# The Matroid Team Surviving Orienteers Problem: Constrained Routing of Heterogeneous Teams with Risky Traversal

Stefan Jorgensen, Robert H. Chen, Mark B. Milam, and Marco Pavone

***Abstract*— Consider a setting where robots must visit nodes in a graph, but each robot may fail when traversing an edge. The goal is to find a set of paths for a team of robots which maximizes the expected number of nodes collectively visited, while guaranteeing that the paths satisfy a notion of "independence" formalized by a matroid (e.g. limits on team size, number of visits to regions), and that the probabilities that each robot survives to its destination are above a given threshold. We call this problem the Matroid Team Surviving Orienteers (MTSO) problem, which has broad applications such as environmental monitoring in risky regions and search and rescue in dangerous conditions. We present the MTSO formally and detail numerous examples of matroids in a path planning context. We then propose an approximate greedy algorithm for selecting a feasible set of paths and prove that the value of the output is within a factor $\frac{p_s}{p_s+\lambda}$ of the optimum, where $p_s$ is the per-robot survival probability threshold and $1/\lambda \leq 1$ is the approximation factor of an oracle routine for the well known orienteering problem. We demonstrate the efficiency of our approach by applying it to a scenario where a team of robots must gather information while avoiding pirates in the Coral Triangle.**

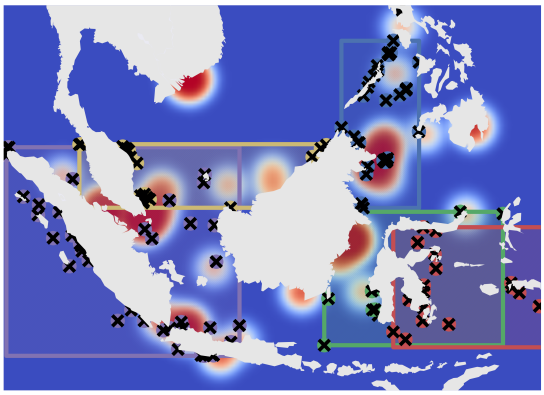

Fig. 1. Illustration of the MTSO setting for an ocean monitoring scenario. Various regions in the Coral Triangle are outlined by boxes, sites to visit within each region are marked by 'X', and the heatmap indicates the risk of robot failure inferred from piracy incidents. Data is from the Coral Triangle Atlas [4] and IMB Piracy Reporting Centre [5]. The objective is to find a set of paths for a heterogeneous team which maximizes the expected number of sites visited subject to survival probability constraints and independence constraints (e.g. limits on team size or sensor quantities).

## I. Introduction

Consider a scenario where mobile robotic sensors are used to monitor a number regions of the ocean, each of which may require different types of sensors. Due to weather and piracy, there is a risk that robots may "fail" when traveling from one region to another. A fleet manager seeks a set of paths which maximizes the expected number of sites monitored while satisfying various resource constraints. For example there may be limits imposed by the number of available robots of each type, the logistics of deploying the robots, the probability a given robot reaches its destination, or the amount of traffic a given region can support. If these constraints are downward closed (meaning any subset of a feasible set of paths is feasible) and satisfy an exchange property, then we can represent them using a *matroid* [1], which generalizes linear independence to set systems.

We formalize this exploration problem as a generalization of the orienteering problem [2], where one seeks a path which visits as many nodes in a graph as possible given a budget constraint and travel costs. In the aforementioned example the travel costs are the probability that a robot fails while traversing between sites, and we are looking for an independent set of a matroid which maximizes the expected number of nodes visited by at least one robot and ensures that the probabilities each vehicle reaches its destination is above a specified threshold. We call this problem formulation the "Matroid Team Surviving Orienteers" (MTSO) problem, illustrated in Figure 1. The MTSO problem is a significant generalization of our earlier work on the Team Surviving Orienteers (TSO) problem [3] which only imposes a maximum team size constraint (a very special case of a matroid constraint). Both the MTSO and TSO are distinct from previous work because of the notion of *risky traversal:* when a robot traverses an edge, there is a probability that it fails and does not visit any other nodes. This creates a complex, history-dependent coupling between the edges chosen and the distribution of nodes visited, which precludes the application of existing approaches available for the traditional orienteering problem.

The objective of this paper is to devise a constant-factor approximation algorithm for the MTSO problem. An efficient algorithm for the TSO was designed by exploiting a diminishing returns property known as submodularity [3], which for set functions means that $f(A \cup B) + f(A \cap B) \leq f(A) + f(B)$. Our key technical insight is that we can further exploit submodularity of the objective function to design an efficient solution algorithm for the much more general MTSO problem. We describe numerous applications of matroid constraints to path planning problems, and describe an

Stefan Jorgensen is with the Department of Electrical Engineering, Stanford University, Stanford, California 94305. His work is supported by NSF grant DGE-114747 stefantj@stanford.edu

Robert H. Chen and Mark B. Milam are with NG Next, Northrop Grumman, Redondo Beach, California 90278 {robert.chen,mark.milam}@ngc.com

Marco Pavone is with the Department of Aeronautics & Astronautics, Stanford University, Stanford, California 94035 pavone@stanford.edu

approximate greedy algorithm which enjoys a constant-factor approximation guarantee. Although a number of works have considered routing problems with submodular objectives [6], [7], [8], chance constraints [9], [10], or downward closed constraints [11], [12], [13] separately, the MTSO is novel because it combines all three aspects.

*Related work.* The orienteering problem (OP) has been extensively studied [14] and is known to be NP-hard. Over the past decade a number of constant-factor approximation algorithms have been developed for special cases of the problem. The *submodular orienteering* problem considers finding a single path which maximizes an arbitrary reward function of the nodes visited. The recursive greedy algorithm proposed in [6] yields a solution in quasi-polynomial time with a constant factor guarantee, and the cost-benefit greedy algorithm proposed in [8] yields a solution in polynomial time but only has a constant-factor guarantee with respect to a relaxed problem. Our work is distinct from these efforts because we consider a specific submodular function, but find a set of *paths* rather than a set of *edges* which form a path. The *risk-sensitive orienteering problem* [10] considers random edge weights and seeks to maximize the sum of rewards subject to a constraint on the probability that the path cost is large. Only a single path is considered, so there is no notion of independence between paths as in the MTSO.

A second closely-related area of research is represented by the vehicle routing problem (VRP) [15], which is a family of problems focused on finding a set of paths that maximize quality of service subject to budget or time constraints. The *rich vehicle routing problem* (RVRP) [12] considers much more general settings such as routing heterogeneous teams [11], "fleet dimensioning" (choosing team composition) [13], and incompatability constraints [16]. The vast majority of solution algorithms for the RVRP are heuristic [12] and do not consider risky traversal. We consider a narrower yet still expressive set of constraints and derive a polynomial time solution algorithm with a constant factor approximation guarantee.

