The Matroid Team Surviving Orienteers Problem and its Variants: Constrained Routing of Heterogeneous Teams with Risky Traversal

Journal:	*International Journal of Robotics Research*
Manuscript ID	IJR-19-3523
Manuscript Type:	Original Article
Date Submitted by the Author:	22-Feb-2019
Complete List of Authors:	Jorgensen, Stefan; Stanford University, William F. Durand building Pavone, Marco; Stanford University, Aeronautics & Astronautics
Keyword:	Robotics in Hazardous Fields < Field and Service Robotics, Search and Rescue Robots < Field and Service Robotics, Networked Robots < Mobile and Distributed Robotics SLAM
Abstract:	Consider deploying a team of robots in order to visit sites in a risky environment (i.e., where a robot might be lost during a traversal), subject to team-based operational constraints such as limits on team composition, traffic throughputs, and launch constraints. We formalize this problem using a graph to represent the environment, enforcing probabilistic survival constraints for each robot, and using a matroid (which generalizes linear independence to sets) to capture the team-based operational constraints. The resulting ``Matroid Team Surviving Orienteers'' (MTSO) problem has broad applications for robotics such as informative path planning, resource delivery, and search and rescue. We demonstrate that the objective for the MTSO problem has submodular structure, which leads us to develop two polynomial time algorithms which are guaranteed to find a solution with value within a constant factor of the optimum. The second of our algorithms is an extension of the accelerated continuous greedy algorithm, and can be applied to much broader classes of constraints while maintaining bounds on suboptimality. In addition to in-depth analysis, we demonstrate the efficiency of our approaches by applying them to a scenario where a team of robots must gather information while avoiding dangers in the Coral Triangle and characterize scaling and parameter selection using a synthetic dataset.

Journal Title
XX(X):1–16

DOI: 10.1177/ToBeAssigned
www.sagepub.com/
SAGE

The Matroid Team Surviving Orienteers Problem and its Variants: Constrained Routing of Heterogeneous Teams with Risky Traversal

Stefan Jorgensen[1] and Marco Pavone[2]

Abstract

Consider deploying a team of robots in order to visit sites in a risky environment (i.e., where a robot might be lost during a traversal), subject to team-based operational constraints such as limits on team composition, traffic throughputs, and launch constraints. We formalize this problem using a graph to represent the environment, enforcing probabilistic survival constraints for each robot, and using a matroid (which generalizes linear independence to sets) to capture the team-based operational constraints. The resulting "Matroid Team Surviving Orienteers" (MTSO) problem has broad applications for robotics such as informative path planning, resource delivery, and search and rescue. We demonstrate that the objective for the MTSO problem has submodular structure, which leads us to develop two polynomial time algorithms which are guaranteed to find a solution with value within a constant factor of the optimum. The second of our algorithms is an extension of the accelerated continuous greedy algorithm, and can be applied to much broader classes of constraints while maintaining bounds on suboptimality. In addition to in-depth analysis, we demonstrate the efficiency of our approaches by applying them to a scenario where a team of robots must gather information while avoiding dangers in the Coral Triangle and characterize scaling and parameter selection using a synthetic dataset.

Keywords

Submodular optimization, matroid constraints, path planning, multi-robot control

1 Introduction

Consider a scenario where mobile robotic sensors are used to monitor a number of regions of the ocean, each of which may require different sensing capabilities. Due to uncertain and possibly adversarial events (e.g., bad weather or 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* Schrijver (2002), which generalizes linear independence to set systems and has structure amenable to optimization.

We formalize this class of exploration problems as a generalization of the orienteering problem Golden et al. (1987), where one seeks a path which visits as many nodes in a graph as possible given a budget constraint and travel costs. In our earlier example the travel costs are the probability that a robot fails while traversing between sites, and we are looking for a set of such paths which is an independent set of a matroid, 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.

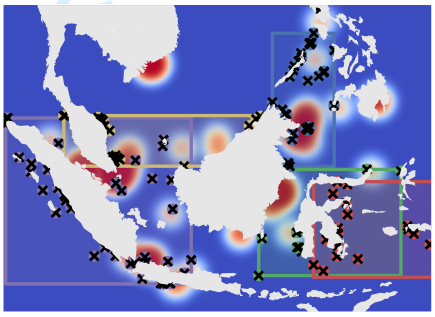

Figure 1. Illustration of an ocean monitoring scenario with various regions and sites to visit within each region marked by 'X', and the heat map indicates the risk of robot failure inferred from piracy incidents. The objective is to find a set of paths for a heterogeneous team which maximizes the expected number of sites visited subject to several constraints. Data is from the Coral Triangle Atlas Cros et al. (2014) and International Chamber of Commerce: Commercial Crime Services (2017).

[1] Robotics Engineer, area17, Oakland, California 94607,
[2] Department of Aeronautics & Astronautics, Stanford University, Stanford, California 94305

Corresponding author:
Marco Pavone
Email: pavone@stanford.edu

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 Jorgensen et al. (2018), 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, path-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 constant-factor approximation algorithms for the MTSO problem. An efficient algorithm for the TSO was designed by exploiting a diminishing returns property known as submodularity Jorgensen et al. (2018), which for set function f and sets A and B 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 efficient solution algorithms for the much more general MTSO problem. We describe numerous applications of matroid constraints to path planning problems, and describe an approximate greedy algorithm which enjoys a constant-factor approximation guarantee. We then develop an approach based on the accelerated continuous greedy algorithm Badanidiyuru and Vondrák (2014) which is the state of the art for matroid optimization and provides tighter guarantees that are easily extended to more general settings. Although a number of works have considered routing problems with submodular objectives Chekuri and Pál (2005); Campbell et al. (2011); Zhang and Vorobeychik (2016), chance constraints Gupta et al. (2012); Varakantham and Kumar (2013), or downward closed constraints Koç et al. (2016); Lahyani et al. (2015); Hoff et al. (2010) separately, the MTSO is novel because it combines all three aspects.

Related work. The orienteering problem (OP) has been extensively studied Gunawan et al. (2016) 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 submodular reward function of the nodes visited. The recursive greedy algorithm proposed in Chekuri and Pál (2005) yields a solution in quasi-polynomial time with a constant factor guarantee, and the cost-benefit greedy algorithm proposed in Zhang and Vorobeychik (2016) 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* Varakantham and Kumar (2013) 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 the vehicle routing problem (VRP) Psaraftis et al. (2016), 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) Lahyani et al. (2015) considers settings such as routing heterogeneous teams Koç et al. (2016), "fleet dimensioning" (choosing team composition) Hoff et al. (2010), and incompatability constraints Kelareva et al. (2014). The vast majority of solution algorithms for the RVRP are heuristic Lahyani et al. (2015) and do not consider risky traversal. A notable exception are *linear temporal logical* (LTL) constraints, which have exact integer programming formulations Karaman and Frazzoli (2011) and transition system formulations Smith et al. (2011); Ulusoy et al. (2014), the complexity of which grows exponentially in the team size. In our work, we consider a narrower yet still expressive set of constraints and derive polynomial time solution algorithms with constant factor approximation guarantees.

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 Singh et al. (2009) extends the recursive greedy algorithm of Chekuri and Pál (2005) 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 Atanasov et al. (2015), 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 applications. In Jawaid and Smith (2015), 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 for a single agent. Multi-robot task allocation problems are cast as decentralized submodular maximization problems with matroid constraints by Segui-Gasco et al. (2015) (who considers private reward functions), and Williams et al. (2017) (who considers communication constraints). In both cases the objective is to assign robots to tasks, rather than to find a high reward set of paths for a team of robots. Recent work by Corah and Michael (2017) uses distributed matroid optimization for informative path planning, though their setting has no notion of risky traversal. In our work we use an orienteering problem oracle in order to find high quality paths for each robot while considering risky traversal. This requires different analysis which utilizes results on approximate greedy algorithms for submodular maximization subject to matroid Fisher et al. (1978) and p-system Calinescu et al. (2007) independence constraints.

Statement of Contributions: Matroids have been applied with great success to many fields of engineering such as electrical and structural network design Murota (2009), but they have rarely been used for robotic routing problems. The contribution of this paper is six-fold. First, we propose a generalization of the TSO 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, traffic constraints, team-wise risk constraints, and nested cardinality constraints. Third, we extend the approximate greedy algorithm from Jorgensen et al. (2018) to the MTSO setting, and prove that the value of its output is at least $\frac{p_s}{p_s+\lambda}$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 (in practice, p_s is close to unity). Fourth, we extend the accelerated continuous greedy algorithm Badanidiyuru and Vondrák (2014) to the MTSO problem, which is its first application to robotics. We develop a fast implementation specific to the MTSO and show its output is at least $(1-\delta)(1-e^{-p_s/(\lambda+\delta p_s)})$, where $\delta \ll 1$ is the discretization step size. Fifth, we demonstrate the effectiveness of both algorithms for complex problems by considering an environmental monitoring application and give an empirical timing characterization for both algorithms. Finally, we highlight a number of extensions of this work in detail, such as p-system constraints, linear packing constraints, and coverage variants.

This paper is a significant expansion of our conference paper Jorgensen et al. (2017a), which introduced the MTSO and greedy algorithm. In particular, we present and analyze the accelerated continuous greedy algorithm which is a new approach to the problem that provides stronger guarantees with broader applications. We have also significantly expanded the numerical experiments and discuss extensions beyond matroids.

