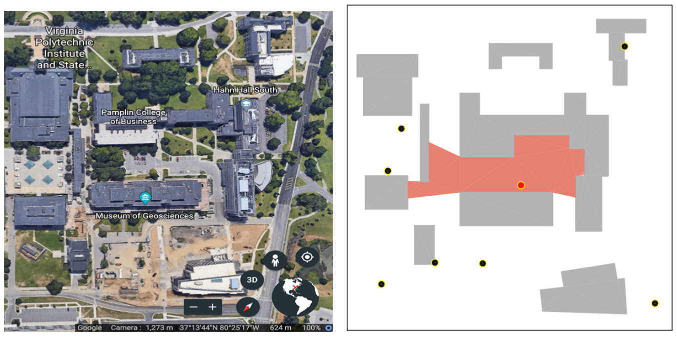
$$\begin{aligned}&max\quad \tau-\frac{1}{\alpha}\mathbb{E}[(\tau-f(\mathcal{S},y))_+]\\ &s.t.\quad \mathcal{S}\in\mathcal{I},\mathcal{S}\subseteq\mathcal{X},\tau\in[0,\Gamma],\end{aligned}\tag{6}$$

where $\Gamma$ is the upper bound of the parameter $\tau$. Problem 1 gives a risk-aware version of maximizing submodular set functions.

### B. Case Studies

The risk-aware submodular maximization has many applications, as it has been written in Section III-B. 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 Figure 3 where seven self-driving vehicles must be assigned to three demand locations to pick up passengers. We follow the same constraint setting in [18]— 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 [18]. This may be too conservative for consumer



(a) Part of Virginia Tech campus from Google Earth. (b) Top view of part of a campus and ground sensor's visibility region.

Fig. 4: Campus monitoring by using a set of sensors with visibility regions.

mobility-on-demand services but can be crucial for urgent and time-critical tasks such as delivering medical supplies [34].

Assume the arrival time for the vehicle to arrive at a demand location is a random variable. The distribution 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 *efficiency* 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 efficiency. The arrival efficiency is also a random variable, and has a distribution depending on mean-arrival efficiency. Denote the arrival efficiency for vehicle $j\in\{1,...,R\}$ arriving at demand location $i\in\{1,...,N\}$ as $e_{ij}$. Denote the assignment utility as the arrival efficiency at all locations, that is,

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

with $\bigcup_{i=1}^N\mathcal{S}_i=\mathcal{S}$ and $\mathcal{S}_i\cap\mathcal{S}_k=\emptyset,\ i,k\in\{1,\cdots,N\}$. $\mathcal{S}_i\cap\mathcal{S}_k=\emptyset$ indicates 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 trade-off between efficiency and uncertainty. Our goal is to maximize the total efficiencies 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 (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 (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 are 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 $i, i \in \{1, ..., N\}$ 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, $\mathcal{S}$, that is,

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

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 trade-off 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 the Sequential Greedy Algorithm (SGA) for solving Problem 1 by leveraging the useful properties of the auxiliary function $H(\mathcal{S}, \tau)$. The pseudo-code is given in Algorithm 1. SGA mainly consists of searching for the appropriate value of $\tau$ by solving a subproblem for a fixed $\tau$ under a matroid constraint. Even for a fixed $\tau$, 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 $\tau$. We explain each stage in detail next.

### A. Sequential Greedy Algorithm

These are four stages in SGA:

*a) Initialization (Alg. 1, line 1):* Algorithm 1 defines a storage set $\mathcal{M}$ and initializes it to be the empty set. Note that, for each specific $\tau$, 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 $\tau$.

*b) Searching for $\tau$ (**for** loop in Alg. 1, lines 2–10):* We use a user-defined separation $\Delta$ (Alg. 1, line 3) to sequentially search for all possible values of $\tau$ within $[0, \Gamma]$. $\Gamma$ is an upper bound on $\tau$ and can be set by the user based on the specific problem at hand. We show how to find $\Gamma$ for the specific cases in Section VI.

*c) Greedy algorithm (Alg. 1, lines 4–8):* For a specific $\tau$, say $\tau_i$, 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 (Alg. 1, line 4). Under a matroid constraint, $\mathcal{S}_i^G \in \mathcal{I}$ (Alg. 1, line 5), we add a new element $s$ which gives the maximum marginal gain of $\hat{H}(\mathcal{S}_i^G, \tau_i)$ (Alg. 1, line 6) into set $\mathcal{S}_i^G$ (Alg. 1, line 7) in each round.