A third related branch of literature is the informative path planning problem (IPP), which seeks to find a set of paths for mobile robotic sensors in order to maximize the information gained about an environment. One of the earliest IPP approaches [17] extends the recursive greedy algorithm of [6] using a spatial decomposition to generate paths for multiple robots, and provides performance guarantees using submodularity of information gain. While the structure of the IPP has many similarities to the MTSO problem (it is a multi-robot path planning problem with a submodular objective function which is non-linear and history dependent), it captures neither the constraints nor the notion of risky traversal which are central to the MTSO problem. Our general approach is inspired by works such as [18], which iteratively assigns paths to each robot, but for the MTSO problem we further exploit the problem structure to derive constant-factor guarantees.

A handful of papers apply submodular maximization and matroid constraints directly to robotics path planning applications. In [19], path constraints are represented using a $p$-system (which generalizes a matroid), and a submodular maximization problem with $p$-system constraints is solved to find the approximately most informative path. On the other end of the spectrum, [20] considers a decentralized submodular multi-robot task allocation problem, where the goal is to assign robots disjoint task sets which maximize private submodular functions. As with the submodular orienteering problem, the items in the ground set are *edges* in the graph, rather than *paths* in the graph as in the MTSO, and hence only a single path is planned, rather than a collection of paths. We apply submodular maximization at a higher level of abstraction and use an orienteering problem oracle in order to find high quality paths for each robot. This requires different analysis which utilizes results on approximate greedy algorithms for submodular maximization subject to matroid [21] and $p$-system [22] independence constraints.

*Statement of Contributions.* Matroids have been applied with great success to many fields of engineering such as electrical and structural network design [23], but they have only sparingly been used for vehicle routing problems. The contribution of this paper four-fold. First, we propose a generalization of the TSO problem (which itself is a generalization of the orienteering problem), referred to as the Matroid TSO problem. By considering matroid constraints, we extend the state of the art for the team orienteering problem, and by considering the risky traversal model we extend the state of the art for the rich vehicle routing problem. Second, we demonstrate how to use matroids to represent a variety of constraints such as coverage, deployment, team size limitations, sub-graph diversity constraints, team-wise risk constraints and nested cardinality constraints. Third, we extend the approximate greedy algorithm from [3] to the MTSO setting, and prove that the value of its output is $\Omega(\frac{p_s}{p_s+\lambda}\text{OPT})$, where OPT is the optimum value, $p_s$ is the per-robot survival probability threshold and $1/\lambda \leq 1$ is the approximation factor of an oracle routine for the solution to the orienteering problem (we note that, in practice $p_s$ is close to unity). Finally, we demonstrate the effectiveness of our algorithm for complex problems by considering an environmental monitoring application.

## II. Background

### A. Sets and Submodular Functions

Submodularity is the property of 'diminishing returns' for set functions. The following definitions are summarized from [24]. Given a set $\mathcal{X}$, its possible subsets are represented by $2^{\mathcal{X}}$. For two sets $X$ and $X'$, the set $X' \setminus X$ contains all elements in $X'$ but not $X$. A collection of disjoint subsets $\{X_m\}_{m=1}^M$ is called a *partition* of $\mathcal{X}$ if $\cup_{m=1}^M \mathcal{X}_m = \mathcal{X}$. A set function $f : 2^{\mathcal{X}} \to \mathbb{R}$ is said to be *normalized* if $f(\emptyset) = 0$ and to be *monotone* if for every $X \subseteq X' \subseteq \mathcal{X}$, $f(X) \leq f(X')$. A set function $f : 2^{\mathcal{X}} \to \mathbb{R}$ is *submodular* if for every $X \subseteq X' \subset \mathcal{X}$, $x \in \mathcal{X} \setminus X'$, we have

$$f(X \cup \{x\}) - f(X) \geq f(X' \cup \{x\}) - f(X').$$

The quantity on the left hand side is the *discrete derivative* of $f$ at $X$ with respect to $x$, which we write as $\Delta f(x \mid X)$.

### B. Independence Systems and Matroids

The following definitions are summarized from [1]. An independence system is a tuple of a finite ground set $\mathcal{X}$ and a downward closed family of independent sets $\mathcal{I} \subseteq 2^{\mathcal{X}}$, that is if $I' \subseteq I$ and $I \in \mathcal{I}$, then $I' \in \mathcal{I}$. A *base* is an independent set $I \in \mathcal{I}$ which is inclusion-wise maximal, that is for every $x \in \mathcal{X} \setminus I$, $I \cup x \notin \mathcal{I}$. A matroid is an independence system for which all bases have the same size (which is called the *rank* of the matroid), hence it extends the notion of linear independence to sets. There are many equivalent characterizations of matroids which are outside of the scope of this work, we refer the interested reader to [1] for more detail.

### C. The Approximate Greedy Algorithm

Given a matroid $(\mathcal{X}, \mathcal{I})$ and a submodular function $f$, a typical submodular maximization problem entails finding a set $X \in \mathcal{I}$ that maximizes $f$. Finding an optimal solution, $X^*$, is NP-hard for general submodular functions [21]. The *greedy algorithm* constructs a set $\bar{X}_K$ by iteratively adding an element $x$ from the feasible set,

$$\mathcal{X}_F(\bar{X}_\ell, \mathcal{I}) := \{x \in \mathcal{X} \setminus \bar{X}_\ell \mid \bar{X}_\ell \cup \{x\} \in \mathcal{I}\},$$

which maximizes the discrete derivative of $f$ at the partial set already selected. In other words the $\ell$th element satisfies:

$$\bar{x}_\ell \in \operatorname*{argmax}_{x \in \mathcal{X}_F(\bar{X}_{\ell-1}, \mathcal{I})} \Delta f(x \mid \bar{X}_{\ell-1}),$$

and items are chosen until no more can be added, that is $\mathcal{X}_F(\bar{X}_K, \mathcal{I}) = \emptyset$. If the function $f$ is monotone and non-negative, the greedy algorithm will choose $K$ items, where $K$ is the rank of the matroid $(\mathcal{X}, \mathcal{I})$. We refer to the optimization problem above as 'the greedy sub-problem' at step $\ell$. A well-known theorem proven by [21] states that if $f$ is a monotone, normalized, non-negative, and submodular function, then $f(\bar{X}_K) \geq \frac{1}{2} f(X^*)$. This is a powerful result, but if the set $\mathcal{X}$ is large we might only be able to approximately solve the greedy sub-problem. An $\alpha$–approximate greedy algorithm constructs the set $\hat{X}_K$ by iteratively adding elements which *approximately* maximize the discrete derivative of $f$ at the partial set already selected. Formally, for some fixed $\alpha \leq 1$, the $\ell$th element $\hat{x}_\ell$ satisfies:

$$\Delta f(\hat{x}_\ell \mid \hat{X}_{\ell-1}) \geq \alpha \Delta f(x \mid \hat{X}_{\ell-1}) \qquad \forall\ x \in \mathcal{X}_F(\hat{X}_{\ell-1}, \mathcal{I}).$$

In this setting, [22] shows in Appendix B that the value of the approximate greedy set is close to optimal:

*Theorem 1 (Approximate greedy guarantee [22]):* Let $(\mathcal{X}, \mathcal{I})$ be a matroid with rank $K$ and $f : 2^{\mathcal{X}} \to \mathbb{R}^+$ a non-negative monotone submodular set function. If $\hat{X}_K$ is a set chosen by an $\alpha$-approximate greedy algorithm, then for any $X \in \mathcal{I}$,

$$f(\hat{X}_K) \geq \frac{\alpha}{\alpha + 1} f(X).$$

### D. Graphs

Let $\mathcal{G}(\mathcal{V}, \mathcal{E})$ denote an undirected graph, where $\mathcal{V}$ is the node set and $\mathcal{E} \subset \mathcal{V} \times \mathcal{V}$ is the edge set. Explicitly, an edge is a pair of nodes $(i, j)$, and represents the ability to travel between nodes $i$ and $j$. If the graph is directed, then the edge is an ordered pair of nodes, and represents the ability to travel from the *source node* $i$ to the *sink node* $j$. A graph is called *simple* if there is only one edge which connects any given pair of nodes. A path is an ordered sequence of *unique* nodes such that there is an edge between adjacent nodes. For $n \geq 0$, we denote the $n$th node in path $\rho$ by $\rho(n)$ and the number of edges in $\rho$ by $|\rho|$. Note that $\rho(|\rho|)$ is the last node in path $\rho$. The graph $\mathcal{G}_m(\mathcal{V}_m, \mathcal{E}_m)$ is called a sub-graph of $\mathcal{G}$ if $\mathcal{V}_m \subseteq \mathcal{V}$ and $\mathcal{E}_m \subseteq \mathcal{E}$. The sub-graph of $\mathcal{G}$ induced by $\mathcal{V}' \subseteq \mathcal{V}$ is the graph $\mathcal{G}'(\mathcal{V}', \mathcal{E}')$ where $\mathcal{E}' := \{(i, j) \in \mathcal{E} \mid i, j \in \mathcal{V}'\}$.

## III. Problem Statement

We use a similar setting and notation as the TSO problem presented in [3] which we repeat below to keep this paper self-contained. We then generalize the problem statement for the matroid constraint case.

### A. Formal Problem Description

Let $\mathcal{G}$ be a simple graph with $|\mathcal{V}| = V$ nodes. Edge weights $\omega : \mathcal{E} \to (0, 1]$ correspond to the probability of survival for traversing an edge. Time is discretized into iterations $n = 0, 1, \ldots, N$. At iteration $n$ a robot following path $\rho$ traverses edge $e^n_\rho = (\rho(n-1),\ \rho(n))$. Robots are indexed by variable $k$, and for each we define the independent Bernoulli random variables $s^k_n(\rho)$ which are 1 with probability $\omega(e^n_\rho)$ and 0 with probability $1 - \omega(e^n_\rho)$. If robot $k$ follows path $\rho$, the random variables $a^k_n(\rho) := \prod_{i=1}^n s^k_i(\rho)$ can be interpreted as being 1 if the robot 'survived' all of the edges taken until iteration $n$ and 0 if the robot 'fails' on or before iteration $n$.

Given a start node $v_s$, a terminal node $v_t$, and survival probability $p_s$ we must find at most $K \geq 1$ paths $\{\rho_k\}_{k=1}^K$ (one for each of $K$ robots) such that, for all $k$, the probability that $a^k_{|\rho_k|}(\rho_k) = 1$ is at least $p_s$, $\rho_k(0) = v_s$ and $\rho_k(|\rho_k|) = v_t$. The set of paths which satisfy these constraints is written as $\mathcal{X}(p_s, \omega)$. One can readily test whether $\mathcal{X}(p_s, \omega)$ is empty as follows: Set edge weights as $-\log(\omega(e))$, and for each node $j$, compute the shortest path from $v_s$ to $j$, delete the edges and nodes in that path, and compute the shortest path from $j$ to $v_t$. If the sum of edge weights along both paths is less than $-\log(p_s)$ then the node is reachable, otherwise it is not. Using Dijkstra's algorithm this approach can prove whether $\mathcal{X}(p_s, \omega)$ is empty after $O(V^2 \log(V))$ computations, so we assume that $\mathcal{X}(p_s, \omega)$ is non-empty.

Define the indicator function $\mathbb{I}\{x\}$, which is 1 if $x$ is true (or nonzero) and zero otherwise. Define the Bernoulli random variables for $j = 1, \ldots, V$:

$$z^k_j(\rho) := \sum_{n=1}^{|\rho|} a^k_n(\rho)\, \mathbb{I}\{\rho(n) = j\},$$

which are 1 if robot $k$ following path $\rho$ visits node $j$ and 0 otherwise ($z^k_j(\rho)$ is binary because a path is defined as a unique set of nodes). Note that $z^k_j(\rho)$ is independent of $z^{k'}_j(\rho')$ for $k \neq k'$. The number of times that node $j$ is visited by robots following the paths $\{\rho_k\}_{k=1}^K$ is given by $\sum_{k=1}^K z^k_j(\rho_k)$, and we write the probability that exactly $m$ robots visit node $j$ as $p_j(m, \{\rho_k\}_{k=1}^K)$. In this paper we are

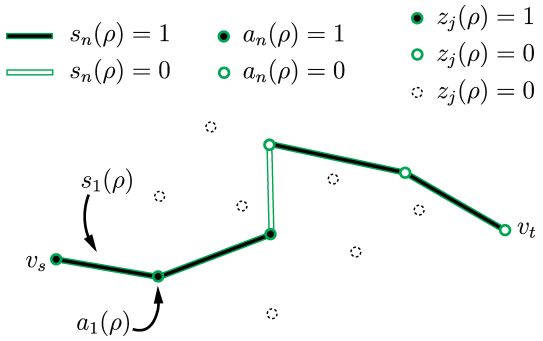


Fig. 2. Illustration of the notation used (. A robot plans to take path $\rho$, whose edges are represented by lines. The fill of the lines represent the value of $s_n(\rho)$. In this example $s_3(\rho) = 0$, which means that $a_3(\rho) = a_4(\rho) = a_5(\rho) = 0$. The variables $z_j(\rho)$ are zero if either the robot fails before reaching node $j$ or if node $j$ is not on the planned path.

primarily interested in the probability that no robots visit node $j$, which has the simple expression:

$$p_j\left(0, \{\rho_k\}_{k=1}^K\right) = \prod_{k=1}^K (1 - \mathbb{E}[z_j^k(\rho_k)]).$$

Let $d_j > 0$ be the reward accumulated for visiting node $j$ at least once, and define $\mathcal{X}$ as the set containing $K$ copies of *each* path in $\mathcal{X}(p_s, \omega)$. Given a matroid $(\mathcal{X}, \mathcal{I})$ with rank $K$, we are interested in finding an independent set which maximizes the weighted expected number of nodes visited. Since the objective is non-negative and submodular [3], we assume without loss of generality that the size of the optimizing set is $K$, and state the Matroid TSO problem formally:

**Matroid TSO (MTSO) Problem:** Given a graph $\mathcal{G}$, edge weights $\omega$, survival probability threshold $p_s$ and matroid $(\mathcal{X}, \mathcal{I})$, maximize the weighted expected number of nodes visited by at least one robot:

$$\begin{aligned}
\underset{\{\rho_k\}_{k=1}^K \in \mathcal{I}}{\text{maximize}} \quad & \sum_{j=1}^V d_j \left(1 - p_j\left(0, \{\rho_k\}_{k=1}^K\right)\right) \\
\text{subject to} \quad & \mathbb{P}\left\{a^k_{|\rho_k|}(\rho) = 1\right\} \geq p_s \quad k = 1, \ldots, K \\
& \rho_k(0) = v_s \quad k = 1, \ldots, K \\
& \rho_k(|\rho|) = v_t \quad k = 1, \ldots, K
\end{aligned}$$

The optimization is over the feasible sets of the matroid, and the solution is a set of $K$ paths. The objective represents the cumulative expected reward obtained by the robots following the selected paths. The first set of constraints enforces the survival probability, the second and third sets of constraints enforce the initial and final node constraints.