Organization: The rest of this paper is organized as follows. In Section 2 we review key background information necessary for our results. In Section 3 we present the formal problem statement for the MTSO and describe several applications in Section 4. In Section 5 we outline the algorithms and give performance guarantees. Section 6 demonstrates the practicality of our algorithm via numerical experiments. In Section 7 we discuss a number of extensions of our work which can be solved using the continuous greedy algorithm. Finally, in Section 8, we draw our conclusions and discuss directions for future work.

2 Background

2.1 Sets and Submodular Functions

Submodularity is the property of 'diminishing returns' for set functions. The following definitions are summarized from Krause and Golovin (2014). 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 $\Delta f(x \mid X)$.

The *multilinear extension* of a submodular function f at $y \in [0,1]^{|\mathcal{X}|}$ can be understood as taking the expected value of the function with respect to a random set $R(y)$ which contains each item $x \in \mathcal{X}$ with probability $y_x \in [0,1]$. Formally, we have for $x \in \mathcal{X}$,

$$\mathbb{P}\{x \in R(y)\} = y_x,$$

and for $X \subseteq \mathcal{X}$,

$$\mathbb{P}\{R(y) = X\} = \prod_{x \in X} y_x \prod_{x' \in X^c} (1 - y_{x'}).$$

The multilinear extension of f at y is then defined as the expected value of $f(R(y))$:

$$F(y) := \mathbb{E}[f(R(y))] = \sum_{X \subseteq \mathcal{X}} f(X) \prod_{x \in X} y_x \prod_{x' \in X^c} (1 - y_{x'}).$$

For optimization problems the multilinear extension is used in a similar fashion as a linear programming relaxation of a mixed integer program, where integer variables (i.e., whether element x is in the solution) are replaced by continuous relaxations (i.e., $y_x \in [0,1]$), and the resulting solution is rounded to get a set. We discuss this optimization framework more in Section 2.3.2.

2.2 Independence Systems and Matroids

The following definitions are summarized from Schrijver (2002). 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 $x \in \mathcal{X} \setminus I$, we have $I \cup x \notin \mathcal{I}$. A matroid $(\mathcal{X}, \mathcal{I})$ 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 Schrijver (2002) for more detail.

2.3 Approximate Greedy Algorithms

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. In general finding an optimal solution, X^*, requires an exponential number of evaluations of the function, is NP-hard for special cases such as coverage functions Fisher et al. (1978), and cannot be approximated to closer than factor $(1 - e^{-1})$ in polynomial time.

2.3.1 Greedy Algorithm Given a matroid $(\mathcal{X}, \mathcal{I})$ with rank K and an independent set X, the *feasible set*, $\mathcal{X}_F(X, \mathcal{I})$, is the set of all items not in X which can be added to X while preserving independence:

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

The *greedy algorithm* constructs a set $\bar{X}_K$ by iteratively adding an element x from the feasible set which maximizes the discrete derivative of f at the partial set already selected. In other words, the ℓth 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 ℓ. A well-known theorem proven by Fisher et al. (1978) 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 α–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 $\alpha \in (0, 1]$, the ℓth element $\hat{x}_\ell$ satisfies:

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

In this setting, Calinescu et al. (2007) shows in Appendix B that the value of the approximate greedy set is close to optimal:

Theorem 1. Approximate greedy guarantee Calinescu et al. (2007). *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 α-approximate greedy algorithm, then*

$$f(\hat{X}_K) \geq \frac{\alpha}{\alpha + 1} f(X), \qquad \textit{for all } X \in \mathcal{I}.$$

Note that while the approximate greedy algorithm is simple and relatively fast, in the best case ($\alpha = 1$) the guarantee is $1/2$, but the state of the art has a $1 - e^{-1} \simeq 0.63$ guarantee with matching hardness bounds.

2.3.2 Accelerated Continuous Greedy Algorithm The *accelerated continuous greedy algorithm* (ACGA) recently proposed by Badanidiyuru and Vondrák (2014) optimizes the multilinear extension using coordinate gradient ascent (see pseudocode in Algorithm 1) and achieves a $1 - e^{-1}$ guarantee. For practical implementations the algorithm is discretized using steps of size $\delta \ll 1$. During each step the algorithm constructs an independent set by greedily maximizing the multilinear extension (line 6). After selecting each item, the corresponding component of the vector y is incremented by δ. After $1/\delta$ independent sets are selected, the fractional solutions represented by y are rounded into an integral solution by calling the `SwapRounding` procedure from Chekuri et al. (2010). In particular, Theorem 2.1 from Chekuri et al. (2010) states that if X is the output of `SwapRounding` given a vector y in a matroid polytope (which always holds in the context of this paper), then X is independent, $\mathbb{E}[f(X)] \geq F(y)$, and the probability $f(X) < (1 - \epsilon)F(y)$ is bounded above by $\exp(-\epsilon^2 F(y)/8)$. The ACGA enjoys strong theoretical performance: Badanidiyuru and Vondrák (2014) guarantees that the expected output (with respect to randomness from rounding) is at least a fraction $(1 - e^{-1} - \delta)$ of the optimum, which matches the hardness bounds. However the ACGA selects $1/\delta$ more items than the greedy algorithm and optimizing the multilinear extension is generally quite difficult.

Algorithm 1 Pseudocode for the Accelerated Continuous Greedy Algorithm

1: **procedure** ACCELCONTINUOUSGREEDY($\mathcal{I}, F, \delta$)
2: $y \leftarrow \vec{0}$
3: **for** $i = 1, \ldots, 1/\delta$ **do**
4: $X_i \leftarrow \emptyset$
5: **for** $\ell = 1, \ldots, K$ **do**
6: $x_\ell \leftarrow \operatorname*{argmax}_{x \in \mathcal{X}_F(X_i, \mathcal{I})} F(y + \delta \mathbf{1}_x) - F(y)$
7: $y_{x_\ell} \leftarrow y_{x_\ell} + \delta$, $X_i \leftarrow X_i \cup x_\ell$
8: **end for**
9: **end for**
10: $\hat{X} \leftarrow$ `SwapRounding`(y)
11: **end procedure**

2.4 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 nth node in path ρ by $\rho(n)$ and the number of edges in ρ by $|\rho|$. Note that $\rho(|\rho|)$ is the last node in path ρ. 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 $\mathcal{E}' := \{(i, j) \in \mathcal{E} \mid i, j \in \mathcal{V}'\}$.

3 Formal Problem Statement

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 ρ traverses edge $e_\rho^n = (\rho(n-1), \rho(n))$. Robots are indexed by variable k, and for each robot and iteration we define the independent Bernoulli random variables $s_n^k(\rho)$ which are 1 with probability $\omega(e_\rho^n)$ and 0 with probability $1 - \omega(e_\rho^n)$. If robot k follows path ρ, the random variables $a_n^k(\rho) := \prod_{i=1}^n s_i^k(\rho)$ 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 (see Figure 2).

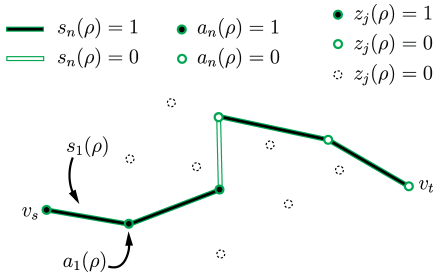

Figure 2. Illustration of the notation used. A robot plans to take path ρ, whose edges are represented by lines. The fill of the lines represent the value of $s_n(\rho)$. In this example $s_3(\rho) = 0$, which means that $a_3(\rho) = a_4(\rho) = a_5(\rho) = 0$. The variables $z_j(\rho)$ are zero if either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

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 test whether $\mathcal{X}(p_s, \omega)$ is empty with $O(V^2 \log(V))$ computations using the method described in Jorgensen et al. (2018), so we assume that it 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 $j = 1, \ldots, V$, $k = 1, \ldots, K$:

$$z_j^k(\rho) := \sum_{n=1}^{|\rho|} a_n^k(\rho)\, \mathbb{I}\{\rho(n) = j\},$$

which are 1 if robot k following path ρ visits node j and 0 otherwise ($z_j^k(\rho)$ is binary because a path is defined as a unique set of nodes). Note that $z_j^k(\rho)$ is independent of $z_j^{k'}(\rho')$ 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 $\sum_{k=1}^K z_j^k(\rho_k)$, 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 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, monotone, and submodular Jorgensen et al. (2018), 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 ω, 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 independent 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 Jorgensen et al. (2018), Theorem 1 implies that the greedily selected set of paths will achieve reward close to the optimum.

4 Examples and Applications

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

4.1 Uniform Matroid

Given a positive integer K, the independent sets of a uniform matroid are all subsets of the ground set $\mathcal{X}$ 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.

4.2 Linear Matroid

Given a function $\phi : \mathcal{X} \to \{0, 1\}^M$, the independent sets of a linear matroid induced by ϕ 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_ρ be the smallest index corresponding to a subset which path ρ 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_ρth canonical basis vector in $\mathbb{R}^M$. Requiring the solution to be an independent set of the binary matroid induced by ϕ and with ground set $\mathcal{X}(p_s, \omega)$, enforces the desired diversity of focus.

4.3 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 traffic constraints: Suppose we have traffic constraints which require that no more than K_m robots may visit subgraph $\mathcal{G}_m = (\mathcal{V}_m, \mathcal{E}_m)$ for $m = 1, \ldots, M$ with $v_s, v_t \in \mathcal{V}_m, \mathcal{V}_m \subseteq \mathcal{V}, \mathcal{E}_m \subseteq \mathcal{E}$. Let $\mathcal{X}_m$ correspond to the set of feasible paths in sub-graph $\mathcal{G}_m$. Requiring that the solution is an independent set of a transversal matroid induced by K_m copies of the sets $\mathcal{X}_m$ enforces the traffic constraint.