**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
          $\tau^G$
 1: $\mathcal{M} \leftarrow \emptyset$
 2: for $i = \{0, 1, \cdots, \lceil \frac{\Gamma}{\Delta} \rceil\}$ do
 3:     $\tau_i = i\Delta$
 4:     $\mathcal{S}_i^G \leftarrow \emptyset$
 5:     while $\mathcal{S}_i^G \in \mathcal{I}$ do
 6:         $s = \underset{s \in \mathcal{X} \setminus \mathcal{S}_i^G, \mathcal{S}_i^G \cup \{s\} \in \mathcal{I}}{\operatorname{argmax}} \hat{H}((\mathcal{S}_i^G \cup \{s\}), \tau_i) - \hat{H}(\mathcal{S}_i^G, \tau_i)$
 7:         $\mathcal{S}_i^G \leftarrow \mathcal{S}_i^G \cup \{s\}$
 8:     end while
 9:     $\mathcal{M} = \mathcal{M} \cup \{(\mathcal{S}_i^G, \tau_i)\}$
10: end for
11: $(\mathcal{S}^G, \tau^G) = \underset{(\mathcal{S}_i^G, \tau_i) \in \mathcal{M}}{\operatorname{argmax}} \hat{H}(\mathcal{S}_i^G, \tau_i)$
```

*d) Find the best pair (Alg. 1, line 11):* Based on the collection of pairs $\mathcal{M}$ (Alg. 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 $\mathcal{S}_i^G$. We denote this value of $\tau$ by $\tau^G$.

*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 [28, Lemmas 4.1-4.5], if the number of samples is $n_s = O(\frac{\Gamma^2}{\epsilon^2} \log \frac{1}{\delta})$, $\delta, \epsilon \in (0,1)$, the CVaR approximation error is less than $\epsilon$ with the probability at least $1 - \delta$. We provide a brief proof for the number of samples in the appendix.

### B. Performance Analysis of SGA

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

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

$$\begin{aligned} H(\mathcal{S}^G, \tau^G) \geq & \frac{1}{1+k_f}(H(\mathcal{S}^\star, \tau^\star) - \Delta) \\ & - \frac{k_f}{1+k_f}\Gamma(\frac{1}{\alpha} - 1) - \epsilon, \end{aligned} \tag{9}$$

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 $\Gamma$ and $\Delta$ are the upper bound on $\tau$ and searching separation parameter, $|\mathcal{X}|$ is the cardinality of the ground set $\mathcal{X}$.

SGA gives $1/(1+k_f)$ approximation of the optimal with three additional approximation errors. One approximation error comes from the searching separation $\Delta$. We can make this error arbitrarily small by setting $\Delta$ 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 $\epsilon$ from the oracle. Evidently, $\epsilon$ can be reduced by increasing the number of samples $n_s$. The third one comes from the additive term,

$$H^{\text{add}} = \frac{k_f}{1+k_f}\Gamma(\frac{1}{\alpha} - 1), \tag{10}$$

which depends on the curvature $k_f$ and the risk level $\alpha$. When the risk level $\alpha$ is very small, this error is very large which means SGA may not give a good performance guarantee of the optimal. However, if the function $H(\mathcal{S}, \tau)$ is close to modular in $\mathcal{S}$ ($k_f \to 0$), this error is close to zero. Notably, when $k_f \to 0$ 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, 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 $\tau_i$ that maximizes $H(\mathcal{S}, \tau = \tau_i)$ and $H(\mathcal{S}^\star, \tau^\star)$ be the maximum value of $H(\mathcal{S}, \tau)$. We sequentially search for $\tau \in [0, \Gamma]$ with a searching separation $\Delta$. We first describe the relationship between the maximum $H(\mathcal{S}, \tau)$ founded by searching $\tau$ with a searching separation $\Delta$ and $H(\mathcal{S}^\star, \tau^\star)$.

*Lemma 4:*

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

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 $\tau_i$.

*Lemma 5:*

$$H(\mathcal{S}_i^G, \tau_i) \geq \frac{1}{1+k_f} H(\mathcal{S}_i^\star, \tau_i) - \frac{k_f}{1+k_f}\Gamma(\frac{1}{\alpha} - 1). \tag{12}$$

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

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

## V. Triggering Vehicle Assignment

In this section, we study an online version of the resilient mobility-on-demand described in Section III-B. Although this strategy is specific to the mobility-on-demand case study, the general idea can be used for other applications.

Different from the offline assignment where the assignment is done at the initial time step, we allow the assignment to be changeable as vehicles travel towards the previously assigned demand locations. The advantage of online assignment is that vehicles can use real-time information to update the assignment for achieving shorter arrival time (i.e., higher efficiency) at demand locations as the actual travel time information is revealed.

One intuitive online strategy is to schedule assignments for vehicles at all time steps by using Algorithm 1. Clearly, the *all-step* assignment can achieve a better performance than the offline assignment, since real-time traffic conditions and vehicles' positions can be used to update the assignment at all time steps. However, scheduling assignment at all time steps can be energy-costly and, sometimes, are not even possible, since computing assignment takes time when the number of vehicles and/or the number of demand locations is large. Also, scheduling assignments at all time steps may not be necessary. Thus, we seek an assignment strategy that performs comparably with the all-step assignment, while avoiding unnecessary assignments. We design such an assignment strategy by exploiting an event-driven mechanism where the assignment is updated or recomputed only when certain conditions are met. We call this assignment strategy as the Online Triggering Assignment (OTA). We introduce a general version of OTA in this section (Alg. 2). Please see a specific implementation of OTA for mobility-on-demand in street networks in the appendix (Alg. 3).

Similar to the resilient mobility-on-demand setting (Section III-B), we have $R$ vehicles whose positions are denoted by, $p_j,\ j \in \{1, \cdots, R\}$, to be assigned to $N$ demand locations, $d_i,\ i \in \{1, \ldots, N\}$. We denote the stochastic arrival time for each vehicle $j$ to each demand $i$ as $t_{ij}$ with mean $\bar{t}_{ij}$ and variance $\texttt{var}(t_{ij})$. Clearly, the arrival efficiency $e_{ij} = 1/t_{ij}$. Note that, the arrival time $t_{ij}$ can be updated based on real-time traffic conditions and vehicles' positions. Similarly, for each demand $i$, the set of vehicles assigned to it is denoted by $\mathcal{S}_i$.

