Autonomous Robots manuscript No.
(will be inserted by the editor)

The Team Surviving Orienteers Problem

Routing teams of robots in uncertain environments with survival constraints

Stefan Jorgensen · **Robert H. Chen** · **Mark B. Milam** · **Marco Pavone**

Received: date / Accepted: date

Abstract We study the following multi-robot coordination problem: given a graph, where each edge is weighted by the probability of surviving while traversing it, find a set of paths for K robots that maximizes the expected number of nodes collectively visited, subject to constraints on the probabilities that each robot survives to its destination. We call this the Team Surviving Orienteers (TSO) problem, which is motivated by scenarios where a team of robots must traverse a dangerous environment, such as aid delivery after disasters. We present the TSO problem formally along with several variants, which represent "survivability-aware" counterparts for a wide range of multi-robot coordination problems such as vehicle routing, patrolling, and informative path planning. We propose an approximate greedy approach for selecting paths, and prove that the value of its output is within a factor $1-e^{-p_s/\lambda}$ of the optimum where p_s is the per-robot survival probability threshold, and $1/\lambda \leq 1$ is the approximation factor of an oracle routine for the well-known orienteering problem. We also formalize an on-line update version of the TSO problem, and a generalization to heterogeneous teams where both robot types and paths are selected. We provide numerical simulations which verify our theoretical findings, apply our approach to real-world scenarios, and demonstrate its effectiveness in large-scale problems with the aid of a heuristic for the orienteering problem.

Keywords Multi-robot planning · Risk-averse planning · Vehicle routing problems · Orienteering problem · Submodular optimization

1 Introduction

Consider the problem of delivering humanitarian aid in a disaster or war zone with a team of robots. There are a number of sites which need the resources, but traveling among these sites is dangerous. While the aid agency wants to de-

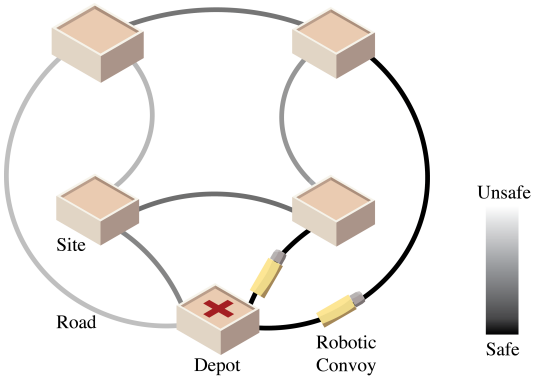


Fig. 1 Illustration of the TSO problem applied to an aid delivery scenario. The objective is to maximize the expected number of sites visited by at least one robotic convoy. Travel between sites is risky (as emphasized by the gray color scale for each edge), and paths must be planned to ensure that the return probability for each vehicle is above a survival threshold.

Partially supported by National Science Foundation grant DGE-114747, the Office of Naval Research: Science of Autonomy program, and Northrop Grumman Aerospace Systems. This article solely reflects the opinions and conclusions of the authors.

S. Jorgensen
Department of Electrical Engineering, Stanford University, Stanford, California 94305.
Tel.: +1-650-723-3212
E-mail: stefantj@stanford.edu

R. H. Chen · M. B. Milam
NG Next, Northrop Grumman Aerospace Systems, Redondo Beach, California 90278
E-mail: {robert.chen,mark.milam}@ngc.com

M. Pavone
Department of Aeronautics & Astronautics, Stanford University, Stanford, California 94035
E-mail: pavone@stanford.edu

liver aid to every city, it also seeks to limit the number of assets that are lost. We formalize this problem as an extension of the team orienteering problem (Golden et al 1987; Chao et al 1996), whereby one seeks to find a collection of paths in a doubly weighted graph which maximizes the sum of weights along all of the unique nodes in the paths while ensuring that the sum of edge weights in each path is less than a given budget. In the aid delivery case, the goal is to maximize the *expected* number of sites visited by at least one of the vehicles, while keeping the return *probability* for each vehicle above a specified survival threshold (i.e., while fulfilling a chance constraint for the survival of each vehicle). This can be seen as an extension of the team orienteering problem where edge weights are the negative log of the probability of surviving an edge, the budget is the negative log of the survival probability threshold, and node weights are the probability that the node is visited by at least one robot in the team. We call this problem formulation the "Team Surviving Orienteers" (TSO) problem, illustrated in Figure 1. The TSO problem builds on previous work in robotics, vehicle routing, and orienteering problems by considering *risky traversal:* when a robot traverses an edge, there is a probability that it is lost and does not visit any other nodes. This creates a complex, history-dependent coupling between the edges chosen and the distribution of nodes visited, which precludes the application of existing approaches available for solving the traditional orienteering problem.