Application to risk constraints: Suppose we have M survival probability 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 probability 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.

4.4 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.

Note that this form of a gammoid is also known as a *laminar matroid*.

4.5 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.

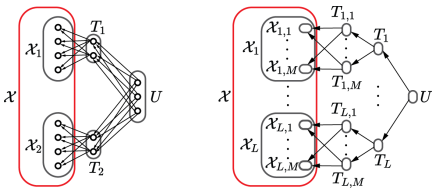

Figure 3. Illustration of multi-partite graphs which form gammoids. The nodes in subgraph $\mathcal{X}$ are *linked* to the nodes U. Left: An illustration of the graph with two layers of cardinality constraints. Right: An illustration of the graph with three layers. Boxes represent clusters of nodes, and lines represent edges which connect each node of the right cluster to each node of the left cluster.

5 Algorithms

In this section we give greedy and accelerated continuous greedy approaches to solving the MTSO problem and give bounds on their suboptimality. Both algorithms solve single-robot sub-problems where a path from the feasible set is sought which approximately maximizes a path-dependent reward function. Accordingly we start by describing this sub-problem in detail in Section 5.1, which shows how to solve it as a series of orienteering problems, provides guarantees on the path quality, and a detailed discussion of implementation considerations for the examples given in Section 4. We then describe the greedy and accelerated continuous greedy approaches in Sections 5.2 and 5.3, respectively. Both of these sections describe how to form the appropriate sub-problem, the complete algorithm, performance guarantees, and computational complexity.

5.1 The Greedy Sub-Problem

In this section we discuss the greedy sub-problem, giving a description of our solution approach in Section 5.1.1, discuss performance considerations in Sections 5.1.2, and its applications in Section 5.1.3.

5.1.1 Objective and Algorithm Given a (possibly empty) previously selected set of paths, $X_{L-1} = \{\rho_\ell\}_{\ell=1}^{L-1}$, the greedy sub-problem involves finding a path from the set $\mathcal{X}_F(X_{L-1}, \mathcal{I})$ which maximizes the sum of path-dependent node rewards $\nu(\rho, j) \geq 0$ (corresponding to the discrete derivative of the appropriate objective function). Solving the greedy sub-problem is a critical step in both of our solution algorithms for the MTSO, but it is a very difficult task as it requires maximizing *path dependent* node rewards subject to a budget constraint. Specifically, the path dependence is induced by the risky traversal model, since the reward gained for visiting node j depends on the probability that the robot successfully visits node j, which in turn depends on the path taken to the node. Furthermore, we must ensure that the path chosen is in $\mathcal{X}_F(X_{L-1}, \mathcal{I})$.

We address the path dependence by using a path *independent* approximation, $\hat{\nu}(j)$, of $\nu(\rho, j)$ which satisfies, for some $\gamma \in (0, 1]$:

$$\nu(\rho, j) \leq \hat{\nu}(j)\mathbb{I}_j(\rho) \leq \frac{1}{\gamma}\nu(\rho, j).$$

We address the matroid constraint by noting that 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 $\mathcal{G}_\rho$ which has exactly the nodes and edges in ρ. 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 $\mathcal{G}_\rho$ contains only one feasible path, ρ. In practice, we can often partition $\mathcal{X}_F(X_{L-1}, \mathcal{I})$ using a very small number of sub-graphs, as detailed in Section 5.1.3.

Algorithm 2 Pseudocode for solving the greedy sub-problem

procedure SOLVESUBPROBLEM($X_{L-1}, \hat{\nu}$)
  for $\mathcal{G}_m$ in `Partition`(X_{L-1}) **do**
    Form $\mathcal{G}_{m,O}$ from $\mathcal{G}_m$
    $\hat{\rho}_m \leftarrow$ `Orienteering`$(\mathcal{G}_{m,O}, \hat{\nu})$
  end for
  return $\underset{\rho \in \hat{\rho}_1, \ldots, \hat{\rho}_M}{\arg\max} \sum_{j=1}^V \hat{\nu}(j)\mathbb{I}_j(\rho)$
end procedure

Our approach to solving the sub-problem is given as pseudocode in Algorithm 2. We begin by calling the `Partition` routine, which returns a list of sub-graphs $\{\mathcal{G}_m\}_{m=1}^M$ which partitions the feasible set $\mathcal{X}_F(X_{L-1}, \mathcal{I})$. For each sub-graph, we form a log-transformed graph $\mathcal{G}_{m,O}$ which has edge weights $\omega_o = -\log(\omega)$. This graph and the approximate node reward $\hat{\nu}$ are passed to an `Orienteering` oracle, which finds a path which has at least fraction $1/\lambda$ the optimal reward with edge costs at most $B = -\log(p_s)$ (we discuss this oracle in depth in Section 5.1.2). This ensures that the path is of high quality and satisfies the survival probability threshold constraint. Finally the routine returns the best of the paths computed (with respect to either $\hat{\nu}$ or ν, depending how hard ν is to compute). The following lemma guarantees that the path returned has similar value as the optimum:

Lemma 1. Sub-problem guarantee. *Consider an MTSO problem with node weights d_j, let ρ be a feasible path in $\mathcal{X}_F(X_{L-1}, \mathcal{I})$, and $\hat{\rho}_L$ be the path output by the* `SolveSubproblem` *routine described in Section 5.1.1. Then*

$$\sum_{j=1}^V d_j \nu(\hat{\rho}_L, j) \geq \frac{\gamma}{\lambda} \sum_{j=1}^V d_j \nu(\rho, j).$$

Proof. By definition of the partition $\{\mathcal{X}_m\}_{m=1}^M$, for any $\rho \in \mathcal{X}_F(X_{L-1}, \mathcal{I})$ there is a set indexed by m_ρ such that $\rho \in \mathcal{X}_{m_\rho}$. We have from the definitions of $\hat{\rho}_L$, $\hat{\nu}(\rho, j)$, and the `Orienteering` routine:

$$\sum_{j=1}^V d_j \nu(\rho, j) \leq \sum_{j=1}^V d_j \mathbb{I}_j(\rho)\hat{\nu}(j) \leq \lambda \sum_{j=1}^V d_j \mathbb{I}_j(\hat{\rho}_L)\hat{\nu}(j)$$
$$\leq \lambda \left(\frac{1}{\gamma} \sum_{j=1}^V d_j \nu(\hat{\rho}_L, j) \right).$$

□

Suppose the computational complexity of the `Orienteering` routine given any sub-graph is bounded by C_O, the complexity of the `Partition` routine is bounded by C_P, and the `Partition` routine returns at most M sub-graphs. Then, the complexity of the `SolveSubproblem` routine is $C_P + MC_O$. Each of the M orienteering problems can be solved independently, so by leveraging parallel computation the complexity will scale as $O(C_P + C_O)$ (assuming M is reasonably small). We describe each of these terms in more detail below.

5.1.2 Complexity of the Orienteering Problem Although the orienteering problem is NP-hard, various communities of researchers have developed solution approaches focusing on different properties of the solution - performance guarantees, on-line bounds, and run time. Depending on the oracle routine used, the `SolveSubproblem` routine has different guarantees and properties, namely:

Complexity theory guarantees – From a theoretical standpoint, if a polynomial-time approximation scheme (PTAS) for the orienteering problem is used and the partition routine is polynmomial time, then our algorithm is a PTAS for the greedy sub-problem. Since our higher level routines call `SolveSubproblem` a polynomial number of times, this is a meaningful result on the complexity of the MTSO problem: although the MTSO is NP-hard, it can be approximated within a constant factor in polynomial time when there is an efficient partition routine. The complexity of the best known PTAS routines for the orienteering problem and its variants are high order polynomials - for example Chen and Har-Peled (2006) gives a $\lambda = 1 + \epsilon$ PTAS for the case where points are on a $d-$dimensional plane that runs in $O(V^{16d^{3/2}/\epsilon})$ time. Even for $\epsilon = 1$ and $d = 2$, this is $O(V^{46})$, which is not a practically useful bound.

Certifiable performance guarantees – Practitioners who require guarantees on the quality of the solution and more modeling flexibility can use mixed integer linear programming (MILP) formulations of the orienteering problem (e.g., Kara et al. (2016)). Commercial and open source software for solving MILP problems are readily available, and return an optimality gap along with the solution. Such solvers can be configured to terminate after a set amount of time or when the ratio between the current solution and upper bound becomes greater than $1/\lambda$.

Real-time properties – Finally, practitioners who require fast execution but not guarantees can use a heuristic to solve the orienteering problem. There are a number of fast, high quality heuristics with open source implementations such as Wagner and Affenzeller (2005); Vansteenwegen et al. (2009). While these heuristics do not provide guarantees, they often produce near-optimal solutions and are capable of solving large problems in seconds.

5.1.3 Efficiently Partitioning the Feasible Set For many examples we can partition the feasible set using a small number of sub-graphs, as detailed below.

Coverage – We can partition the sets for the coverage example by adding a constraint to a mixed integer formulation which requires the solution to have a specific focus. In this case M would be the number of regions to be covered.

Launch constraints – 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 $\mathcal{E}' = \mathcal{E} \setminus \{(v_s, \rho_\ell(1))\}_{\ell=1}^{k}$.