OTA (Alg. 2) mainly contains two assignment steps, initial assignment (Alg. 2, line 1) and triggering assignment (Alg. 2, line 7). OTA terminates when all demand locations are reached by vehicles (Alg. 2, line 2). After the initial assignment (Alg. 2, line 1), each vehicle $i$ knows its first assigned demand $j$. Then each vehicle $i$ travels towards demand $j$ (Alg. 2, line 3) and updates its arrival time $t_{ij}$ and corresponding arrival efficiency $e_{ij}$ at each time step (Alg. 2, line 4).

The reassignment is triggered by checking the arrival time of vehicles assigned to each demand location $d_i$ (Alg. 2, line 5). Specifically, among the set of vehicles $\mathcal{S}_i$ assigned to demand $i$, if the arrival time of one vehicle $j \in \mathcal{S}_i$ is less than that of the other vehicle $j' \in \mathcal{S}_i$ on average and has smaller uncertainty (Alg. 2, line 6), a new assignment is triggered (Alg. 2, line 7). In this case, there is no need to continue assigning vehicle $j'$ to demand $i$. 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, i.e., a larger $\gamma$ triggers more assignments. In the next section, we evaluate the effect of $\gamma$ in terms of the arrival time when all damned locations are reached and the number of assignments.

## VI. Simulations

We perform numerical simulations to verify the performance of SGA in risk-aware mobility-on-demand and robust envi-

**Algorithm 2:** Online Triggering Assignment

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

ronment monitoring and the performance of OTA in online mobility-on-demand. Our code is available online.[2]

### *A. Resilient Mobility-on-Demand under Arrival Time Uncertainty*

We consider assigning $R = 6$ supply vehicles to $N = 4$ demand locations in a 2D environment. The positions of the demand locations and the supply vehicles are randomly generated within a square environment of 10-unit side length. Denote the Euclidean distance between demand location $i \in \{1, ..., N\}$ and vehicle position $j \in \{1, ..., R\}$ as $d_{ij}$. Based on the distribution discussion of the arrival efficiency distribution in Section III-B, we assume each arrival efficiency $e_{ij}$ 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 efficiency 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 arrival efficiency 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 $\max\{\bar{e}_{ij}\} = \max_{i,j} e_{ij}, i \in \{1, ..., N\}, j \in \{1, ..., R\}$.

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

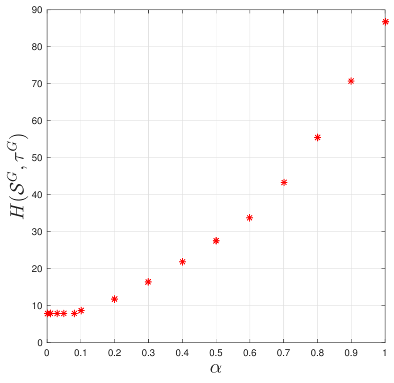
$$f(\mathcal{S}, \tilde{y}) := \sum_{i \in N} \max_{j \in \mathcal{S}_i} \tilde{e}_{ij}$$

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

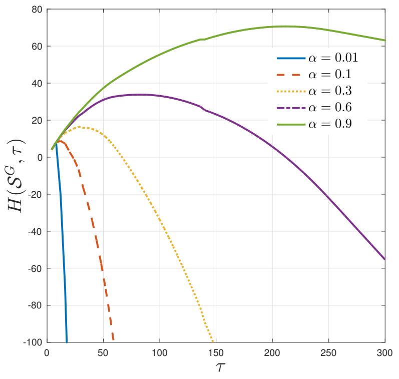
$$\hat{H}(\mathcal{S}, \tau) = \tau - \frac{1}{n_s \alpha} \sum_{\tilde{y}} [(\tau - \sum_{i \in N} \max_{j \in \mathcal{S}_i} \tilde{e}_{ij})_+].$$

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

[2]https://github.com/raaslab/risk_averse_submodular_selection



(a) The value of $H(\mathcal{S}^G, \tau^G)$ with respect to several risk levels $\alpha$.



(b) Function $H(\mathcal{S}^G, \tau)$ with respect to several risk levels $\alpha$.

Fig. 5: The value of $H(\mathcal{S}, \tau)$ by SGA with respect to different risk levels.

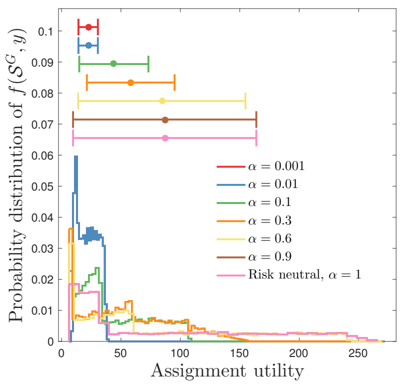


Fig. 6: Distribution of the assignment utility $f(\mathcal{S}^G, y)$ by SGA.

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 $\alpha$ in Figure 5. Figure 5-(a) shows that $H(\mathcal{S}^G, \tau^G)$ increases when $\alpha$ increases. This suggests that SGA correctly maximizes $H(\mathcal{S}, \tau)$. Figure 5-(b) shows that $H(\mathcal{S}^G, \tau)$ is concave or piecewise concave, which is consistent with the property of $H(\mathcal{S}, \tau)$.

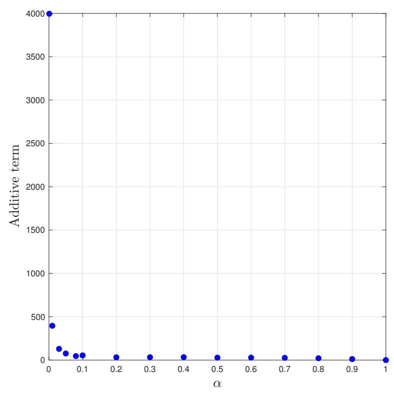


Fig. 7: Additive term (Eq. 10) in the approximation ratio with respect to risk level $\alpha$.

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 Figure 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 $i$ by SGA. $\mathcal{S}^G = \bigcup_{i=1}^N \mathcal{S}_i^G$. When the risk level $\alpha$ is small, vehicle-demand pairs with low efficiencies (equivalently, low uncertainties) are selected. This is because a small risk level indicates the assignment is conservative and only willing to take a little risk. Thus, lower efficiency with lower uncertainty is assigned to avoid the risk induced by the uncertainty. In contrast, when $\alpha$ is large, the assignment is allowed to take more risk to gain more assignment utility. Vehicle-demand pairs with high efficiencies (equivalently, high uncertainties) are selected in such a case. Note that, when the risk level is close to zero, SGA may not give a correct solution because of a large approximation error (Fig. 7). However, this error decreases quickly to zero when the risk level increases.

We also compare SGA (using CVaR measure) with the greedy algorithm (using the expectation, i.e., risk-neutral measure [18]) in Figure 8. Note that risk-neutral measure is a special case of CVaR$_\alpha$ measure when $\alpha = 1$. We give an illustrative example of the assignment by SGA for two extreme risk levels, $\alpha = 0.1$ and $\alpha = 1$. When $\alpha$ is small ($\alpha = 0.1$), the assignment is conservative and thus further vehicles (with lower efficiency and lower uncertainty) are assigned to each demand (Fig. 8-(a)). In contrast, when $\alpha = 1$, nearby vehicles (with higher efficiency and higher uncertainty) are selected for the demands (Fig. 8-(b)). Even though the mean value of the assignment utility distribution is larger at $\alpha = 1$ than at $\alpha = 0.1$, 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 $\alpha = 0.1$ (Fig. 8-(c)).