The MTSO problem can be viewed as a set maximization problem with a matroid constraint, where the domain of optimization is the family of independent sets $\mathcal{I}$. Crucially, since the objective is a submodular function [3], the guarantee from [22] implies that the greedily selected set of paths will achieve reward close to the optimum.

## IV. Examples and Applications

In this section we highlight several examples of matroids and their applications in the context of the MTSO problem.

### A. Uniform Matroid

Given a positive integer $K$, the independent sets of a uniform matroid are all subsets of the ground set with at most $K$ elements. Optimization with a uniform matroid constraint is equivalent to imposing cardinality constraints on the solution size, which is the standard TSO problem.

### B. Binary Matroid

Given a function $\phi : \mathcal{X} \to \{0, 1\}^M$, the independent sets of a binary matroid induced by $\phi$ are all subsets $X \subseteq \mathcal{X}$ such that the vectors $\{\phi(x)\}_{x \in X}$ are linearly independent.

*Application to coverage:* Consider a setting where we require each robot to focus on a different region. Define the regions as $M$ node subsets $S_m \subseteq \mathcal{V}$, and define the 'focus' of a path as the index of the region which contains the most nodes of the path (with ties broken deterministically). Let $m_\rho$ be the smallest index corresponding to a subset which path $\rho$ focuses on, that is $|S_{m_\rho} \cap \rho| \geq |S_m \cap \rho|$ for $m = 1, \ldots, M$. Now define $\phi(\rho)$ as the $m_\rho$th canonical basis vector in $\mathbb{R}^M$. Requiring the solution to be an independent set of the binary matroid induced by $\phi$ and with ground set $\mathcal{X}(p_s, \omega)$, enforces the desired diversity of focus.

### C. Transversal Matroid

Given subsets $\{\mathcal{X}_m\}_{m=1}^M$ of the ground set, the independent sets of the transversal matroid are all subsets $X$ which are partial transversals of $\{\mathcal{X}_m\}_{m=1}^M$. In other words, if $X \in \mathcal{I}$, we can assign each element $x_i \in X$ a unique number $m_i \in \{1, \ldots, M\}$ such that $x_i \in \mathcal{X}_{m_i}$.

*Application to launch constraints:* Consider a scenario where only one robot can start on each outgoing edge of $v_s$. This could arise for example when robots are aerial vehicles which must launch from runways, and only one can launch from a runway in each direction. Let $N(v_s)$ be the set of nodes with an edge to $v_s$, and define the subsets $\mathcal{X}_m = \{\rho \in \mathcal{X}(p_s, \omega) \mid \rho(1) = n_m\}$ for $n_m \in N(v_s)$. That is, we have one subset for each starting edge. Requiring that the solution is an independent set of the transversal matroid induced by $\{\mathcal{X}_m\}_{m=1}^M$ enforces the launch constraints.

*Application to heterogeneous teams:* Suppose we have $M$ types of robots, a robot of type $m$ has feasible path set $\mathcal{X}_m$, and that we can deploy at most $K_m$ robots of type $m$. Requiring the solution to be an independent set of a transversal matroid induced by $K_m$ copies of $\mathcal{X}_m$ for $m = 1, \ldots, M$ enforces that no more than $K_m$ robots of type $m$ are selected.

*Application to sub-graph diversity:* Suppose we are given sub-graphs $\mathcal{G}_m = (\mathcal{V}_m, \mathcal{E}_m)$ for $m = 1, \ldots, M$ with $v_s, v_t \in \mathcal{V}_m$, $\mathcal{V}_m \subseteq \mathcal{V}$, $\mathcal{E}_m \subseteq \mathcal{E}$, and we require that no more than $K_m$ robots take paths in sub-graph $\mathcal{G}_m$. Let $\mathcal{X}_m$ correspond to the set of feasible paths in sub-graph $\mathcal{G}_m$. Requiting that the solution is an independent set of a transversal matroid induced by $K_m$ copies of the sets $\mathcal{X}_m$ enforces the sub-graph diversity constraint.

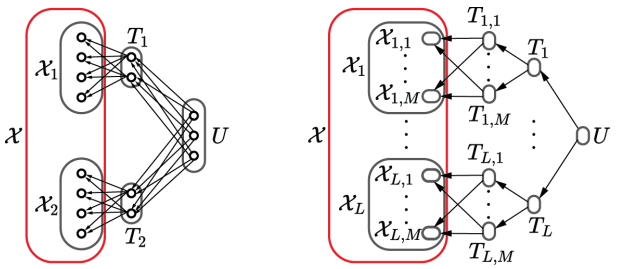


Fig. 3. Illustration of multi-partite graphs which form gammoids. Left: An illustration of the graph with two layers of cardinality constraints. Right: An illustration of the graph with three layers. Boxes represent clusters of nodes, and lines represent edges which connect each node of the right cluster to each node of the left cluster.

*Application to risk constraints:* Suppose we have $M$ risk thresholds $p_s^1 \leq \cdots \leq p_s^M$. This setting could arise when there is a constraint on the *risk* of many robot failures, but requiring uniform survival thresholds would be too conservative to visit all of the nodes. Then we can choose $\{p_s^m\}_{m=1}^M$ in order to provide the necessary flexibility while still maintaining tight control on risk. Requiring the solution to be an independent set of the transversal matroid induced by $\{\mathcal{X}(p_s^m, \omega)\}_{m=1}^M$ enforces the desired constraints.

### D. Gammoid

A gammoid is induced by a directed graph $D(S, E)$ with a subset of nodes corresponding to elements in the ground set, e.g. $\mathcal{X} \subseteq S$, and a subset $U \subseteq S$. We say that two sets of nodes $X, Y \subseteq S$ are *linked* if $|X| = |Y|$ and there are $|X|$ node-disjoint paths from $X$ to $Y$. The independent sets of a gammoid induced by $D$, $U$ are all subsets $X \subseteq \mathcal{X}$ such that some subset of $U$ is linked to $X$.