Heterogeneous teams – 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.

Traffic constraints – 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 assignment returns all subgraphs $\mathcal{G}_m$ which had a valid assignment. The complexity of each assignment is $O(K^{2.5})$ (using the Hopcraft-Korp algorithm Hopcroft and Karp (1971)), and there are at most MK assignments, giving complexity $O(MK^{3.5})$.

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 node set $V_{\hat{m}} := \{j \in \{1, \ldots, V\} \mid \zeta_j \geq p_s^{\hat{m}}\}$, which are all nodes reachable within the desired survival probability threshold. Note that in this case we also must change the budget used by the `Orienteering` routine to $-\log(p_s^{\hat{m}})$.

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 traffic example. 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.

5.1.4 Lazy implementation Each time a sub-problem is solved, we can use Theorem 1 to give a bound on the value of any path in the corresponding sub-graph. Since the value of a path is a decreasing function as the algorithm progresses (due to submodularity), these bounds can be used in the following iterations. The *lazy* variant of the greedy algorithm solves the most promising sub-problem to get an initial solution, and then only solves sub-problems which have bounds that exceed the value of this initial solution. In our experiments this allowed us to skip solving approximately 75% of the sub-problems.

5.2 The Approximate Greedy Algorithm

In this section we describe a greedy algorithm for solving the MTSO problem. We describe the objective function and detailed algorithm in Section 5.2.1, then give performance guarantees in Section 5.2.2, and characterize the complexity in Section 5.2.3.

5.2.1 Objective and Algorithm The greedy algorithm operates by iteratively selecting a path which maximizes the discrete derivative of the objective function for the MTSO given the set of previously chosen paths $X_{L-1} \subset \mathcal{X}$. A node-wise decomposition of this objective was given in Jorgensen et al. (2018) as,

$$\Delta J(\rho \mid X_{L-1}) = \sum_{j=1}^{V} d_j \mathbb{E}[z_j^L(\rho)] p_j(0, X_{L-1}),$$

which can be interpreted as the weighted sum over the probabilities that robot L is the first to visit node j. In the context of Lemma 1 we have $\nu(\rho, j) = \mathbb{E}[z_j^L(\rho)] p_j(0, X_{L-1})$, which can be approximated by setting $\hat{\nu}(j) = p_j(0, X_{L-1})$, with $\gamma = p_s$. Pseudocode for the approximate greedy algorithm is given in Algorithm 3. The algorithm alternates between updating the approximate node rewards, $\hat{\nu}$, and calling the `SolveSubproblem` routine to find an approximately optimal solution to the greedy sub-problem.

Algorithm 3 Approximate greedy algorithm for solving the MTSO problem.

procedure MGREEDYSURVIVORS($\mathcal{G}, \mathcal{M}$)
  for $j = 1, \ldots, V$ **do**
    $\hat{\nu}_1(j) \leftarrow d_j$
  end for
  $\rho_1 \leftarrow$ `SolveSubproblem`$(\emptyset, \hat{\nu}_1)$
  for $k = 1, \ldots, K-1$ **do**
    $\mathbb{E}[a_0^k(\rho_k)] \leftarrow 1$
    for $n = 1, \ldots, |\rho_k|$ **do**
      $\mathbb{E}[a_n^k(\rho_k)] \leftarrow \mathbb{E}[a_{n-1}^k(\rho_k)] \omega(e_{\rho_k}^n)$
      $\hat{\nu}_{k+1}(\rho_k(n)) \leftarrow (1 - \mathbb{E}[a_n^k(\rho_k)]) \nu_k(\rho_k(n))$
    end for
    $\rho_{k+1} \leftarrow$ `SolveSubproblem`$(\{\rho_\ell\}_{\ell=1}^{k}, \hat{\nu}_{k+1})$
  end for
end procedure

5.2.2 Guarantees We combine the results from Section 2.3 and 5.1.1 to guarantee that the value of the output of the `MGreedySurvivors` algorithm is close to the value of the optimal solution to the MTSO problem.

Theorem 2. Approximate greedy guarantee. *Let X^* be an optimal solution to the MTSO problem, and $\hat{X}_K$ the set output by the `MGreedySurvivors` routine. If the survival probability constraint is p_s and each orienteering sub-problem is solved within constant factor $1/\lambda$, then the weighted expected number of nodes visited by a team of robots following the paths $\hat{X}_K$ is at least a fraction $\frac{p_s}{p_s+\lambda}$ of the optimum, that is:*

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

Proof. Setting $\nu(\rho, j) = \mathbb{E}[z_j^L(\rho)]p_j(0, X_{L-1})$ and $\hat{\nu}(j) = p_j(0, X_{L-1})$, we have from Lemma 1 that the `SolveSubproblem` routine solves the sub-problems within factor $\alpha = p_s/\lambda$. Now, applying Theorem 1 to our objective function (which by Lemma 2 of Jorgensen et al. (2018) is normalized, non-negative, monotone and submodular) we have the desired result. □

In many scenarios robots are either valuable or difficult to replace, and so operators will choose survival probability thresholds p_s which are close to 1. Combined with an effective oracle routine, Theorem 2 guarantees that the greedy approach never collects less than approximately half of the optimal reward. This is a strong statement, since solving this problem optimally is extraordinarily difficult (especially as the team size grows).

5.2.3 Complexity The `MGreedySurvivors` routine has complexity $O(KV^2 + K(MC_O + C_P))$, where the first term is the complexity of updating the V node weights K times (each update costs at most V multiplications), and the second term is the complexity of calling the `SolveSubproblem` routine K times. In typical scenarios, the complexity will be dominated by KMC_O, as solving the orienteering problem is typically several orders of magnitude more difficult than partitioning the feasible set or updating the weights.

5.3 The Accelerated Continuous Greedy Algorithm

In this section we describe an accelerated continuous greedy algorithm for solving the MTSO problem. We begin by describing the objective function and detailed algorithm in Section 5.3.1, then give performance guarantees in Section 5.3.2 and characterize the complexity in Section 5.3.3.

5.3.1 Objective and Algorithm The accelerated continuous greedy algorithm Badanidiyuru and Vondrák (2014) performs a discretized coordinate gradient ascent which optimizes the multilinear extension along 'coordinates' corresponding to independent sets in the matroid. The state of the algorithm is given by a sparse vector $y \in [0,1]^{|\mathcal{X}|}$ which can be interpreted as a probability distribution over elements in $\mathcal{X}$. The algorithm runs for $1/\delta \in \mathbb{Z}_+$ steps (indexed by i), each consisting of K iterations (indexed by ℓ). During every iteration a single component of y corresponding to a path $\rho \in \mathcal{X}$ is incremented by δ, that is $y \leftarrow y + \delta \mathbf{1}_\rho$ where $\mathbf{1}_\rho$ is the indicator vector which has all zero entries except the component corresponding to ρ, which is 1. The component ρ is selected to ensure that (1) the paths corresponding to the set of components updated in a given step are independent, and (2) the component approximately maximizes $F(y + \delta \mathbf{1}_\rho) - F(y)$, which is equivalent to maximizing the discretized gradient with discretization step δ.

The objective for the ACGA can be written as an expectation with respect to the random set $R(y)$, which contains elements in $\mathcal{X}$ sampled independently with probability y_ρ, that is:

$$\mathbb{P}\{R(y) = X\} = \prod_{x \in X} y_x \prod_{x' \in X^c} (1 - y_{x'}).$$

The following lemma gives an alternate expression for the objective in terms of a sum over path dependent node rewards:

Lemma 2. Objective function for the ACGA. *Let f be the objective of the MTSO and F its multilinear extension. The objective for the ACGA with state y during the ℓth iteration of the ith step can be written as*

$$F(y + \delta \mathbf{1}_\rho) - F(y) = \delta \sum_{j=1}^{V} \frac{d_j \mathbb{E}[z_j^\ell(\rho)]}{1 - y_\rho \mathbb{E}[z_j^\ell(\rho)]} \mathbb{E}[p_j(0, R(y))].$$

This result is derived directly from the definitions of the relevant variables (a detailed proof is given in the Appendix).

Pseudocode for our algorithm is given in Algorithm 4, which consists of three main parts: (1) efficiently computing $\mathbb{E}[p_j(0, R(y))]$, (2) removing path dependence for the sub-problem, and (3) rounding the solution. We discuss each part in detail below.

1) Efficiently computing $\mathbb{E}[p_j(0, R(y))]$: In general, computing an expectation over functions of the random set $R(y)$ is difficult because the function must be evaluated for each of the $2^{K/\delta}$ realizations of $R(y)$. In our case, we exploit the product form of $p_j(0, R(y))$ to efficiently compute the expected value for fixed sizes of $R(y)$, that is $\mathbb{E}[p_j(0, R(y))\mathbb{I}\{|R(y)| = m\}]$, which are then summed to compute the desired expectation. The algorithm is given in pseudocode in Algorithm 5 and details on its correctness are given in the Appendix. The complexity of this approach is $O(VK^2\delta^{-2})$.