## B. Robust Environment Monitoring

We consider selecting $M = 4$ locations from $N = 8$ candidate locations to place sensors in the environment (Fig. 4). Denote the area of the free space as $v^{\text{free}}$. The positions of $N$ candidate locations are randomly generated within the free space $v^{\text{free}}$. We calculate the visibility region for each sensor $v_i$ by using the VisiLibity library [35]. 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

$$\begin{cases} \Pr[A_i = v_i] = p_i, \\ \Pr[A_i = 0] = 1 - p_i. \end{cases} \tag{13}$$

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

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

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

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

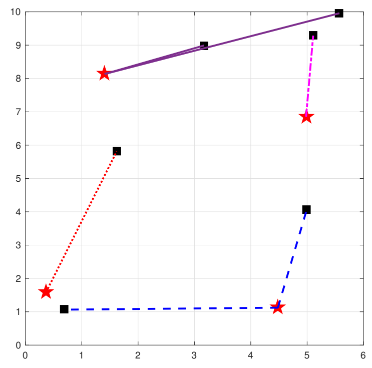
We set the upper bound for the parameter $\tau$ as the area of all the free space $v^{\text{free}}$ and set the searching separation for $\tau$ as $\Delta = 1$.

We use SGA to find the pair $(\mathcal{S}^G, \tau^G)$ with respect to several risk levels $\alpha$. We plot the value of $H(\mathcal{S}^G, \tau^G)$ for several risk levels in Figure 9-(a). A larger risk level gives a larger $H(\mathcal{S}^G, \tau^G)$, which means the pair $(\mathcal{S}^G, \tau^G)$ found by SGA correctly maximizes $H(\mathcal{S}, \tau)$ with respect to the risk level $\alpha$. Moreover, we plot functions $H(\mathcal{S}^G, \tau)$ for several risk levels $\alpha$ in Figure 9-(b). Note that $\mathcal{S}^G$ is computed by SGA at each $\tau$. For each $\alpha$, $H(\mathcal{S}^G, \tau)$ shows the concavity or piecewise concavity of function $H(\mathcal{S}, \tau)$.

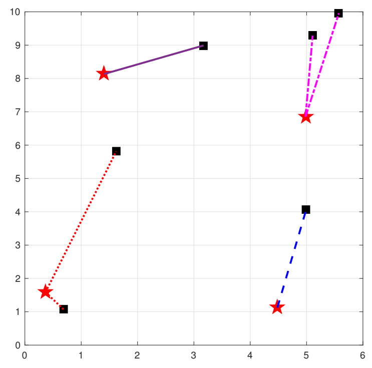
Based on the $\mathcal{S}^G$ calculated by SGA, we sample $n_s = 1000$ times from the underlying distribution of $y$ and plot the distribution of the selection utility,

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

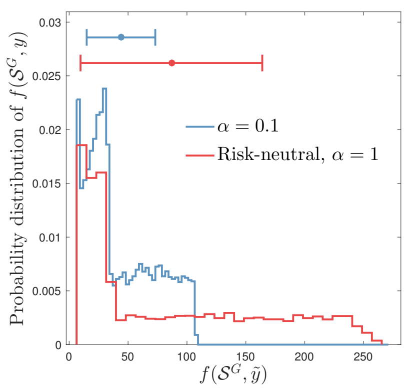
in Figure 10. Note that, when the risk level $\alpha$ is small, the sensors with smaller visibility region and a higher probability of success should be selected. Lower risk level suggests a conservative selection. Sensors with a higher probability of success are selected to avoid the risk induced by the sensor failure. In contrast, when $\alpha$ 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. Figure 10 demonstrates this behavior except between $\alpha = 0.001$ to $\alpha = 0.01$. This is because when $\alpha$ is very small, the approximation error (Eq. 10) is very large as shown in Figure 11, and thus SGA may not give a good solution.



(a) Assignment when $\alpha = 0.1$.



(b) Assignment when $\alpha = 1$ (Risk-neutral)).



(c) Assignment utility distributions at $\alpha = 0.1$ and $\alpha = 1$.

Fig. 8: 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.

We also compare SGA by using CVaR measure with the greedy algorithm by using the expectation, i.e., risk-neutral measure (mentioned in [2, Section 6.1]) in Figure 12. In fact, the risk-neutral measure is equivalent to the case of $\text{CVaR}_\alpha$ when $\alpha = 1$. We give an illustrative example of the sensor selection by SGA for two extreme risk levels, $\alpha = 0.1$ and $\alpha = 1$. When risk level $\alpha$ is small ($\alpha = 0.1$), the selection is conservative and thus the sensors with small visibility region are selected (Fig. 12-(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 (Fig. 12-(b)). The mean-std bars of the selection utility distributions in Figure 12-(c) show that the selection utility at the expectation ($\alpha = 1$) has larger mean value than the selection at $\alpha = 0.1$. 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 $\alpha = 0.1$.

## C. Online Resilient Mobility-on-Demand

In this section, we show the effectiveness of OTA (Alg. 2, 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 OTA with the other two assignment strategies, offline assignment and all-step assignment, in terms of the arrival time, the number of assignments, and the running time. Particularly, we compute the arrival time as the summation of all per-step time intervals ($T^{\texttt{step}}$ in Eq. 18) from time of the first step to the final time step when all demand locations are reached. The difference of these three assignments comes from how often the assignment is scheduled— In the offline 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 OTA, the assignment is triggered only at specific time steps. In both offline assignment and all-step assignment, all vehicles apply the "per-step online travel" rule (Alg. 3, lines 16-25) to go towards the assigned demand locations, as in the case of OTA.

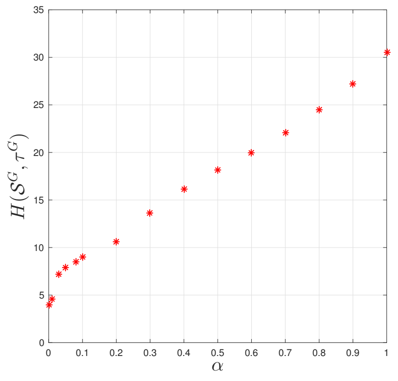
We consider assigning $R$ supply vehicles to $N$ demand locations on a street network, $\mathcal{G}$, created using OSMnx [36]. OSMnx is a python package that allows for easily constructing, projecting, visualizing, and analyzing complex street networks [36]. We give an example showing a network of drivable public streets created by OSMnx in Blacksburg, VA, USA in Figure 13. OSMnx also allows for finding a shortest path between two nodes (intersections) of the networks by Dijkstra's algorithm if there exists one (Fig. 13). Thus, we use OSMnx to compute a shortest path from each vehicle's initial position to 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,

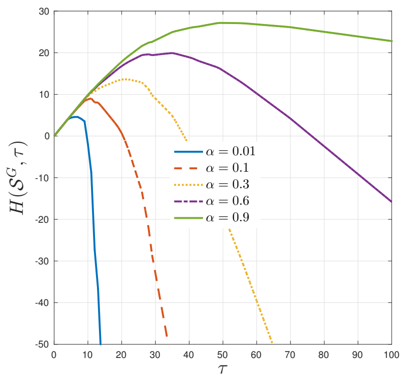
$$t(v_i) \sim \mathcal{N}^{\texttt{Truc}}(0, \beta_1 \texttt{deg}(v_i)), \quad t(v_i) \in [0, T(v_i)] \qquad (14)$$

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 distribution (Eq. 14). In practice, the real-time



(a) The value of $H(\mathcal{S}^G, \tau^G)$ with respect to different risk levels $\alpha$.



(b) Function $H(\mathcal{S}^G, \tau)$ with respect to several risk levels $\alpha$.

Fig. 9: The value of $H(\mathcal{S}, \tau)$ by SGA with respect to different risk levels.

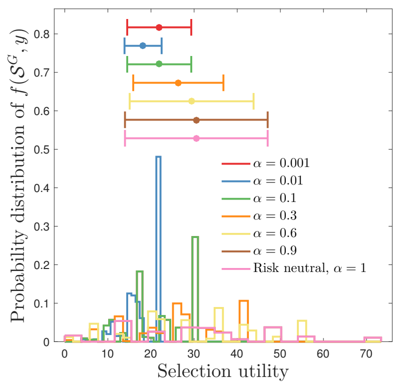


Fig. 10: Distribution of the selection utility $f(\mathcal{S}^G, y)$ by SGA.

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 Eq. 15). If a demand location is not reachable from a vehicle's position, we set the path travel time (Eq. 16) from the vehicle to the demand as infinity. We set the risk level $\alpha = 0.1$ in SGA (Alg. 1).