*Application to nested cardinality constraints:* Consider a simple setting where the ground set is partitioned by the sets $\{\mathcal{X}_1, \mathcal{X}_2\}$ and we may choose up to 2 items from $\mathcal{X}_1$, 2 items from $\mathcal{X}_2$ and 3 items total. The independent sets of a gammoid induced by the multi-partite graph in Figure 3 satisfy these constraints. This setting is easily extended to more complicated scenarios. For the MTSO, we could first partition based on robot types, then by sub-graphs, and finally by risk thresholds (or in a different order). An illustration of the graph for three layers of partitioning (e.g. robot types, sub-graphs, and team size) is shown in Figure 3. It is not necessary for the node groups to form a partition.

### E. Truncation

Given $K \in \mathbb{N}$ and a matroid $(\mathcal{X}, \mathcal{I})$, let $\mathcal{I}' := \{I \in \mathcal{I} \mid |I| \leq K\}$ which is the set of independent sets with at most $K$ elements. Then $(\mathcal{X}, \mathcal{I}')$ is a matroid and is called the $K$-truncation of $(\mathcal{X}, \mathcal{I})$. If the maximum team size is $K$, we can represent this constraint in addition to any of the scenarios above by using the $K$-truncation of the appropriate matroid.

## V. Algorithm

At a high level our approach to solving the MTSO problem is the same as for the TSO problem, where we exploit submodularity of the objective function using an approximate greedy algorithm. The crucial difference is that now we must restrict the domain of the greedy sub-problem to independent sets of the matroid constraint. We show how to do this in Section V-A and get a $p_s/\lambda$–approximate greedy algorithm. We describe the rest of our algorithm for the MTSO problem in Section V-B, and combine results derived in this paper with results from [3] and [22] to prove an approximation guarantee in Section V-C. In Section V-D we characterize the complexity of our approach, which depends on the ability to efficiently partition the independent sets of a matroid, so in Section V-E we give efficient partition routines for several matroids.

### A. Linear Relaxation for Greedy Sub-problem

Given a previously selected set of paths, $X_{L-1} = \{\rho_\ell\}_{\ell=1}^{L-1}$, the greedy sub-problem for the MTSO problem at step $L$ requires us to find a path $\rho_L$ from the set $\mathcal{X}_F(X_{L-1}, \mathcal{I})$ which maximizes the discrete derivative of the objective function at $X_{L-1}$ with respect to $\rho_L$. We denote this discrete derivative by $\Delta J(\rho_L \mid X_{L-1})$. The greedy sub-problem is very difficult for the MTSO problem: it requires finding a path which maximizes submodular node rewards subject to a budget constraint (this is the submodular orienteering problem). Although there are negative results about the efficiency of constant-factor approximation algorithms for *general* submodular orienteering problems [6], a polynomial time algorithm was devised specifically for the objective of the TSO problem (where $\mathcal{X}_F(X_{L-1}, \mathcal{I}) = \mathcal{X}$) in [3] by using a linearization procedure. The problem is relaxed by replacing the probability that robot $L$ traversing path $\rho$ visits node $j$, $\mathbb{E}[z_j^L(\rho)]$, with $\zeta_j$, the maximum probability that any robot following a feasible path can visit node $j$:

$$\zeta_j := \max_{\rho \in \mathcal{X}(p_s, \omega)} \mathbb{E}[z_j^L(\rho)].$$

For a given graph this upper bound can be found easily by using Dijkstra's algorithm with log-transformed edge weights $\omega_O(e) := -\log(\omega(e))$. Let $\mathbb{I}_j(\rho)$ be equal to 1 if node $j$ is in $\rho$ and 0 otherwise. In the relaxed problem we are looking to maximize the sum:

$$\Delta \bar{J}(\rho \mid X_{L-1}) := \sum_{j=1}^{V} \mathbb{I}_j(\rho) \zeta_j d_j p_j \left(0, X_{L-1}\right),$$

which represents an *optimistic* estimate of $\Delta J(\rho \mid X_{L-1})$.

The feasible set, $\mathcal{X}_F(X_{L-1}, \mathcal{I})$, can always be partitioned into sets $\{\mathcal{X}_m\}_{m=1}^M$, where $\mathcal{X}_m$ is the subset of paths in $\mathcal{X}$ which have all nodes and edges in a corresponding sub-graph $\mathcal{G}_m$. This is apparent from the fact that for each $\rho \in \mathcal{X}_F(X_{L-1}, \mathcal{I})$, we can construct a sub-graph $\mathcal{G}_m$ which has exactly the nodes and edges in $\rho$. Since a path is defined as a *unique* list of nodes and edges, and since feasible paths must start at $v_s$ and end at $v_t$, the sub-graph contains only one feasible path, $\rho$. In practice we can often partition $\mathcal{X}_F(X_{L-1}, \mathcal{I})$ using a small number of sub-graphs, as detailed in Section V-E.

We can find the (approximately) best path by solving an orienteering problem with budget $-\log(p_s)$ on each of the sub-graphs $\mathcal{G}_{m,O}$, which have the same edges and nodes as $\mathcal{G}_m$ but have log-transformed edge weights $\omega_O(e)$ and node

rewards $\nu_L(j) = \zeta_j d_j p_j(0, X_{L-1})$. The best of the solutions to the orienteering problems is a path in the feasible set that maximizes the sum of node rewards (which is $\Delta\bar{J}(\rho \mid X_{L-1})$), and satisfies $\sum_{e\in\rho} -\log(\omega(e)) \leq -\log(p_s)$, which is equivalent to $\mathbb{P}\{a^L_{|\rho|}(\rho) = 1\} \geq p_s$.

Although solving the orienteering problem is NP-hard, several polynomial-time approximation algorithms exist which guarantee that the returned objective is lower bounded by a factor of $1/\lambda \leq 1$ of the optimal objective. For undirected planar graphs $\lambda = (1+\epsilon)$ [25], for undirected graphs $\lambda = (2+\epsilon)$ [26], and for directed graphs [6] gives a guarantee in terms of the number of nodes. Using such an oracle, we extend Lemma 2 from [3] to the matroid setting:

*Lemma 1 (Single robot constant-factor guarantee):* Let `Orienteering` be a routine that solves the orienteering problem within constant-factor $1/\lambda$, that is for $c_j > 0$ and node weights $\nu(j) = c_j\zeta_j$, path $\hat{\rho}_m$ output by the routine when given graph $\mathcal{G}_m$ and any feasible path $\rho_m$ in $\mathcal{G}_m$,

$$\sum_{j=1}^{V} \mathbb{I}_j(\hat{\rho}_m)\nu(j) \geq \frac{1}{\lambda}\sum_{j=1}^{V} \mathbb{I}_j(\rho_m)\nu(j).$$