The objective of this paper is to devise constant-factor approximation algorithms for the TSO problem, its extension to an on-line setting, and to heterogeneous teams. Our key technical insight is that, under mild conditions, the expected number of nodes visited (or functions thereof) satisfies a diminishing returns property known as submodularity, which for set functions means that $f(A \cup B) + f(A \cap B) \leq f(A) + f(B)$. Building upon this insight, we develop a *linearization procedure* for the problem, which leads to a greedy algorithm that enjoys a constant-factor approximation guarantee. We emphasize that while a number of works have considered orienteering problems with submodular objectives (Campbell et al 2011; Chekuri and Pál 2005; Zhang and Vorobeychik 2016) or chance constraints (Gupta et al 2012; Varakantham and Kumar 2013) separately, the combination of the two makes the TSO problem novel, as detailed next.

Related Work. The orienteering problem (OP) has been extensively studied (Gunawan et al 2016; Vansteenwegen et al 2011) 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 (Chekuri et al 2012). Below we highlight several variants which share either similar objectives or constraints as the TSO problem.

The *submodular orienteering problem* is a generalization of the orienteering problem which considers finding a *single* path which maximizes a 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 reward lower bounded by $\frac{\text{OPT}}{\log(\text{OPT})}$, where OPT is the optimum value. More recently, (Zhang and Vorobeychik 2016) developed a (polynomial-time) generalized cost-benefit algorithm and applied it to the submodular orienteering problem. The authors show that the output of their algorithm is lower bounded by $\frac{1}{2}(1-1/e)\text{OPT}^*$, where OPT^* is the optimum given a tighter budget (i.e., $\text{OPT}^* \leq \text{OPT}$). In our context, OPT^* roughly corresponds to the maximum expected number of nodes which can be visited when the survival probability threshold is increased to $\sqrt{p_s}$. For example, if the original problem has $p_s = 0.8$, then the guarantees provided by (Zhang and Vorobeychik 2016) would be with respect to the maximum expected number of nodes visited when the survival probability threshold is 0.894. Depending on the node and edge weights, this may be significantly different than the optimum for the original problem, making the bound loose. Our work extends the submodular orienteering problem to the team setting for a specific class of submodular functions (i.e., the coupled node rewards and edge weights which come from risky traversal) and we provide guarantees with respect to the optimum of the original problem.

In the *orienteering problem with stochastic travel times* proposed in (Campbell et al 2011), travel times are stochastic and reward is accumulated at a node only if it is visited before a deadline. This setting could be used to solve the single robot special case of the TSO problem by using a logarithmic transformation on the survival probabilities, but (Campbell et al 2011) does not provide any polynomial-time guarantees. In the *risk-sensitive orienteering problem* (Varakantham and Kumar 2013), the goal is to maximize the sum of rewards (which is history independent) subject to a constraint on the probability that the path cost is large. The TSO problem unifies the models of the risk-sensitive and stochastic travel time variants of the orienteering problem by considering both a submodular objective and a chance constraint on the total cost. In the TSO, we seek a *set* of paths for a team of robots which maximizes a history dependent objective function, specifically functions of the expected number of nodes visited by the team of robots. We also provide extensions for functions of multiple visits to a node, which allows broad applications such as informative path planning and property classification. Furthermore, we give an on-line version of the algorithm and provide a constant-factor guarantee for the heterogeneous team version of this problem (referred to as heterogeneous TSO –HTSO–), where robots may have different capabilities.

A second closely-related area of research is represented by the vehicle routing problem (VRP) (Pillac et al 2013; 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 *probabilistic VRP* (PVRP) considers stochastic edge costs with chance constraints on the path costs – similar to the risk-sensitive orienteering and the TSO problem constraints. The authors of (Laporte et al 1989) pose the simultaneous location-routing problem, where both paths and depot locations are selected to minimize path costs subject to a probabilistic connectivity constraint, which specifies the average case risk rather than individual risks. More general settings were considered in (Golden and Yee 1979), which considers several distribution families (such as the exponential and normal distributions), and (Stewart and Golden 1983), which considers non-linear risk constraints. In contrast to the TSO problem, the PVRP requires *every* node to be visited and seeks to minimize the travel cost. In the TSO problem, we require every path to be safe and maximize a function of the number of visits to each node.