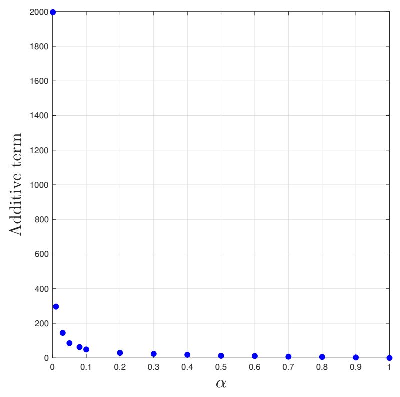
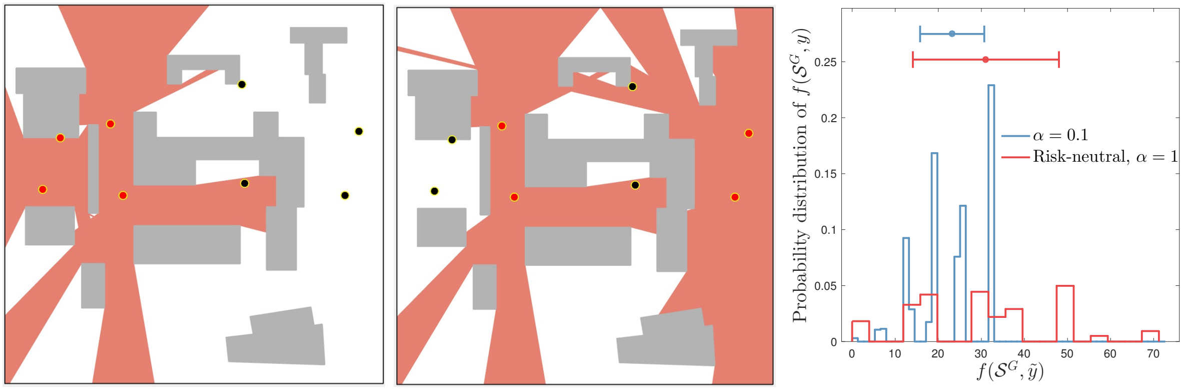


Fig. 11: Additive term (Eq. 10) in the approximation ratio with respect to risk level $\alpha$.

*1) Qualitative Results:* We first show one trial of OTA in action in Figure 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, OTA can end up with different results in different trials, even with the same vehicles' initial positions and demand locations.

In Figure 16, OTA starts at time step 1 (Fig. 16-(a)) and ends at time step 34 (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 10 time steps (steps 1, 2, 19, 22, 23, 24, 25, 26, 27, 28) as shown in Figure 14. At these 10 steps, the triggering conditions with $\gamma = 0.5$ (Eqs. 19 and 20) are satisfied. For example, at step 1 (Fig. 16-(a)), the path length from $p_3$ to $d_2$ is less than half of that from $p_2$ to $d_2$ (Eq. 19) and the path degree from $p_3$ to $d_2$ is less than that from $p_2$ to $d_2$ (Eq. 20). 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, Fig. 16-(b)). Similar assignment flip happens from step 22 (Fig. 16-(e)) to step 23 (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 (Fig. 14), the assignment remains the same at step 20 (Fig. 16-(c) & (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 $d_2$ (Fig. 16-(c) & (d)). OTA also guarantees that once a demand is reached, e.g., in step 28, $d_2$ is reached by vehicle 3 (Fig. 16-(g)), no vehicle will be assigned to it later (Fig. 16-(h), (i), (j), (k), (l)). Particularly, in this 3-vehicle 2-demand system, there is not even reassignment triggered afterwards (Fig. 14). In Figure 16-(i) & (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 Figure 17. We execute all three assignment strategies in 10 trials with the same initial vehicles' positions



(a) Selection when $\alpha = 0.1$. (b) Selection when $\alpha = 1$ (Risk-neutral). (c) Selection utility distributions at $\alpha = 0.1$ and $\alpha = 1$.

Fig. 12: Sensor selection and utility distributions by SGA with two extreme risk level values. The red solid circle represents the sensor selected by SGA.

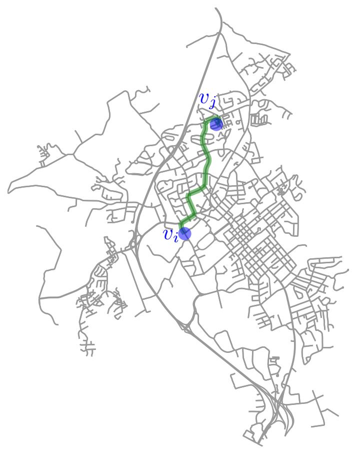
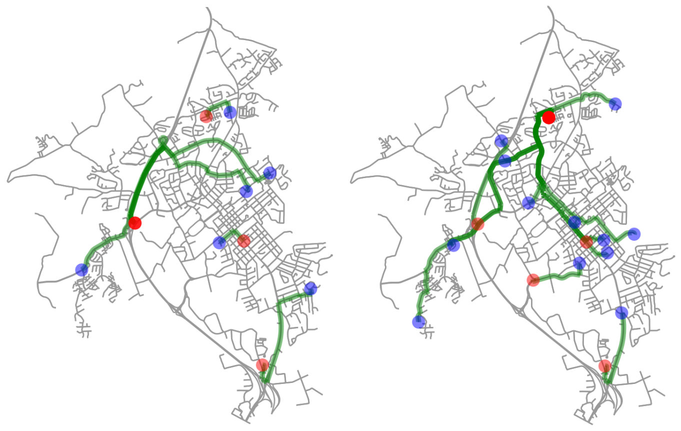


Fig. 13: Street network created by OSMnx in Blacksburg, VA, USA with a shortest path (green path) from node $v_i$ to node $v_j$ (lighter blue dots).



(a) 6-vehicle 4-demand system (b) 12-vehicle 5-demand system

Fig. 15: The initial assignment in (a), 6-vehicle 4-demand system and (b), 12-vehicle 5-demand system.

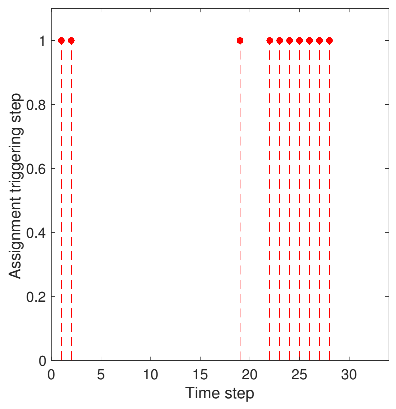
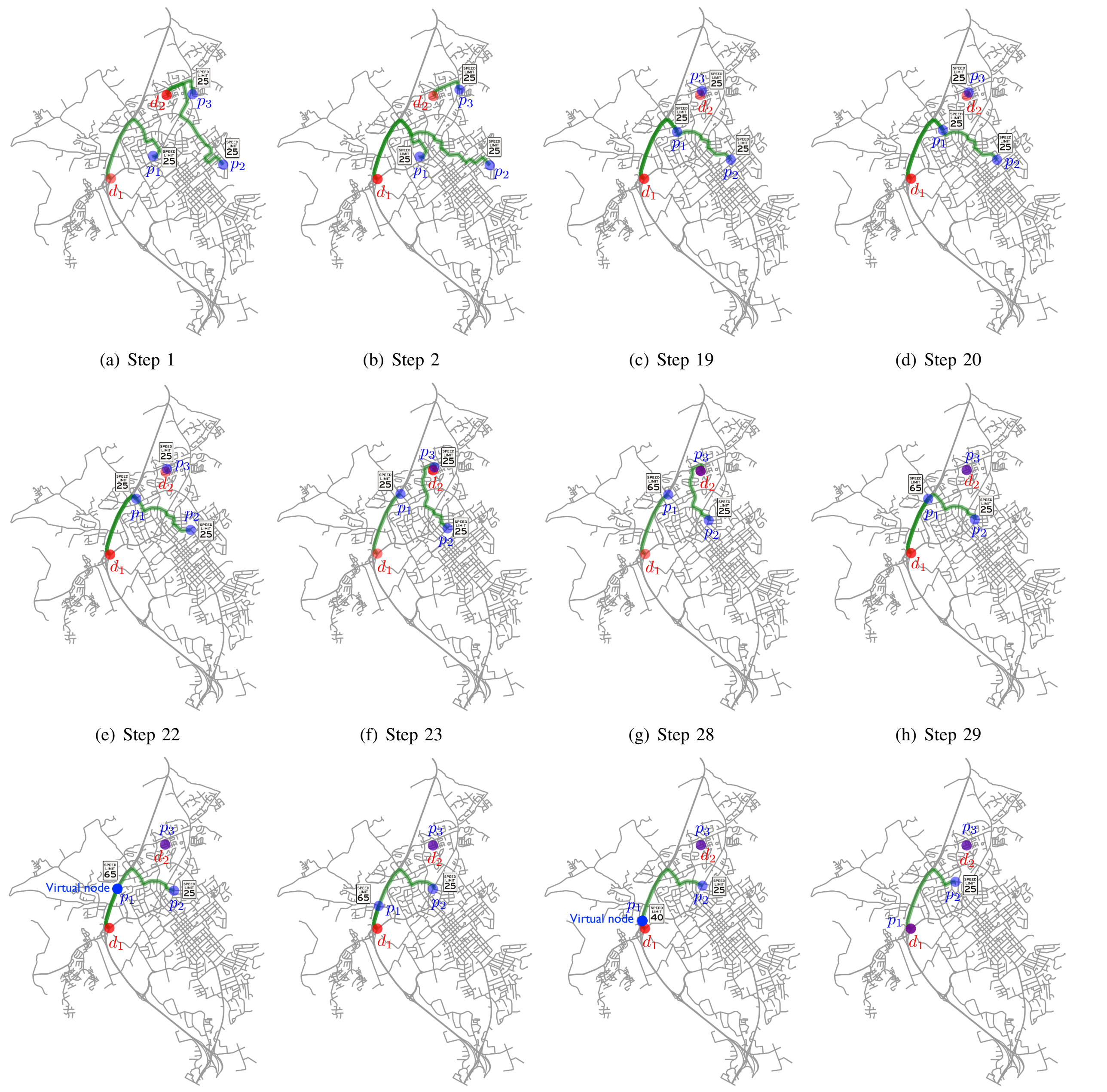


Fig. 14: The triggering time steps in one trial of OTA (Alg. 3) in action.

and demand locations. In particular, we consider three scales of vehicle-demand system, e.g., $R = 3, N = 2$ (Fig. 16), $R = 6, N = 4$ (Fig. 15-(a)) and $R = 12, N = 5$ (Fig. 15-(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.

Figure 17-(a), (b) & (c) show that OTA and the all-step assignment reach all demand locations earlier than the offline assignment. This is expected since OTA and the all-step assignment, we reassign vehicles to reduce the arrival time using the real-time information. For the same reason, the offline assignment takes less running time and results in fewer total assignments, as shown in Figure 17-(d) to (i).

Figure 17-(a), (b) & (c) show that OTA performs comparably with the all-step assignment in terms of the arrival time while using fewer assignments and with lower running time (Figure 17-(d) to (i)). In particular, Figure 17-(d), (e), & (f) shows that OTA with $\gamma = 0.5$ avoids at least half of the unnecessary assignments compared with that of the all-step case. Figures 17-(g), (h) & (i) show that all-time assignment



(a) Step 1 (b) Step 2 (c) Step 19 (d) Step 20

(e) Step 22 (f) Step 23 (g) Step 28 (h) Step 29

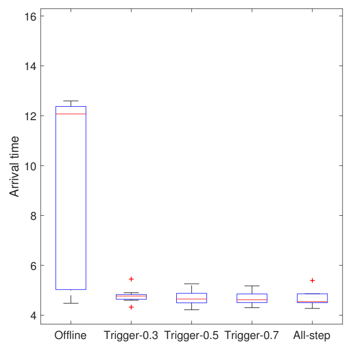
(i) Step 30 (j) Step 31 (k) Step 32 (l) Step 34

Fig. 16: One trial of OTA (Alg. 3) in action with 3 vehicles and 2 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.

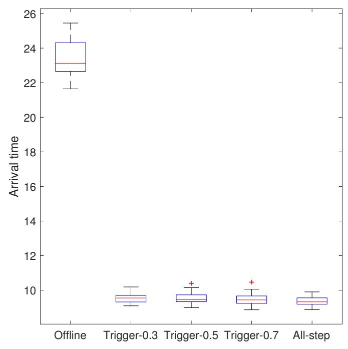
has longer running time when the problem becomes larger. OTA achieves a comparable performance with much lower running time when $\gamma$ is less than 0.5. We also observe from Figure 17 that the triggering ratio $\gamma$ trades-off performance (arrival time) and cost (number of assignments and running time). Particularly, a larger $\gamma$ triggers more assignments to reduce the arrival time, while at the same time, it takes more running time.

## VII. Conclusion and Discussion

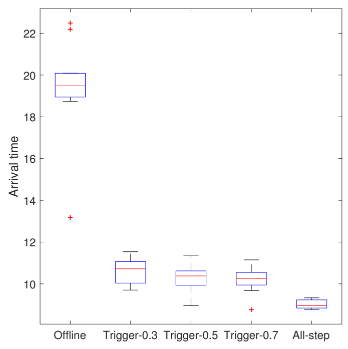
We studied a risk-aware discrete submodular maximization problem. We provided the first positive results for discrete CVaR submodular maximization for selecting a set under matroidal constraints. In particular, we proposed the Sequential Greedy Algorithm and analyzed its approximation ratio and the running time. We implemented the Sequential Greedy Algorithm in an online mobility-on-demand case and proposed an online-triggering assignment strategy (Algorithm 2). We demonstrated the two practical use-cases of the CVaR submodular maximization problem and verified the effectiveness



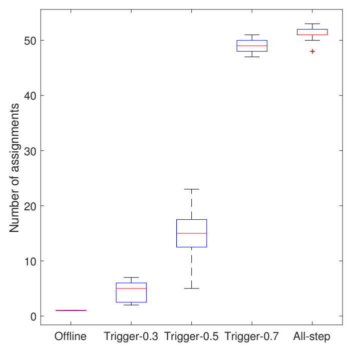
(a) Arrival time, 3 vehicles & 2 demands



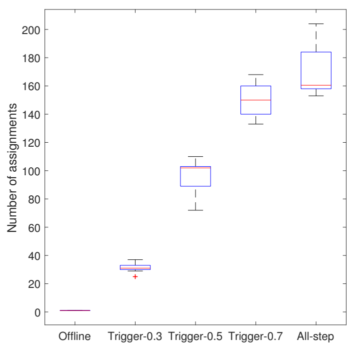
(b) Arrival time, 6 vehicles & 4 demands



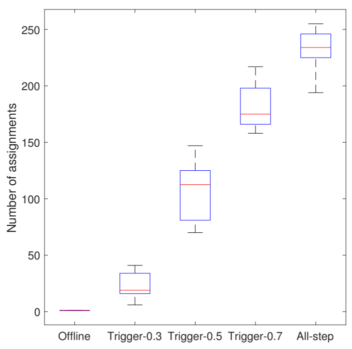
(c) Arrival time, 12 vehicles & 5 demands



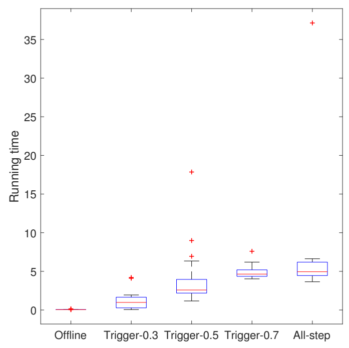
(d) Number of assignments, 3 vehicles & 2 demands



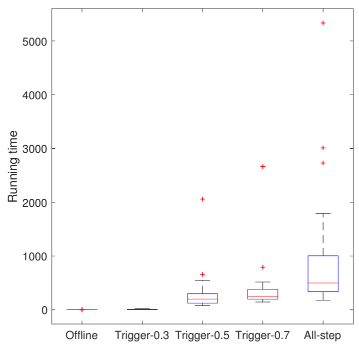
(e) Number of assignments, 6 vehicles & 4 demands



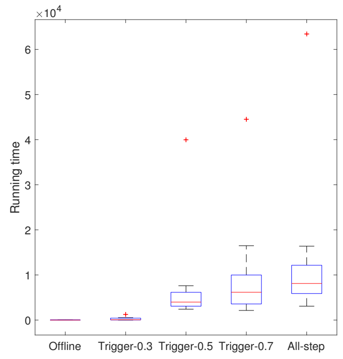
(f) Number of assignments, 12 vehicles & 5 demands



(g) Running time, 3 vehicles & 2 demands



(h) Running time, 6 vehicles & 4 demands



(i) Running time, 12 vehicles & 5 demands