Let $\mathcal{X}_F(X_L, \mathcal{I})$ be partitioned by $\{\mathcal{X}_m\}_{m=1}^M$, where $\mathcal{X}_m$ is the set of feasible paths in sub-graph $\mathcal{G}_m$, and define $\hat{\rho}$ as the best path returned by the `Orienteering` routine among the $m$ sub-graphs. Then for any $X_L \in \mathcal{I}$, $c_j > 0$ and any $\rho \in \mathcal{X}_F(X_L, \mathcal{I})$, we have:

$$\sum_{j=1}^{V} c_j\mathbb{E}[z_j(\hat{\rho})] \geq \frac{p_s}{\lambda}\sum_{j=1}^{V} c_j\mathbb{E}[z_j(\rho)].$$

*Proof:* By definiton of $\{\mathcal{X}_m\}_{m=1}^M$, for any $\rho \in \mathcal{X}_F(X_L, \mathcal{I})$ there is an $m_\rho$ such that $\rho \in \mathcal{X}_{m_\rho}$. We have from the definitions of $\hat{\rho}$, $\zeta_j$ and the `Orienteering` routine:

$$\begin{aligned}\sum_{j=1}^{V} c_j\mathbb{E}[z_j(\rho)] &\leq \sum_{j=1}^{V} \mathbb{I}_j(\rho)\zeta_j c_j \leq \lambda\sum_{j=1}^{V} \mathbb{I}_j(\hat{\rho}_{m_\rho})\zeta_j c_j \\ &\leq \lambda\sum_{j=1}^{V} \mathbb{I}_j(\hat{\rho})\zeta_j c_j.\end{aligned}$$

Since $\hat{\rho}$ is feasible, $\mathbb{I}_j(\hat{\rho})p_s\zeta_j \leq \mathbb{I}_j(\hat{\rho})p_s \leq \mathbb{E}[z_j(\hat{\rho})]$, which combined with the equation above completes the proof. ■

The intuition behind this result is that for $p_s$ close to unity no feasible path can be very risky and so the probability that a robot *actually* reaches a node will not be too far from the maximum probability that it *could* reach the node.

### B. Greedy Approximation for the MTSO Problem

Using this relaxation with $c_j = d_j p_j(0, X_{L-1})$ we have an $p_s/\lambda$–approximate algorithm for the greedy sub-problem at step $L$. This gives us a $(\frac{p_s}{p_s+\lambda})$–approximate greedy algorithm for the MTSO problem, as detailed next.

Define the method `Dijkstra`($\mathcal{G}$, $i$, $j$), which returns the length of the shortest path from node $i$ to $j$ on the edge weighted graph $\mathcal{G}$ using Dijkstra's algorithm. Given an edge weighted graph $\mathcal{G}$, node rewards $\nu$, the `Orienteering`$(\mathcal{G}, \nu)$ routine solves the orienteering problem (assuming $v_s = 1$, $v_t = V$ and budget $-\log(p_s)$) within factor $1/\lambda$, and returns the best path. Given an independent set $X \in \mathcal{I}$, the `Partition`$(X)$ routine returns a set of sub-graphs $\{\mathcal{G}_m\}_{m=1}^M$ such that $\{\mathcal{X}_m\}_{m=1}^M$ partitions $\mathcal{X}_F(X, \mathcal{I})$. Pseudocode for our algorithm is given in Figure 4. We begin by forming the graph $\mathcal{G}_O$ with log-transformed edge weights $\omega_O(e)$, and then use Dijkstra's algorithm to compute the maximum probability that a node can be reached. For each robot $k = 1, \ldots, K$, we partition the feasible set and solve the orienteering problem for each sub-graph $\{\mathcal{G}_m\}_{m=1}^M$ to greedily choose the path that maximizes $\Delta\bar{J}(\rho \mid \hat{X}_{k-1})$.

**procedure** MGREEDYSURVIVORS($\mathcal{G}, K$)
  Form $\mathcal{G}_O$ from $\mathcal{G}$, such that $v_s = 1$, $v_t = V$
  **for** $j = 1, \ldots, V$ **do**
    $\zeta_j \leftarrow \exp(-$`Dijkstra`$(\mathcal{G}_O, 1, j))$
    $\nu_1(j) \leftarrow \zeta_j d_j$
  **end for**
  $\rho_1 \leftarrow$ `Orienteering`$(\mathcal{G}_O, \nu_1)$
  **for** $k = 1, \ldots, K-1$ **do**
    $\mathbb{E}[a^k_0(\rho_k)] \leftarrow 1$
    **for** $n = 1, \ldots, |\rho_k|$ **do**
      $\mathbb{E}[a^k_n(\rho_k)] \leftarrow \mathbb{E}[a^k_{n-1}(\rho_k)]\omega(e^n_{\rho_k})$
      $\nu_{k+1}(\rho_k(n)) \leftarrow (1 - \mathbb{E}[a^k_n(\rho_k)])\nu_k(\rho_k(n))$
    **end for**
    **for** $\mathcal{G}_m$ in `Partition`$(\{\rho_\ell\}_{\ell=1}^k)$ **do**
      Form $\mathcal{G}_{m,O}$ from $\mathcal{G}_m$
      $\hat{\rho}_m \leftarrow$ `Orienteering`$(\mathcal{G}_{m,O}, \nu_{k+1})$
    **end for**
    $\rho_{k+1} \leftarrow \underset{\rho\in\hat{\rho}_1,\ldots,\hat{\rho}_M}{\arg\max}\ \Delta J\left(\rho \mid \{\rho_\ell\}_{\ell=1}^k\right)$
  **end for**
**end procedure**

Fig. 4. Approximate greedy algorithm for solving the MTSO problem.

### C. Approximation Guarantees

In this section we combine the results from Section II-C and V-A to prove that the output of the `MGreedySurvivors` algorithm is close to the optimal solution to the MTSO problem.

*Theorem 2 (Multi-robot constant-factor guarantee):* Let $1/\lambda$ be the constant-factor guarantee for the `Orienteering` routine as in Lemma 1, and assign robot $\ell$ the path $\hat{\rho}_\ell$, which is the best of the outputs from the orienteering routine given graphs $\mathcal{G}_{m,O}$ with node weights

$$\nu_\ell(j) = \zeta_j d_j p_j\left(0, \{\hat{\rho}_k\}_{k=1}^{\ell-1}\right).$$

Let $X^* = \{\rho^*_k\}_{k=1}^K$ be an optimal solution to the MTSO. Then the weighted expected number of nodes visited by a team of robots following the paths $\hat{X}_K = \{\hat{\rho}_\ell\}_{\ell=1}^K$ is at least a fraction $\gamma = \frac{p_s}{p_s+\lambda}$ of the optimum:

$$\sum_{j=1}^{V} d_j\left(1 - p_j\left(0, \hat{X}_K\right)\right) \geq \gamma\sum_{j=1}^{V} d_j\left(1 - p_j\left(0, X^*\right)\right).$$

*Proof:* Using Lemma 1 with $c_j$ chosen appropriately for the objective function, we have a constant-factor guarantee $\alpha = p_s/\lambda$ for the problem of finding the path from $\mathcal{X}_F(X_\ell, \mathcal{I})$ that maximizes the discrete derivative of the

objective function at step $\ell + 1$. Now applying Theorem 1 to our objective function (which by Lemma 2 of [3] is normalized, non-negative, monotone and submodular) we have the desired result. ■

In many scenarios of interest $p_s$ is quite close to 1, since robots are typically valuable or difficult to replace.