2) Removing path dependence for the sub-problem: The objective is path dependent because the term $\frac{\mathbb{E}[z_j(\rho)]}{1 - y_\rho \mathbb{E}[z_j(\rho)]}$ depends on the probability that node j is visited (which is a function of the path taken) and the weight assigned to path ρ. We handle this term in two stages: we start by assuming that $y_\rho = 0$ (which is true for most paths) in which case we have $\frac{\mathbb{E}[z_j(\rho)]}{1 - y_\rho \mathbb{E}[z_j(\rho)]} = \mathbb{E}[z_j(\rho)]$ and use the strategies developed in Section 5.2.1 to approximately maximize the objective by solving an orienteering problem. We then compute the value of the $O(K/\delta)$ paths which have $y_\rho > 0$ (corresponding to the solutions to the greedy subproblem in previous iterations) explicitly and select the better of these paths or the output of the orienteering routine solved in the first stage. We show in the Appendix that this two-stage approach produces a path with value within factor p_s/λ of the optimal.

3) Rounding the solution: We use the `SwapRounding` procedure to round y in order to get our final solution. In order to ensure that we achieve a good result we repeat the rounding procedure until the result is at least $(1 - \delta)F(y)$. The expected number of calls to the rounding routine is upper bounded by $1/(1 - e^{-p_s\delta^2/8})$, which is $O(\delta^{-2})$ (we defer details of these calculations to the Appendix).

5.3.2 Guarantees Combining our results from Sections 2.3, 5.1.1, and 5.3.1 we provide the following performance guarantee for the `MCGreedySurvivors` routine:

Theorem 3. Suboptimality bound. *Let X^* be an optimal solution to the MTSO problem and $\hat{X}$ be the output of the `MCGreedySurvivors` routine with parameters δ and λ. Then the value of the set $\hat{X}$ is lower bounded by a constant*

Algorithm 4 Approximate continuous greedy algorithm for solving the MTSO problem.

procedure MCGreedySurvivors ($\mathcal{G}, \mathcal{M}, \delta$)
  $y \leftarrow \vec{0}$
  for $i = 1, \ldots, 1/\delta$ **do**
    $X(i) \leftarrow \emptyset$
    for $\ell = 1, \ldots, K$ **do**
      $\hat{\nu} \leftarrow$ UpdateWeights(y)
      $\rho' \leftarrow$ SolveSubproblem($X(i), \hat{\nu}$)
      $\rho \leftarrow \underset{\rho', \{\rho \in \mathcal{X}_F(X(i), \mathcal{I}) : y_\rho > 0\}}{\arg\max} \sum_{j=1}^{V} d_j \nu(\rho, j)$
      $y_\rho \leftarrow y_\rho + \delta$
      $X(i) \leftarrow X(i) \cup \rho$
    end for
  end for
  while True **do**
    $\hat{X} \leftarrow$ SwapRounding(y)
    if $f(\hat{X}) \geq (1-\delta)F(y)$ **then return** $\hat{X}$
    end if
  end while
end procedure

Algorithm 5 Efficient weight update routine.

procedure UpdateWeights(y)
  for $j = 1, \ldots, V$ **do**
    $w = \vec{0}_{|y|+1}$
    $w_0 \leftarrow 1$
    for $x : y_x > 0$ **do**
      for $m = |y|, \ldots, 1$ **do**
        $w_m \leftarrow w_{m-1} \cdot y_x \cdot (1 - \mathbb{E}[z_j(x)]) + w_m \cdot (1 - y_x)$
      end for
      $w_0 \leftarrow w_0 \cdot (1 - y_x)$
    end for
    $\hat{\nu}_j \leftarrow \sum_{m=0}^{|y|} w_m$
  end for
end procedure

factor of the optimum:

$$f(\hat{X}) \geq f(X^*)(1-\delta)\left(1 - \exp\left(-\frac{p_s}{\lambda + \delta p_s}\right)\right).$$

The detailed proof is given in the Appendix. The basic idea is to first use the properties of the SolveSubproblem routine to lower bound the incremental increase in the multilinear extension between subsequent steps of the algorithm, then to use a recursive argument to show that $F(y)$ satisfies the desired inequality, which in turn ensures that $f(\hat{X})$ satisfies the guarantee.

Tightness of the guarantee – As the approximation parameters converge to their ideals ($\delta \to 0, \lambda \to 1, p_s \to 1$), the guarantee converges to $f(\hat{X}) \geq f(X^*)(1 - e^{-1})$, which matches the hardness bounds for the general problem of maximizing a submodular function subject to a matroid constraint. We also note that for $\delta \ll \lambda$, the guarantee is approximately $1 - e^{-p_s/\lambda}$, which matches the guarantee given for the TSO problem Jorgensen et al. (2018).

5.3.3 Complexity The MCGreedySurvivors routine calls the UpdateWeights routine K/δ times, solves K/δ sub-problems, and calls the SwapRounding routine an average of $O(\delta^{-2})$ times. Hence the total complexity is

$$O(\delta^{-1}K(MC_O + C_P) + \delta^{-3}K^3V + \delta^{-3}K),$$

and since generally $C_O \gg \delta^{-2}K^2V$, this is $O(\delta^{-1}KMC_O)$, which is a factor of δ^{-1} greater than the approximate greedy routine.

6 Experimental Results

In this section we validate and compare our solution approaches using simulations. In particular, we demonstrate the rich sets of constraints a matroid can model in Section 6.1, where we consider a challenging environmental monitoring scenario and compare two choices for the Orienteering oracle. Section 6.2 quantifies the empirical scaling of our algorithm using synthetic data. In particular, section 6.2.1 shows that the MGreedySurvivors algorithm scales as expected as the team size and number of sub-problems required to partition the feasible set grows, and Section 6.2.2 shows that the ACGA algorithm scales as expected as δ shrinks. Finally, in Section 6.3 we compare the MGreedySurvivors and ACGA approaches for a range of δ values and discuss the strengths of each approach.

6.1 Environmental Monitoring Application

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 International Chamber of Commerce: Commercial Crime Services (2017) and model attacks as a Poisson random variable in a manner similar to Vaněk et al. (2013). We selected 106 marine protected areas from Cros et al. (2014) 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 survival probability threshold $p_s = 0.64$, 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 $p_s^6, \ldots, p_s^{25} = 0.859$, that is we allow five robots to take more risky paths and constrain the rest to more conservative paths. All of the constraints above can be represented using a single gammoid, and the feasible set can be partitioned using at most $M = 15$ sub-graphs. Note how expressive gammoids are - this example uses just three levels of 'nested constraints' (robot types, traffic limits, and risk constraints) but we could easily use more.

We consider two choices for the Orienteering routine: an open-source Variable Neighborhood Search (VNS) heuristic produced by HeuristicLab Wagner and Affenzeller

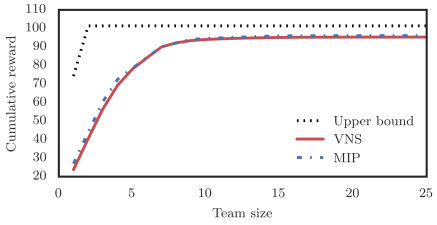

Figure 4. Cumulative reward for paths planned by `MGreedySurvivors` using the MIP and VNS `Orienteering` routines. Both approaches compute high quality sets of paths, though VNS is somewhat faster.

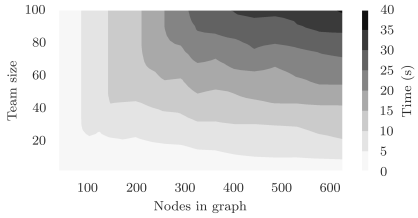

Figure 5. Scaling of `MGreedySurvivors` as K and V grow. Data shown is the median of 110 samples, and agrees with an $\Theta(MKC_O)$ trend.

(2005) and a mixed-integer linear program (MILP) formulation solved using CPLEX with a 5% tolerance. We observe that the VNS heuristic gives near-optimal paths, and solves all of the sub-problems 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. Optimally solving the sub-problems does not necessarily lead to better overall behavior, as shown in Figure 4. The upper bound shown is computed as the smaller of 1) the upper bound found using the greedily selected paths and the approximation ratio, and 2) the sum of all node rewards $\sum_{j=1}^{V} d_j = 106$. This upper bound is loose because it ignores the constraints (particularly on the traffic flow through a region) which becomes apparent as the team size grows larger. Note that although the theoretical approximation guarantee is 0.36, a team of 10 robots achieves 0.928 the maximum possible reward, despite only planning one path at a time.

6.2 Synthetic problem generation

We demonstrate the scaling properties of the `MGreedySurvivors` and ACGA algorithms using synthetic data. Nodes are placed randomly on a 2D plane and clustered into groups of 15-20 nodes (this ensures the complexity of solving the orienteering problem, C_O, is constant). We consider a traffic constraint with sub-graphs induced by the clusters. Edge weights are chosen so that $-\log(\omega(e))$ is proportional to the length of the edge e, the survival probability threshold is set to 0.7, and we use a MILP formulation for the planar orienteering problem with tolerance 5%.

6.2.1 Sub-problem complexity scaling We demonstrate the complexity of the `MGreedySurvivors` algorithm with respect to K and M using the synthetic dataset described above. We use the lazy variant of the greedy algorithm Krause and Golovin (2014), which only solves a sub-problem if it is not dominated by another (this allows us to skip 76% of the sub-problems in our experiment). The median computation times are shown in Figure 5. The trends agree with the $O(MKC_O)$ scaling predicted in Section 5.2.3, and our approach solves very large problems with very large teams in under a minute when sub-graphs are small.