Fig. 17: Comparison of the number of assignments, arrival time, and running time by the offline assignment, OTA (Alg. 3) with three triggering ratios, $\gamma = 0.3, 0.5, 0.7$, and all-step assignment.

of online triggering assignment approach (Algorithm 2).

Notably, our Sequential Greedy Algorithm works for any matroid constraint. In particular, the multiplicative approximation ratio can be improved to $1/k_f(1-e^{-k_f})$ if we know that the constraint is a uniform matroid [32, Theorem 5.4].

The additive term in our analysis depends on $\alpha$. This term can be large when the risk level $\alpha$ is very small. Our ongoing work is to remove this dependence on $\alpha$, 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. The algorithm also requires knowing $\Gamma$. We showed how to find $\Gamma$ (or an upper bound for it) for the two case studies considered in this paper. Devising a general strategy for finding $\Gamma$ is part of our ongoing work.

Our second line of ongoing work focuses on applying the risk-aware strategy to multi-vehicle routing, patrolling, and informative path planning in dangerous environments [37] and mobility on demand with real-world data sets (2014 NYC Taxicab Factbook).[3]

[3]http://www.nyc.gov/html/tlc/downloads/pdf/2014_taxicab_fact_book.pdf

## References

[1] J. O'Rourke, *Art gallery theorems and algorithms*. Oxford University Press Oxford, 1987, vol. 57.

[2] A. Krause, A. Singh, and C. Guestrin, "Near-optimal sensor placements in gaussian processes: Theory, efficient algorithms and empirical studies," *Journal of Machine Learning Research*, vol. 9, no. Feb, pp. 235–284, 2008.

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

[4] J. Vondrák, "Optimal approximation for the submodular welfare problem in the value oracle model," in *Proceedings of the fortieth annual ACM symposium on Theory of computing*. ACM, 2008, pp. 67–74.

[5] J. R. Spletzer and C. J. Taylor, "Dynamic sensor planning and control for optimally tracking targets," *The International Journal of Robotics Research*, vol. 22, no. 1, pp. 7–20, 2003.

[6] O. Tekdas and V. Isler, "Sensor placement for triangulation-based localization," *IEEE transactions on Automation Science and Engineering*, vol. 7, no. 3, pp. 681–685, 2010.

[7] P. Tokekar, V. Isler, and A. Franchi, "Multi-target visual tracking with aerial robots," in *Intelligent Robots and Systems (IROS 2014), 2014 IEEE/RSJ International Conference on*. IEEE, 2014, pp. 3067–3072.

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

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

[10] A. Singh, A. Krause, C. Guestrin, and W. J. Kaiser, "Efficient informative sensing using multiple robots," *Journal of Artificial Intelligence Research*, vol. 34, pp. 707–755, 2009.

[11] G. L. Nemhauser, L. A. Wolsey, and M. L. Fisher, "An analysis of approximations for maximizing submodular set functions–i," *Mathematical programming*, vol. 14, no. 1, pp. 265–294, 1978.

[12] M. L. Fisher, G. L. Nemhauser, and L. A. Wolsey, "An analysis of approximations for maximizing submodular set functionsii," in *Polyhedral combinatorics*. Springer, 1978, pp. 73–87.

[13] H. Ding and D. Castanón, "Multi-agent discrete search with limited visibility," in *Decision and Control (CDC), 2017 IEEE 56th Annual Conference on*. IEEE, 2017, pp. 108–113.

[14] R. K. Williams, A. Gasparri, and G. Ulivi, "Decentralized matroid optimization for topology constraints in multi-robot allocation problems," in *Robotics and Automation (ICRA), 2017 IEEE International Conference on*. IEEE, 2017, pp. 293–300.

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

[16] P. Dames, P. Tokekar, and V. Kumar, "Detecting, localizing, and tracking an unknown number of moving targets using a team of mobile robots," *The International Journal of Robotics Research*, vol. 36, no. 13-14, pp. 1540–1553, 2017.

[17] D. Kempe, J. Kleinberg, and É. Tardos, "Maximizing the spread of influence through a social network," in *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2003, pp. 137–146.

[18] A. Prorok, "Redundant robot assignment on graphs with uncertain edge costs," in *Distributed autonomous robotic systems*. Springer, 2019, pp. 313–327.

[19] G. C. Pflug, "Some remarks on the value-at-risk and the conditional value-at-risk," in *Probabilistic constrained optimization*. Springer, 2000, pp. 272–281.

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

[21] ——, "Conditional value-at-risk for general loss distributions," *Journal of banking & finance*, vol. 26, no. 7, pp. 1443–1471, 2002.

[22] J. Morgan, "Riskmetrics technical document," 1996.

[23] F. Yang and N. Chakraborty, "Algorithm for optimal chance constrained linear assignment," in *Robotics and Automation (ICRA), 2017 IEEE International Conference on*. IEEE, 2017, pp. 801–808.

[24] ——, "Algorithm for optimal chance constrained knapsack problem with applications to multi-robot teaming," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 1043–1049.

[25] H. Markowitz, "Portfolio selection," *The journal of finance*, vol. 7, no. 1, pp. 77–91, 1952.

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

[27] T. Maehara, "Risk averse submodular utility maximization," *Operations Research Letters*, vol. 43, no. 5, pp. 526–529, 2015.

[28] N. Ohsaka and Y. Yoshida, "Portfolio optimization for influence spread," in *Proceedings of the 26th International Conference on World Wide Web*. International World Wide Web Conferences Steering Committee, 2017, pp. 977–985.

[29] B. Wilder, "Risk-sensitive submodular optimization," in *Proceedings of the 32nd AAAI Conference on Artificial Intelligence*, 2018.

[30] L. Zhou and P. Tokekar, "An approximation algorithm for risk-averse submodular optimization," in *International Workshop on the Algorithmic Foundations of Robotics (WAFR)*, 2018, in press.

[31] A. Schrijver, *Combinatorial optimization: polyhedra and efficiency*. Springer Science & Business Media, 2003, vol. 24.

[32] 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 applied mathematics*, vol. 7, no. 3, pp. 251–274, 1984.

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