### D. Computational Complexity

Suppose that the complexity of the `Orienteering` oracle on the graph $\mathcal{G}$ is $C_O$, the complexity of the partition routine at each iteration is at most $C_P$, and the largest number of sets in the partition of the feasible set is $M$. Then the complexity of our algorithm is $O(V^2 \log(V)) + O(KV^2) + O(KMC_O) + O(KC_P)$. The first term is the complexity of running Dijkstra's algorithm to calculate $\zeta_j$ for all nodes, the second term is the complexity of updating the $V$ weights $K$ times (each update costs at most $|\rho_k| \leq V$ flops), the third term is the complexity of solving the orienteering problems, and the final term is the cost of partitioning the feasible sets. For $\lambda = 1 + \epsilon$ or $\lambda = 2 + \epsilon$, the oracle complexity is $C_O = V^{O(1/\epsilon)}$, and so the complexity is dominated by $KMC_O$. Each of the $M$ orienteering problems can be solved independently, so on a highly parallel system the complexity will scale as $KC_O$. If $M$ is small and a suitable approximation algorithm is used for `Orienteering` (such as [6], [26], [25]), the procedure described above will have reasonable computation time even for large team sizes.

### E. Efficiently Partitioning the Feasible Set

In this section we demonstrate how to partition the feasible set using a small number of sub-graphs for several examples.

*1) Launch constraints (Transversal matroid):* Given a set $X_k$, all paths which take edges $(v_s, \rho_\ell(1))$, $\ell = 1, \ldots, k$ are infeasible. Hence $M = 1$ and the sub-graph is the graph induced by the edges $\mathcal{E}' = \mathcal{E} \setminus \{(v_s, \rho_\ell(1)\}_{\ell=1}^{k}$.

*2) Heterogeneous teams (Transversal matroid):* The feasible sets are already partitioned by graphs $\mathcal{G}_m$ corresponding to the graph that a robot of type $m$ uses. Given a set $X_k$, the partition routine returns all sub-graphs $\mathcal{G}_m$ for which $X_k$ has fewer than $K_m$ paths.

*3) Sub-graph diversity:* If the feasible path sets $\mathcal{X}_m$ are disjoint, then the routine simply returns any sub-graph for which $X_k$ has fewer than $K_m$ paths in $\mathcal{X}_m$.

If the feasible path sets are not disjoint, then the routine solves an assignment problem to see whether an additional robot can be assigned a path in $\mathcal{X}_m$ without violating the constraints. The routine then returns any of the corresponding sub-graphs $\mathcal{G}_m$ which robots can still be assigned a path from. The worst-case complexity of each assignment is $O(K^{2.5})$ (using the Hopcraft-Korp algorithm), and there are at most $MK$ assignments, giving worst-case complexity $O(MK^{3.5})$.

*4) Risk constraints:* Given a set of paths $X_k$, the partition routine finds the smallest value $\hat{m}$ such that we can add a path with survival probability threshold $p_s^{\hat{m}}$ (this can be done naively with complexity $O(M^2)$). Then the routine returns the sub-graph induced by the vertex set $V_{\hat{m}} := \{j = 1, \ldots, V \mid \zeta_j \geq p_s^{\hat{m}}\}$, which are all vertices reachable within the desired survival probability threshold. Note that in this case we also must change the budget used by the `Orienteering` routine to $-\log(p_s^{\hat{m}})$.

*5) Nested Cardinality constraints:* Given a set of paths $X_k$, the partition routine tests whether a path can be added to each of the sets in the deepest layer (which partitions $\mathcal{X}$). If the sets are disjoint, this can be done in linear time, otherwise an assignment routine is run as in the Section V-E.3. Then a sub-graph corresponding to each of the subsets which we can still assign a path from is returned. The manner in which these sub-graphs are computed depends on the application, as described in each of the sections above.

## VI. Empirical Results

We consider the application illustrated in Figure 1: the Coral Triangle has a diverse ecosystem and the goal is to use a robotic fleet with multiple sensor configurations to monitor the wildlife populations in 'Marine Protected Areas'. We use piracy incident data [5] and model attacks as a Poisson random variable in a manner similar to [27]. We selected 106 marine protected areas from [4] as nodes in a complete graph and computed the shortest (minimum risk) paths between each pair of sites to find the edge weights. In our scenario, we can deploy up to 25 robots of three types (at most twelve of each type), and each robot type has a different utility in each region. Each region can support the traffic of at most three robots of each type.

We require that the expected losses over the worst 5% of outcomes be no more than five failed robots (this is called the Conditional Value at Risk, and denoted $\text{CVaR}_{0.05}$). For the data shown, the most difficult node to reach requires survival probability 0.64, but if we use a uniform $p_s = 0.64$, the risk is unacceptably high ($\text{CVaR}_{0.05} = 16.26$). We satisfy the constraint $\text{CVaR}_{0.05} \leq 5$ by setting $p_s^1, \ldots, p_s^5 = 0.6$, and $p_s^6, \ldots, p_s^{25} = 0.859$.

All of the constraints above can be represented using a gammoid, and the feasible set can be partitioned using at most $M = 15$ sub-graphs. For the `Orienteering` routine we use both an open-source Variable Neighborhood Search (VNS) heuristic produced by HeuristicLab [28] and a mixed-integer linear program (MIP) formulation solved using CPLEX with a 5% tolerance. We observe that the VNS heuristic gives near-optimal paths, and solves all of the subproblems for each of the twenty five robots in 51 seconds on a quad-core 4GHz processor with 32GB RAM available. The MIP oracle takes 60 seconds and yields nearly the same results. While the MIP oracle can find high quality paths quickly, as the tolerance is decreased the computation time dramatically increases. The value of optimally solving the sub-problems is small and does not necessarily lead to better overall behavior, as shown in Figure 5. The upper bound is computed as the smaller of 1) the upper bound found using the greedily selected paths and the approximation ratio, 2) the sum of all node rewards. Note that although the theoretical approximation guarantee is 0.36, a team of 10 robots achieves 0.928 the maximum possible reward.