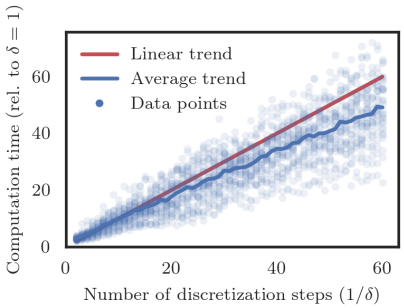

Figure 6. Scaling of the run time of the ACGA routine relative to the `MGreedySurvivors` routine as the number of discretization steps increases. Note that run time increases approximately linearly with δ^{-1}, as predicted in Section 5.3.3.

6.2.2 Discretization complexity scaling As discussed in Section 5.3.3, we expect the run time of the ACGA to be roughly δ^{-1} times that of the `MGreedySurvivors` routine (which is ACGA with $\delta = 1$). We demonstrate this trend empirically by using the synthetic data generated as described at the beginning of this section (with $V = 82$ and $K = 8$). For each problem we compute the baseline by averaging the run time of 5 calls to the `MGreedySurvivors` routine. We then measure the run times of the ACGA with the desired δ values and use the ratio of ACGA run times to the baseline run time as our datapoints. We repeated this for 30 randomly generated problems and show the results in Figure 6. The randomness in run times is due to several factors (1) lazy implementation allows for problems to be skipped when good bounds are available, (2) MIP solver time can depend significantly on the particular problem, and (3) MIP solver times are not deterministic. Note that the average run time (with respect to the random problem instances) actually grows *sub-linearly*, likely due to the lazy implementation of the ACGA.

6.3 Effect of discretization on performance

Using the same data as in Section 6.2.2, we analyze the influence of the discretization parameter on the performance of ACGA relative to `MGreedySurvivors` in terms of the value of their solutions. As shown in Figure 7, increasing δ^{-1}

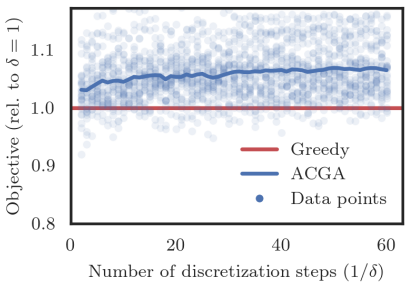

Figure 7. Comparison of objective achieved by the ACGA and `MGreedySurvivors` routines with $\delta^{-1} \in \{2, \ldots, 60\}$ for 30 random MTSO problems. Note that as δ^{-1} increases the ACGA more consistently outperforms the baseline and average performance increases.

generally improves the performance, and generally increases the chance that the result produced by ACGA will be better than the result produced by `MGreedySurvivors`. Note that the benefits of smaller discretization sizes drop off faster than the increase in computation time, so in practice a relatively small value of δ^{-1} (e.g., 8 or 16) should be used. In scenarios where computation time is paramount, the `MGreedySurvivors` routine provides a fast way of achieving high quality solutions.

7 Extensions

In this section we discuss additional extensions and applications of our work. In Section 7.1, we describe p-systems and their applications to robotics. Minor modifications to algorithms presented above have constant factor guarantees for p-systems. In Section 7.2 we describe the coverage variant of the MTSO, which is closely related to our other work in Jorgensen et al. (2017b). Results from Chekuri et al. (2010) can be applied to provide guarantees for our ACGA algorithm in this setting.

7.1 p-system constraints

Consider the MTSO where the matroid $\mathcal{M}$ is replaced by a p-system. Recall that a p-system is an independence system which is downward closed and every base (maximal independent set) has at most p elements. The crucial distinction between p-systems and matroids is that the latter has the exchange property, which implies that *every* base in a matroid has the same number of elements (which is the rank of the matroid).

7.1.1 Applications to robotics Many common constraints in robotics can be modeled as as p-system:

Collision avoidance – Consider a setting where we do not allow two robots to simultaneously traverse the same edge in the graph (this also requires us to introduce a notion of time, which we discuss more in our work Jorgensen et al. (2018)). One can easily verify that collision avoidance is downward closed - any subset of a set of collision free paths will also be collision free. The value of p is easily bounded by the smaller of the out-degree of the starting vertex and the number of robots allowed.

Capacity constraints – Collision avoidance can be generalized to a capacity constraint, where each edge has a capacity, and each robot has a demand. We can require that the sum of demands for robots simultaneously traversing an edge must be below the capacity of the edge. Note that collision avoidance is a special case where all demands and capacities are equal. This scenario can model road networks as well as communications networks, where a video sensor may demand much more bandwidth than other types of sensors.

Matroid intersections – The intersection of p matroids is also a p-system. This can allow us to simultaneously enforce risk constraints, sensor availability, and traffic constraints. Note that this is different from the nested cardinality constraints used in the experiments. For example, we could require that at most 5 of the 25 paths planned are risky and that at most 12 of each robot type can be selected (rather than the nested case, which would say, e.g., at most 12 of the risky paths can be of a given type, and at most 12 of the safe paths can be of a given type).

7.1.2 Algorithms and guarantees We can use a slightly modified version of our algorithms above to find a set which is in the independent family of sets for a p-system and (approximately) maximizes the expected number of nodes visited. The only difference is that now the feasible set $\mathcal{X}_F(X, \mathcal{I})$ is defined with respect to the independent family of sets for a p-system. A similar application-specific partitioning technique can be used in the `SolveSubproblem` routine.

The guarantee we state in Theorem 1 is a special case of the more general statement from Appendix B of Calinescu et al. (2007), which for a p-system gives the constant factor guarantee $\frac{\alpha}{\alpha+p}$. For our applications, we will typically constrain teams to have at most K robots, meaning that the guarantees become $\frac{p_s}{p_s+K\lambda}$, which while a constant factor, quickly becomes loose as the team size grows.

7.2 Multiple objectives and coverage problems

Consider a problem where we are given a set of submodular functions $f_1, \ldots, f_N$, corresponding values $V_1, \ldots, V_N$, and want to find a set X such that $f_n(X) \geq V_n$ for $n = 1, \ldots, N$.

7.2.1 Applications to robotics This problem could model a coverage variant of the MTSO, where we are seeking a set X which satisfies a matroid constraint while also ensuring that the probability node j is visited satisfies a given threshold, $p_v(j)$. In this setting we define the functions $f_j(X) = \min\{1 - p_j(0, X), p_v(j)\}$ and set $V_j = p_v(j)$. This is similar to the coverage variant of the TSO given in Jorgensen et al. (2017b), where we seek the smallest team which satisfies the desired visit probabilities.

7.2.2 Algorithm and guarantees The algorithm for this case only requires minor modifications to our proposed algorithms, and would leverage the modifications to the continuous greedy algorithm outlined in Chekuri et al. (2010). Theorem 2.5 from Chekuri et al. (2010) guarantees that the output of the modified algorithm satisfies $F_n(y) \geq (1 - e^{-1})V_n$ for $n = 1, \ldots, N$, or returns a certificate of

infeasibility. In our case the ACGA has a weaker $1 - e^{-p_s/\lambda}$ guarantee, and so the result would be modified accordingly. If the visit probability thresholds are small enough, we can guarantee feasibility by dividing V_j by $(1 - e^{-p_s/\lambda})$, ensuring that the visit probability is greater than $p_v(j)$.

8 Conclusions

Summary We formulate the *Matroid Team Surviving Orienteers* (MTSO) problem, where we seek a set of paths which forms an independent set of a matroid, 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 Jorgensen et al. (2018), 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) which creates a complex, history-dependent coupling between the edges chosen and the distribution of nodes visited. We present two solution algorithms: 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, and a variant of the ACGA which guarantees that its output achieves at least $\simeq 1 - e^{-p_s/\lambda}$ the optimal reward. The algorithms rely on a partitioning routine to satisfy the matroid constraints, and we show numerous examples where our algorithm runs in polynomial time. We demonstrate the efficiency of our approaches by applying it to a scenario where a team of robots must gather information while avoiding pirates in the Coral Triangle, and study how algorithm parameters influence complexity and performance.

Future work There are many directions for future work beyond those outlined in Section 7. The most promising is the extension to linear packing constraints (also called knapsack constraints). The multiplicative weight update (MWU) framework was used by Chekuri et al. (2015) to find a set X which optimizes a submodular function subject to linear packing constraints, $\mathbf{1}_X A \leq \mathbf{1}$, where A is an $M \times |\mathcal{X}|$ matrix. These constraints generalize common cases of matroid constraints, such as the partition, transversal, and laminar matroids, and can represent more general versions of the applications from Section 4 such as constraining the expected number of failures, the total number of types of sensors available, or limits on traffic through each edge in the graph.

The MWU framework solves a cost-benefit greedy variant of the ACGA, where paths are weighted by their cost (i.e., how much of the constraint they use). Specifically, the algorithm increments the coordinate of y which ensures that the coordinates selected in a particular step satisfy the packing constraints, and which maximizes

$$\frac{1}{\sum_{m=1}^{M} w_m A_{m,\rho}}(F(y + \delta \mathbf{1}_\rho) - F(y)),$$

where w_m is a second state vector used for constraint satisfaction.

In principle this is not much more difficult than the ACGA presented in this paper, however in practice this will require clever tailoring of the MILP formulations and careful selection of the constraint function to ensure that it can be solved as an orienteering problem (or that a suitable oracle routine exists). The `Partition` routine will also need to be updated to represent the feasible space correctly. The guarantees for the MWU approach are very similar to those of the ACGA and borrow the same arguments, so in principle we should expect similar results as given in Theorem 3, however appropriately rounding the fractional solution remains an open problem.