[34] E. Ackerman and E. Strickland, "Medical delivery drones take flight in east africa," *IEEE Spectrum*, vol. 55, no. 1, pp. 34–35, 2018.

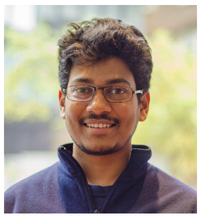
[35] K. J. Obermeyer and Contributors, "VisiLibity: A c++ library for visibility computations in planar polygonal environments," `http://www.VisiLibity.org`, 2008, r-1.

[36] G. Boeing, "Osmnx: New methods for acquiring, constructing, analyzing, and visualizing complex street networks," *Computers, Environment and Urban Systems*, vol. 65, pp. 126–139, 2017.

[37] 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," *Autonomous Robots*, vol. 42, no. 4, pp. 927–952, 2018.

**Lifeng Zhou** received the B.S. degree in Automation from Huazhong University of Science and Technology, Wuhan, China, in 2013, the M.Sc. degree in Automation from Shanghai Jiao Tong University, Shanghai, China, in 2016. He is currently pursuing the Ph.D. degree in Electrical and Computer Engineering, Virginia Tech, Blacksburg, VA, USA.

His research interests include multi-robot coordination, approximation algorithms, combinatorial optimization, deep learning, model predictive control, and resilient and risk-aware decision making.



**Pratap Tokekar** is an Assistant Professor in the Department of Computer Science at the University of Maryland. Previously, he was a Postdoctoral Researcher at the GRASP lab of University of Pennsylvania. Between 2015 and 2019, he was an Assistant Professor at the Department of Electrical and Computer Engineering at Virginia Tech. He obtained his Ph.D. in Computer Science from the University of Minnesota in 2014 and Bachelor of Technology degree in Electronics and Telecommunication from College of Engineering Pune, India in 2008. He is a recipient of the NSF CISE Research Initiation Initiative award and an Associate Editor for the IEEE Robotics and Automation Letters and Transactions of Automation Science and Engineering.

## 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 paper.

### 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 $\mathcal{V} = \{v_1, v_2, \cdots, v_V\}$ and the drive-way 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 the 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 $\{(v_i, v_j), (v_j, v_m), \cdots, (v_n, v_k)\}$. 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 $\mathcal{V}(P_{v_i,v_k})$.

Due 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 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

$$t(w_{v_i,v_j}) = \beta_2 \texttt{len}(w_{v_i,v_j}) / \texttt{maxv}(w_{v_i,v_j}) \tag{15}$$

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 travelling on the edges. Thus, as shown in Equation 15, 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,

$$t(P_{v_i,v_j}) = \sum_{v \in \mathcal{V}(P_{v_i,v_j})} t(v) + \beta_2 \sum_{w \in P_{v_i,v_j}} t(w) \tag{16}$$

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 Equation 16 with appropriate dimension manipulation. Clearly, the arrival efficiency to $v_j$ from $v_i$ through traveling along path $P_{v_i,v_j}$ can be computed as

$$e(P_{v_i,v_j}) = 1/t(P_{v_i,v_j}). \tag{17}$$

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

$$T^{\texttt{step}} = \min_{j \in \{1,\cdots,R\}} t_j^{\texttt{next}} \tag{18}$$

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

For simplicity, we assume the vehicles' initial positions and the demand locations are some nodes (intersections) of graph $\mathcal{G}$. [4]

### 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 (Alg. 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 (Alg. 3, line 1). Then, we use Equations 16, 17 to compute the arrival efficiency of these shortest paths (Alg. 3, line 2). Finally, we select a risk level $\alpha$ and use SGA (Alg. 1) to get an initial vehicle-to-demand assignment $\mathcal{S}$ with $\mathcal{S} = \bigcup_{i=1}^{N} \mathcal{S}_i$ (Alg. 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 Alg. 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 (Alg. 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 (Alg. 3, lines 4, 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 (Alg. 3, line 4).

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

**Per-step online travel (Alg. 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.

- if vehicle $j$ can reach the next intersection $v(p_j)+1$ within per-step time $T^{\texttt{step}}$ (Alg. 3, line 17), it moves to $v(p_j)+1$ and updates its current intersection $v(p_j)$ by the next intersection $v(p_j)+1$ (Alg. 3, line 18).
- if vehicle $j$'s waiting time $t_j^{\texttt{wait}}$ at the current intersection $v(p_j)$ is larger than per-step time $T^{\texttt{step}}$ (Alg. 3, line 19),

[4]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, $v(d_i)$, $i \in \{1, \cdots, N\}$. $R$ vehicles'
         initial positions $p_j$ with corresponding intersections,
         $v(p_j)$, $j \in \{1, \cdots, R\}$
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 efficiency of these shortest paths as
   $e(\{P^\star_{v(p_j),v(d_i)}\})$ by Equations 16, 17
3: $\mathcal{S} = \text{SGA}(e(\{P^\star_{v(p_j),v(d_i)}\}), \alpha)$
4: Set $v_j^{\texttt{prev}} = \emptyset,\ t_j^{\texttt{next}} = 0,\ j \in \{1, \cdots, R\}$ and
   $T^{\texttt{step}} = 0$
5: while not all $N$ demand locations are reached do
6:   for each vehicle $j \in \{1, \cdots, R\}$ 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 $t_j^{\texttt{next}} = t_j^{\texttt{wait}} + t(e_{v(p_j),v(p_j)+1})$
         (Eq. 15)
11:    else
12:      $t_j^{\texttt{next}} = t_j^{\texttt{next}} - T^{\texttt{step}}$
13:    end if
14:  end for
15:  update $T^{\texttt{step}} = \min_{j \in \{1, \cdots, R\}} t_j^{\texttt{next}}$ (Eq. 18)
16:  for each vehicle $j \in \{1, \cdots, R\}$ 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 $d_i,\ i \in \{1, \cdots, N\}$
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 efficiency of these shortest
       paths as $E(\{P^\star_{v(p_j),v(d_i)}\})$ by Equations 16, 17
30:    $\mathcal{S} = \texttt{SGA}(e(\{P^\star_{v(p_j),v(d_i)}\}), \alpha)$,
31:  end if
32: end while
```