## VII. Conclusion

We formulate the *Matroid Team Surviving Orienteers* (MTSO) problem, where we seek a set of paths which form

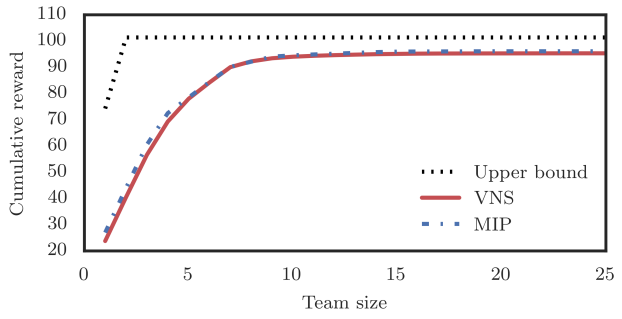


Fig. 5. Cumulative reward for paths planned by `MGreedySurvivors` using the MIP and VNS `Orienteering` routines. Both approaches compute near-optimal sets of paths, though VNS is somewhat faster.

an independent set of a matroid and maximizes the expected number of nodes visited by at least one robot and ensures the probabilities each robot reaches its destination are above a threshold. This problem is a significant generalization of our earlier work on the Team Surviving Orienteers problem [3], and is distinct from previous work because it combines a submodular objective, chance constraints, and matroid constraints. We give numerous applications of matroids to robotic path planning such as coverage, launch constraints, limits on the number of available robots of multiple types, restrictions on the amount of traffic which can flow through a region, and combinations of the above. The MTSO is particularly challenging to solve because of the risky traversal model, where a robot might not complete its planned path. This creates a complex, history-dependent coupling between the edges chosen and the distribution of nodes visited. We present an approximate greedy algorithm for solving the MTSO problem which guarantees that its output achieves at least $\frac{p_s}{p_s+\lambda}$ of the optimal reward. The algorithm relies on a partitioning routine to satisfy the matroid constraints, and we show numerous examples where our algorithm runs in polynomial time. Finally, we demonstrate the efficiency of our approach by applying it to a scenario where a team of robots must gather information while avoiding pirates in the Coral Triangle.

There are numerous directions for future work: First, extending our results to more general set systems such as antimatroids or greedoids would enable significantly more expressive constraints such as precedence constraints. Second, extending the continuous greedy algorithm [22] to our setting could lead to tighter bounds, though the extension is non-trivial. Finally, we are interested in the dual problem, where the objective is to find the smallest team which satisfies a coverage constraint.

## References


[1] A. Schrijver, *Combinatorial optimization: polyhedra and efficiency*. Springer Science & Business Media, 2002, vol. 24.

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

[3] S. Jorgensen, R. H. Chen, M. B. Milam, and M. Pavone, "The team surviving orienteers problem: Routing robots in uncertain environments with survival constraints," in *IEEE Int. Conf. on Robotic Computing*, 2017, to appear.

[4] A. Cros, N. Ahamad Fatan, A. White, S. J. Teoh, S. Tan, C. Handayani, C. Huang, N. Peterson, R. Venegas Li, H. Y. Siry, R. Fitriana, J. Gove, T. Acoba, M. Knight, R. Acosta, N. Andrew, and D. Beare, "The coral triangle atlas: An integrated online spatial database system for improving coral reef management," *PLOS ONE*, vol. 9, no. 6, pp. 1–7, 06 2014.

[5] Imb piracy reporting centre. Available at https://www.icc-ccs.org/piracy-reporting-centre.

[6] C. Chekuri and M. Pál, "A recursive greedy algorithm for walks in directed graphs," in *IEEE Symp. on Foundations of Computer Science*, 2005.

[7] A. M. Campbell, M. Gendreau, and B. W. Thomas, "The orienteering problem with stochastic travel and service times," *Annals of Operations Research*, vol. 186, no. 1, pp. 61–81, 2011.

[8] H. Zhang and Y. Vorobeychik, "Submodular optimization with routing constraints," in *Proc. AAAI Conf. on Artificial Intelligence*, 2016.

[9] A. Gupta, R. Krishnaswamy, V. Nagarajan, and R. Ravi, "Approximation algorithms for stochastic orienteering," in *ACM-SIAM Symp. on Discrete Algorithms*, 2012.

[10] P. Varakantham and A. Kumar, "Optimization approaches for solving chance constrained stochastic orienteering problems," in *International Conference on Algorithmic Decision Theory*. Springer, 2013.

[11] Ç. Koç, T. Bektaş, O. Jabali, and G. Laporte, "Thirty years of heterogeneous vehicle routing," *European Journal of Operational Research*, vol. 249, no. 1, pp. 1–21, 2016.

[12] R. Lahyani, M. Khemakhem, and F. Semet, "Rich vehicle routing problems: From a taxonomy to a definition," *European Journal of Operational Research*, vol. 241, no. 1, pp. 1 – 14, 2015.

[13] A. Hoff, H. Andersson, M. Christiansen, G. Hasle, and A. Løkketangen, "Industrial aspects and literature survey: Fleet composition and routing," *Computers & Operations Research*, vol. 37, no. 12, pp. 2041–2061, 2010.

[14] A. Gunawan, H. C. Lau, and P. Vansteenwegen, "Orienteering problem: A survey of recent variants, solution approaches and applications," *European Journal of Operational Research*, vol. 255, no. 2, pp. 315 – 332, 2016.

[15] H. N. Psaraftis, M. Wen, and C. A. Kontovas, "Dynamic vehicle routing problems: Three decades and counting," *Networks*, vol. 67, no. 1, pp. 3–31, 2016.

[16] P. Kilby and A. Verden, "Flexible routing combing constraint programming, large neighbourhood search, and feature-based insertion," in *Proc. Workshop on Artificial Intelligence and Logistics*, 2011.

[17] 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.

[18] N. Atanasov, J. Le Ny, K. Daniilidis, and G. J. Pappas, "Decentralized active information acquisition: theory and application to multi-robot slam," in *Proc. IEEE Conf. on Robotics and Automation*, 2015.

[19] S. T. Jawaid and S. L. Smith, "Informative path planning as a maximum traveling salesman problem with submodular rewards," *Discrete Applied Mathematics*, vol. 186, pp. 112–127, 2015.

[20] P. Segui-Gasco, H.-S. Shin, A. Tsourdos, and V. Seguí, "Decentralised submodular multi-robot task allocation," in *IEEE/RSJ Int. Conf. on Intelligent Robots & Systems*. IEEE, 2015, pp. 2829–2834.

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

[22] G. Calinescu, C. Chekuri, M. Pál, and J. Vondrák, "Maximizing a submodular set function subject to a matroid constraint," in *Proc. Int. Conf. on Integer Programming and Combinatorial Optimization*, 2007.

[23] K. Murota, *Matrices and Matroids for Systems Analysis*, ser. Algorithms and Combinatorics. Springer Science & Business Media, 2009, vol. 20.

[24] A. Krause and D. Golovin, "Submodular function maximization," *Tractability: Practical Approaches to Hard Problems*, vol. 3, no. 19, p. 8, 2012.

[25] K. Chen and S. Har-Peled, "The orienteering problem in the plane revisited," in *ACM Symp. on Computational Geometry*, 2006.

[26] C. Chekuri, N. Korula, and M. Pál, "Improved algorithms for orienteering and related problems," *ACM Transactions on Algorithms*, vol. 8, no. 3, p. 23, 2012.

[27] O. Vaněk, M. Jakob, O. Hrstka, and M. Pěchouček, "Agent-based model of maritime traffic in piracy-affected waters," *Transportation Research Part C: Emerging Technologies*, vol. 36, pp. 157–176, 2013.

[28] S. Wagner and M. Affenzeller, "Heuristiclab: A generic and extensible optimization environment," in *Adaptive and Natural Computing Algorithms*. Springer, 2005.