A second direction for future work is to investigate notions of abstract *dependence* such as antimatroids, which can model queueing networks or precedence constraints. There are few fundamental results on optimizing with such abstract dependence constraints, but it is an active area of research and there are some recent results for the special case of 'bottleneck functions' which may be useful in robotics.

Acknowledgements

The authors would like to thank Jan Vondrák for helpful discussions about the accelerated continuous greedy algorithm and multilinear extension which started this work. The authors are also grateful to Robert H. Chen and Mark B. Milam for their contributions to the conference version of this work.

References

Atanasov N, Le Ny J, Daniilidis K and Pappas G (2015) Decentralized active information aquisition: Theory and application to multi-robot SLAM. In: *Proc. IEEE Conf. on Robotics and Automation*.

Badanidiyuru A and Vondrák J (2014) Fast algorithms for maximizing submodular functions. In: *ACM-SIAM Symp. on Discrete Algorithms*.

Calinescu G, Chekuri C, Pál M and Vondrák J (2007) Maximizing a submodular set function subject to a matroid constraint. In: *Int. Conf. on Integer Programming and Combinatorial Optimization*.

Campbell AM, Gendreau M and Thomas BW (2011) The orienteering problem with stochastic travel and service times. *Annals of Operations Research* 186(1): 61–81.

Chekuri C, Jayram T and Vondrák J (2015) On multiplicative weight updates for concave and submodular function maximization. In: *Conf. on Innovations in Theoretical Computer Science*.

Chekuri C and Pál M (2005) A recursive greedy algorithm for walks in directed graphs. In: *IEEE Symp. on Foundations of Computer Science*.

Chekuri C, Vondrák J and Zenklusen R (2010) Dependent randomized rounding via exchange properties of combinatorial structures. In: *IEEE Symp. on Foundations of Computer Science*.

Chen K and Har-Peled S (2006) The orienteering problem in the plane revisited. In: *ACM Symp. on Computational Geometry*.

Corah M and Michael N (2017) Efficient online multi-robot exploration via distributed sequential greedy assignment. In: *Robotics: Science and Systems*.

Cros A, Ahamad Fatan N, White A, Teoh S, Tan S, Handayani C, Huang C, Peterson N, Venegas Li R, Siry HY, Fitriana R, Gove J, Acoba T, Knight M, Acosta R, Andrew N and Beare D (2014) The Coral Triangle Atlas: An integrated online spatial database system for improving coral reef management. *PLoS ONE* 9(6): 1–7.

Fisher M, Nemhauser G and Wolsey L (1978) An analysis of approximations for maximizing submodular set functions –II. In: *Polyhedral Combinatorics*. Springer.

Golden BL, Levy L and Vohra R (1987) The orienteering problem. *Naval Research Logistics* 34(3): 307–318.

Gunawan A, Lau H and Vansteenwegen P (2016) Orienteering problem: A survey of recent variants, solution approaches and applications. *European Journal of Operational Research* 255(2): 315–332.

Gupta A, Krishnaswamy R, Nagarajan V and Ravi R (2012) Approximation algorithms for stochastic orienteering. In: *ACM-SIAM Symp. on Discrete Algorithms*.

Hoff A, Andersson H, Christiansen M, Hasle G and Løkketangen A (2010) Industrial aspects and literature survey: Fleet composition and routing. *Computers & Operations Research* 37(12): 2041–2061.

Hopcroft JE and Karp RM (1971) A $n^{5/2}$ algorithm for maximum matchings in bipartite graphs. In: *Proc. of the Symposium on Switching and Automata Theory*.

International Chamber of Commerce: Commercial Crime Services (2017) IMB piracy reporting centre. Available at https://www.icc-ccs.org/piracy-reporting-centre.

Jawaid S and Smith S (2015) Informative path planning as a maximum travelling salesman problem with submodular rewards. *Discrete Applied Mathematics* 186: 112–127.

Jorgensen S, Chen R, Milam M and Pavone M (2017a) The matroid team surviving orienteers problem: Constrained routing of heterogeneous teams with risky traversal. In: *IEEE/RSJ Int. Conf. on Intelligent Robots & Systems*.

Jorgensen S, Chen R, Milam M and Pavone M (2017b) The risk-sensitive coverage problem: Multi-robot routing under uncertainty with service level and survival constraints. In: *Proc. IEEE Conf. on Decision and Control*.

Jorgensen S, Chen R, Milam M and Pavone M (2018) The team surviving orienteers problem: Routing teams of robots in uncertain environments with survival constraints. *Autonomous Robots* 42(4): 927–952.

Kara I, Biçakci P and Derya T (2016) New formulations for the orienteering problem. *Procedia Economics and Finance* 39: 849–854.

Karaman S and Frazzoli E (2011) Sampling-based algorithms for optimal motion planning. *Int. Journal of Robotics Research* 30(7): 846–894.

Kelareva E, Tierney K and Kilby P (2014) CP methods for scheduling and routing with time-dependent task costs. *EURO Journal on Computational Optimization* 2(3): 147–194.

Koç Ç, Bektaş T, Jabali O and Laporte G (2016) Thirty years of heterogeneous vehicle routing. *European Journal of Operational Research* 249(1): 1–21.

Krause A and Golovin D (2014) Submodular function maximization. In: *Tractability: Practical approaches to hard problems*. Cambridge Univ. Press.

Lahyani R, Khemakhem M and Semet F (2015) Rich vehicle routing problems: From a taxonomy to a definition. *European Journal of Operational Research* 241(1): 1–14.

Murota K (2009) *Matrices and Matroids for Systems Analysis*. 1 edition. Springer Science & Business Media.

Psaraftis HN, Wen M and Kontovas CA (2016) Dynamic vehicle routing problems: Three decades and counting. *Networks* 67(1): 3–31.

Schrijver A (2002) *Combinatorial optimization: polyhedra and efficiency*. 1 edition. Springer Science & Business Media.

Segui-Gasco P, Shin H, Tsourdos A and Seguí V (2015) Decentralized submodular multi-robot task allocation. In: *IEEE/RSJ Int. Conf. on Intelligent Robots & Systems*.

Singh A, Krause A, Guestrin C and Kaiser WJ (2009) Efficient informative sensing using multiple robots. *Journal of Artificial Intelligence Research* 34: 707–755.

Smith S, Tůmová J, Belta C and Rus D (2011) Optimal path planning for surveillance with temporal-logic constraints. *Int. Journal of Robotics Research* 30(14): 1695–1708.

Ulusoy A, Smith S and Belta C (2014) Optimal multi-robot path planning with ltl constraints: Guaranteeing correctness through synchronization. In: *Int. Symp. on Distributed Autonomous Robotic Systems*.

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

Vansteenwegen P, Souffriau W, Berghe GV and Van Oudheusden D (2009) Iterated local search for the team orienteering problem with time windows. *Computers & Operations Research* 36(12): 3281–3290.

Varakantham P and Kumar A (2013) Optimization approaches for solving chance constrained stocahstic orienteering problems. In: *Proc. Int. Conf. on Algorithmic Decision Theory*.

Wagner S and Affenzeller M (2005) HeuristicLab: A generic and extensible optimization environment. In: *Adaptive and Natural Computing Algorithms*. Springer.

Williams R, Gasparri A and Ulivi G (2017) Decentralized matroid optimization for topology constraints in multi-robot allocation problems. In: *Proc. IEEE Conf. on Robotics and Automation*.

Zhang H and Vorobeychik Y (2016) Submodular optimization with routing constraints. In: *Proc. AAAI Conf. on Artificial Intelligence*.

Appendix

Derivation of the objective function for the ACGA

In this section we derive the equivalent form of the objective function for the accelerated continuous greedy algorithm given in Lemma 2, that is:

$$F(y+\delta\mathbf{1}_\rho)-F(y)=\delta\sum_{j=1}^{V}\frac{d_j\mathbb{E}[z_j^\ell(\rho)]}{1-y_\rho\mathbb{E}[z_j^\ell(\rho)]}\mathbb{E}[p_j(0,R(y))].$$

Since the algorithm considers an element at most $1/\delta$ times and increments a selected element by δ, at any point in the algorithm $y_\rho < 1$ if ρ is a candidate solution to the greedy sub-problem. We begin by giving three useful identities about the distribution of $R(y)$, then express the objective function in terms of $\mathbb{E}[\Delta f(\rho \mid R(y))]$ (a useful result for subsequent proofs), and finally derive the statement given in Lemma 2.