it still needs to wait at $v(p_j)$ after per-step time $T^{\texttt{step}}$, but its waiting time is reduced by $T^{\texttt{step}}$ (Alg. 3, line 20).

- if vehicle $j$'s waiting time $t_j^{\texttt{wait}}$ at the current intersection $v(p_i)$ is less than per-step time $T^{\texttt{step}}$ (Alg. 3, line 21), it will leave the current intersection $v(p_j)$ or it is already travelling 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 (Alg. 3, line 23), which is computed by leveraging on Equation 15 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 (Alg. 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 (Alg. 3, line 26) and trigger the reassignment as long as there exists a demand location $i$ with,

$$\texttt{len}(P^\star_{v(p_j),v(d_i)}) \leq \gamma \texttt{len}(P^\star_{v(p'_j),v(d_i)}), \text{ and} \tag{19}$$

$$\texttt{deg}(P^\star_{v(p_j),v(d_i)}) \leq \texttt{deg}(P^\star_{v(p'_j),v(d_i)}), \tag{20}$$

where $\gamma \in (0, 1)$ denotes the triggering ratio (Alg. 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 (Alg. 3, lines 28-30). Notably, the triggering ratio regulates the frequency of reassignment, i.e., a larger $\gamma$ triggers more reassignments.

## *C. Proofs*

**Proof of Lemma 1:**

*Proof:* 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 $\mathcal{S}$.

$H(\emptyset, \tau) = \tau(1 - \frac{1}{\alpha})$ given $f(\mathcal{S}, y)$ is normalized ($f(\emptyset, y) = 0$). Thus, $H(\mathcal{S}, \tau)$ is not necessarily normalized since $\tau$ is not necessarily zero. See a similar proof in [20], [27]. ■

**Proof of Lemma 2:**

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

**Proof of Lemma 3:**

*Proof:* By using the result in [20, 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 $n_s$:**

*Proof:* We reach the statement by using the proof by Ohsaka and Yoshida [28, Lemmas 4.1-4.5]. We have via taking $c = \Gamma$ (the upper bound of the search separation $\tau$), in their Lemma 4.4. Then, in their Lemma 4.5, to make

$$|\text{CVaR}_\alpha(X) - \text{CVaR}_\alpha(\hat{X})| \le \epsilon, \tag{21}$$

we need to set

$$\text{sup}_{x\in\mathbb{R}} |F_X(x) - F_{\hat{X}}(x)| \le \frac{\epsilon}{\Gamma}. \tag{22}$$

By Dvoretzky-Kiefer-Wolfowitz inequality, Inequality 22 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:**

*Proof:* 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 $\tau$ and $\frac{\partial H(\mathcal{S},\tau)}{\partial \tau} < 1$. The properties of concavity and bound on the gradient give

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

We illustrate this claim by using Figure 18-(a). Since $\mathcal{S}_i^\star$ is the optimal set at $\tau_i$ for maximizing $H(\mathcal{S},\tau)$, the value of $H(\mathcal{S},\tau)$ with any other set $\mathcal{S} \in \mathcal{I}$ at $\tau_i$ 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 $\tau$ 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 $\tau \in [\tau_i, \tau_{i+1}]$ (Figure 18-(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. 18-(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, $H_i^\star - H(\mathcal{S}_i^\star,\tau_i) \le \frac{\partial H(\mathcal{S},\tau)}{\partial \tau}\Delta = \Delta$, $\forall i \in \{0,1,\cdots,\lceil\frac{\Gamma}{\Delta}\rceil\}$.

Then we have $H_i^\star - \max_i H(\mathcal{S}_i^\star,\tau_i) \le \Delta$, $\forall i \in \{0,1,\cdots,\lceil\frac{\Gamma}{\Delta}\rceil\}$. 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 $H_i^\star, i \in \{0,1,\cdots,\lceil\frac{\Gamma}{\Delta}\rceil\}$. Thus, we reach the claim in Lemma 4. ■

**Proof of Lemma 5:**

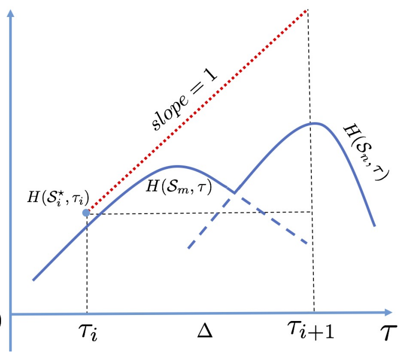
*Proof:* We use a the previous result [32, Theorem 2.3 ] for the proof of this claim. We know that for any given $\tau$, $H(\mathcal{S},\tau)$ is a non-normalized monotone submdoular function in $\mathcal{S}$ (Lemma 1). For maximizing normalized monotone submodular set functions, the greedy approach can give a $1+1/k_f$ approximation of the optimal performance with any matroid constraint [32, Theorem 2.3 ]. After normalizing $H(\mathcal{S},\tau)$ by $H(\mathcal{S},\tau) - H(\emptyset,\tau)$, we have

$$\frac{H(\mathcal{S}_i^G,\tau_i) - H(\emptyset,\tau_i)}{H(\mathcal{S}_i^\star,\tau_i) - H(\emptyset,\tau_i)} \ge \frac{1}{1+k_f}, \tag{23}$$

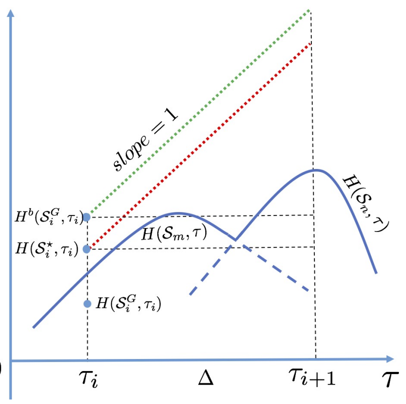
with any matroid constraint. Given $0 \le k_f \le 1$ and $H(\emptyset,\tau) = -\tau(\frac{1}{\alpha} - 1)$, we transform Equation 23 into,

$$\begin{aligned} H(\mathcal{S}_i^G,\tau_i) &\ge \frac{1}{1+k_f}H(\mathcal{S}_i^\star,\tau_i) - \frac{k_f}{1+k_f}\tau_i(\frac{1}{\alpha}-1) \\ &\ge \frac{1}{1+k_f}H(\mathcal{S}_i^\star,\tau_i) - \frac{k_f}{1+k_f}\Gamma(\frac{1}{\alpha}-1), \end{aligned} \tag{24}$$

where Equation 24 holds since $\Gamma$ is the upper bound of $\tau$. Thus, we prove the Lemma 5. ■



(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.

Fig. 18: Illustration of $H(\mathcal{S},\tau)$ within $\tau \in [\tau_i,\tau_{i+1}]$.

**Proof of Theorem 1:**

**Approximation bound.** From Equation 28 in Lemma 5, we have $H(\mathcal{S}_i^\star,\tau_i)$ is bounded by

$$H(\mathcal{S}_i^\star,\tau_i) \le (1+k_f)H(\mathcal{S}_i^G,\tau_i) + k_f\Gamma(\frac{1}{\alpha}-1). \tag{25}$$

Denote this upper bound as

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

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. 18-(a)/(b)) (Lemma 4). $H(\mathcal{S},\tau)$ must be also 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. 18-(b)). Similar to the proof in Lemma 4, we have $H_i^\star - H^b(\mathcal{S}_i^G,\tau_i) \le \Delta$ and

$$\max_{i\in\{0,1,\cdots,\lceil\frac{\Gamma}{\Delta}\rceil\}} H^b(\mathcal{S}_i^G,\tau_i) \ge H(\mathcal{S}^\star,\tau^\star) - \Delta. \tag{26}$$

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 25 and 26, we have

$$(1+k_f)H(\mathcal{S}^G, \tau^G) + k_f \Gamma(\frac{1}{\alpha} - 1) \geq H(\mathcal{S}^\star, \tau^\star) - \Delta. \quad (27)$$

By rearranging the terms, we get

$$\begin{aligned} H(\mathcal{S}^G, \tau^G) \geq & \frac{1}{1+k_f}(H(\mathcal{S}^\star, \tau^\star) - \Delta) \\ & - \frac{k_f}{1+k_f}\Gamma(\frac{1}{\alpha} - 1). \end{aligned} \quad (28)$$

Note that we use an oracle $\mathcal{O}$ to approximate $H(\mathcal{S}, \tau)$ with error $\epsilon$ by the sampling method (Eq. 21). By considering the sampling error $\epsilon$, 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 to 10, we use a "for" loop for searching $\tau$ which takes $\lceil \frac{\Gamma}{\Delta} \rceil$ evaluations. Second, within the "for" loop, we use the greedy algorithm to solve the subproblem (Alg. 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 (Alg. 1, line 5), and calculates the marginal gain of the remaining elements in $\mathcal{X}$ at each round (Alg. 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. Last, finding the best pair from storage set $\mathcal{M}$ (Alg. 1, line 11) takes $O(\lceil \frac{\Gamma}{\Delta} \rceil)$ time. Therefore, the computational complexity for SGA is,

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

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