1. Let $P_y(X)$ be the probability that $R(y) = X$. Then
$$P_{y+\delta \mathbf{1}_\rho}(X) = P_y(X)\left(1 + \delta\left(\frac{\mathbb{I}\{\rho \in X\}}{y_\rho} - \frac{\mathbb{I}\{\rho \notin X\}}{1 - y_\rho}\right)\right), \tag{1}$$
which follows directly from the definition of R.
2. For any X, ρ,
$$P_y(X \cup \rho) = P_y(X \setminus \rho)\frac{y_\rho}{1 - y_\rho}. \tag{2}$$
3. By definition of p_j, we have for all X, ρ:
$$p_j(0, X \cup \rho) = p_j(0, X \setminus \rho)(1 - \mathbb{E}[z_j(\rho)]),$$
which along with Equation 2 gives
$$p_j(0, X \setminus \rho)P_y(X \setminus \rho) + p_j(0, X \cup \rho)P_y(X \cup \rho) = p_j(0, X \setminus \rho)P_y(X \setminus \rho)\left(1 + \frac{y_\rho(1 - \mathbb{E}[z_j(\rho)])}{1 - y_\rho}\right),$$
which implies the third identity:
$$\mathbb{E}[p_j(0, R(y))] = \frac{1 - y_\rho \mathbb{E}[z_j(\rho)]}{1 - y_\rho} \sum_{X \subseteq \mathcal{X} \setminus \rho} p_j(0, X)P_y(X). \tag{3}$$

Now using these identities along with the definition of F, one obtains

$$\begin{aligned} F(y + \delta \mathbf{1}_\rho) - F(y) &= \sum_{X \subseteq \mathcal{X}} f(x)\left(P_{y+\delta}(X) - P_y(X)\right) \\ &= \sum_{X \subseteq \mathcal{X}} \delta f(X)P_y(X)\left(\frac{\mathbb{I}\{\rho \in X\}}{y_\rho} - \frac{\mathbb{I}\{\rho \notin X\}}{1 - y_\rho}\right) \\ &= \delta \sum_{X \subseteq \mathcal{X} \setminus \rho} P_y(X)\left(\frac{f(X \cup \rho)}{y_\rho}\frac{y_\rho}{1 - y_\rho} - \frac{f(X)}{1 - y_\rho}\right) \\ &= \frac{\delta}{1 - y_\rho} \sum_{X \subseteq \mathcal{X} \setminus \rho} \Delta f(\rho \mid X)P_y(X). \end{aligned}$$

The first line is by definition of F, the second is obtained using Equation 1, the third is from Equation 2, and the last is by definition of Δf.

This result combined with Equation 3 and the fact that $\Delta f(\rho \mid X \cup \rho) = 0$ gives the two desired results:

$$F(y + \delta \mathbf{1}_\rho) - F(y) = \frac{\delta}{1 - y_\rho}\mathbb{E}[\Delta f(\rho \mid R(y))] \tag{4}$$
$$= \frac{\delta}{1 - y_\rho}\sum_{j=1}^{V} d_j \mathbb{E}[z_j(\rho)] \sum_{X \subseteq \mathcal{X} \setminus \rho} p_j(0, X)P_y(X) \tag{5}$$
$$= \delta \sum_{j=1}^{V} d_j \frac{\mathbb{E}[z_j(\rho)]}{1 - y_\rho \mathbb{E}[z_j(\rho)]}\mathbb{E}[p_j(0, R(y))]. \tag{6}$$

The second equality is by definition of f and the third is from applying Equation 3. □

Performance guarantee for the ACGA

In this section we provide the proof for the statement of Theorem 3, which we repeat below:

Let X^* be an optimal solution to the MTSO problem and $\hat{X}$ be the output of the `MCGreedySurvivors` *routine with parameters δ and λ. Then the value of the set $\hat{X}$ is lower bounded by a constant factor of the optimum:*

$$f(\hat{X}) \geq f(X^*)(1 - \delta)\left(1 - \exp\left(-\frac{p_s}{\lambda + \delta p_s}\right)\right).$$

Outline of the argument – The proof follows the argument of Badanidiyuru and Vondrák (2014) closely. We begin by lower bounding the increase in $F(y)$ between subsequent steps and then use a recursive argument to bound the value of $F(y)$ after the last iteration. Since this section deals primarily with the evolution of y, we denote the state of y after the ℓth iteration of the ith step as $y(i, \ell)$, and use the shorthand $y(i, K) = y(i) = y(i + 1, 0)$. In words, $y(i)$ is shorthand for the state after all K iterations of the ith step are complete; and $y(i + 1, 0)$ is the state during the $i + 1$st step before any iterations are complete (which is the same as $y(i, K)$). We also denote the path selected by the algorithm during the ℓth iteration of the ith step by $\rho_{i,\ell}$.

Note on feasibility – An important consequence of the exchange properties of matroids is that during any step i there is an ordering of the elements in an optimal set $\rho_1^*, \ldots, \rho_K^*$ such that ρ_ℓ^* is a candidate solution when solving the sub-problem during the ℓth iteration. This means we can combine Lemmas 2 and 1 to bound the increase in $F(y)$ between steps.

Note on the sub-problem – When solving the sub-problem, we consider the paths with $y_\rho > 0$ explicitly and solve the greedy sub-problem using the approximation $y_\rho = 0$. Since the node reward is an increasing function of y_ρ, we still have a λ–approximate guarantee for the sub-problem:

$$\rho \in \underset{\rho \in \mathcal{X}_F(X(i), \mathcal{I})}{\arg\max} \sum_{j=1}^{V} d_j \frac{\hat{\nu}(j)}{1 - y_\rho \mathbb{E}[z_j(\rho)]},$$

which means we can apply Lemmas 1 and 2 to guarantee that $\rho_{i,\ell}$ is within a multiplicative factor of p_s/λ of the optimal.

We are now in a position to bound $F(y(i + 1)) - F(y(i))$ as follows,

$$\begin{aligned} F(y(i + 1)) - F(y(i)) &= \sum_{\ell=1}^{K} F(y(i, \ell)) - F(y(i, \ell - 1)) \\ &= \delta \sum_{\ell=1}^{K} \mathbb{E}[\Delta f(\rho_{i,\ell} \mid R(y))] \\ &\geq \delta \frac{p_s}{\lambda} \sum_{\ell=1}^{K} \mathbb{E}[\Delta f(\rho_\ell^* \mid R(y(i, \ell - 1)))] \\ &\geq \frac{\delta p_s}{\lambda}\left(\mathbb{E}[f(X^* \cup R(y(i)))] - \mathbb{E}[f(R(y(i + 1)))]\right) \\ &\geq f(X^*)\frac{\delta p_s}{\lambda} - \frac{\delta p_s}{\lambda}F(y(i + 1)). \end{aligned}$$

The second line is a telescoping sum, the third is from Lemma 1 and the fact that ρ_ℓ^* is feasible, the fourth is due to submodularity and the fifth due to monotonicity.

Now we can rearrange the inequality above to get

$$(F(y(i+1)) - f(X^*))\left(1 + \frac{\delta p_s}{\lambda}\right) \geq F(y(i)) - f(X^*), \quad (7)$$

which applied recursively gives

$$F(y(1/\delta)) \geq f(X^*)\left(1 - \left(1 + \frac{\delta p_s}{\lambda}\right)^{-1/\delta}\right)$$
$$\geq f(X^*)\left(1 - \exp\left(-\frac{p_s}{\lambda + p_s\delta}\right)\right).$$

The last inequality is because $1 + x \leq e^x$ implies $(1 + x)^{-N} \leq e^{-Nx/(1+x)}$. The theorem statement follows since we enforce that the rounded result $\hat{X}$ satisfies $f(\hat{X}) \geq (1 - \delta)F(y(1/\delta))$. □

Correctness for `UpdateWeights` routine

We introduce some new notation to analyze the `UpdateWeights` routine. Given a vector y, let ρ_n be the nth non-zero coordinate of y, and define $R^\ell(y) := R(y) \cap \{\rho_n\}_{n=1}^{\ell}$ as the random set restricted to the first ℓ nonzero coordinates. Now define

$$w_m^\ell := \mathbb{E}[p_j(0, R^\ell(y))\mathbb{I}\{|R^\ell(y)| = m\}],$$

which is the expected probability that none of the paths in $R^\ell(y)$ visit node j and there are m paths in $R^\ell(y)$. If M is the number of non-zero entries of y, then by definition of the expectation we have,

$$\mathbb{E}[p_j(0, R(y))] = \sum_{m=0}^{M} w_m^M,$$

since the number of elements in the set $R(y)$ is between 0 and M.

For $\ell = 0$, the weights w_m^ℓ are zero unless $m = 0$. For $\ell > 0$ we can use the product form of $p_j(0, X)$ to express w_m^ℓ in terms of $w_{m-1}^{\ell-1}$ and $w_m^{\ell-1}$:

$$w_m^\ell = w_{m-1}^{\ell-1}\left((1 - \mathbb{E}[z_j^\ell(\rho_\ell)])y_{\rho_\ell}\right) + w_m^{\ell-1}\left(1 - y_{\rho_\ell}\right).$$

This expression has an intuitive meaning, as it captures the two ways that $R^\ell(y)$ can have m elements: either $y_\ell \in R^\ell(y)$ and $|R^{\ell-1}(y)| = m - 1$, or $y_\ell \notin R^\ell(y)$ and $|R^{\ell-1}(y)| = m$. In both cases we update the weights and probability of the event occurring by multiplying by the appropriate coefficients.

The `UpdateWeights` routine simply applies this iterative approach to compute $\{w_m^M\}_{m=0}^M$ and then sums the weights to find $\mathbb{E}[p_j(0, R(y))]$.

Expected complexity of `SwapRounding`

Let p_f be the probability that `SwapRounding` fails to return a satisfactory result. Then the expected number of calls is

$$\sum_{n=1}^{\infty} n p_f^{n-1}(1 - p_f) = \frac{1}{1 - p_f}.$$

Now using the bound from Chekuri et al. (2010), $p_f \leq \exp(-F(y)\delta^2/8) \leq \exp(-p_s\delta^2/8)$, and from the inequality $e^{-x} \leq 1 - x + x^2/2$ we get $\frac{1}{1-e^{-x}} \leq \frac{2}{2x-x^2}$. Combining these inequalities gives the cited result.

Note that we could make the while loop terminate after δ^{-2} iterations which would make the statement of Theorem 3 hold with probability at least $1 - e^{-p_s/8}$, and the complexity $O(K^2\delta^{-3})$.