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. They use submodularity of information gain to provide performance guarantees. Sampling-based approaches to IPP were proposed by (Hollinger and Sukhatme 2014), which come with asymptotic guarantees on optimality. The structure of the IPP is most similar to that of the TSO problem (since it is a multi-robot path planning problem with a submodular objective function which is non-linear and history dependent), but it does not capture the notion of risky traversal which is essential to the TSO problem. Our general approach is inspired by works such as (Atanasov et al 2015), but for the TSO problem we are able to further exploit the problem structure to derive constant-factor guarantees for our polynomial-time algorithm.

Statement of Contributions. The contribution of this paper is sixfold. First, we propose the Team Surviving Orienteers (TSO) problem. By considering a multi-robot (team) setting with submodular node rewards (e.g. expected number of nodes visited or information gained about node random variables), we extend the state of the art for the submodular orienteering problem, and by maximizing a submodular quality of service function (with guarantees on solution quality), we extend the state of the art in the probabilistic vehicle routing literature. From a practical standpoint, as discussed in Section 3, the TSO problem represents a "survivability-aware" counterpart for a wide range of multi-robot coordination problems such as vehicle routing, patrolling, and informative path planning. Second, we show that several broad classes of objective functions for the TSO problem are submodular, provide a linear relaxation of the single robot TSO problem (which can be solved as a standard orienteering problem), and show that the solution to the relaxed problem provides a close approximation of the optimal solution of the single robot TSO problem. Third, we propose an approximate greedy algorithm which has polynomial complexity in the number of nodes and linear complexity in the team size, and prove that the value of the output of our algorithm is lower bounded by $(1-e^{-p_s/\lambda})\mathrm{OPT}$, where OPT is the optimum value, p_s is the survival probability threshold, and $1/\lambda \leq 1$ is the approximation factor of an oracle routine for the solution to the orienteering problem (we note that, in practice, p_s is usually close to unity). Fourth, we formalize an on-line version of the TSO problem which enforces the survival constraint while taking into account the survival/failure events as they happen. We give a polynomial-time algorithm to approximately solve the on-line TSO problem, and provide guarantees on the performance of our on-line algorithm in terms of the objective obtained as well as bounds on the probabilities of worst-case events. Fifth, we discuss how to modify our algorithm to form heterogeneous teams, with similar performance guarantees and application scenarios. Finally, we demonstrate the effectiveness of our algorithm for large problems using simulations by solving a problem with 900 nodes and 25 robots. This paper significantly extends our prior work (Jorgensen et al 2017) by considering a wider class of objective functions, on-line updates, and heterogeneous teams.

Organization. In Section 2 we review key background information. In Section 3 we state the static and on-line TSO problems formally. In Section 4 we describe several applications of the TSO problem and show that their objective functions are submodular. In Section 5 we describe the linear relaxation technique and demonstrate how to solve the relaxed problem as an orienteering problem, outline a greedy solution approach for the static TSO problem, give approximation guarantees, and characterize the algorithm's complexity. In Section 6 we describe how to incorporate information gathered on-line to solve the on-line TSO problem, and give guarantees on the cumulative reward and number of surviving robots. In Section 7 we describe how to extend our approach and analysis for the HTSO problem. In Section 8 we verify the performance bounds, apply our approach to real-world scenarios, and demonstrate its scalability. Finally, we outline future work and draw conclusions in Section 9.

2 Background

In this section we review key material for our work and extend a well-known theorem in the combinatorial optimization literature to our setting.

2.1 Submodularity

Submodularity is the property of 'diminishing returns' for set functions. The following definitions are summarized from (Krause and Golovin 2014). Given a set $\mathscr{X}$, its possible subsets are represented by $2^{\mathscr{X}}$. For two sets X and X', the set $X' \setminus X$ contains all elements in X' but not X. The *complement* of a set X contains all elements of $\mathscr{X}$ not in X, and is denoted $X^c = \mathscr{X} \setminus X$. A set function $f : 2^{\mathscr{X}} \to \mathbb{R}$ is said to be *normalized* if $f(\emptyset) = 0$ and to be *monotone* if for every $X \subseteq X' \subseteq \mathscr{X}$, $f(X) \leq f(X')$. A set function $f : 2^{\mathscr{X}} \to \mathbb{R}$ is *submodular* if for every $X \subseteq X' \subset \mathscr{X}$, $x \in \mathscr{X} \setminus X'$, 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 $\Delta f(x \mid X)$.

2.2 The Approximate Greedy Algorithm

A typical submodular maximization problem entails finding a set $X \subseteq \mathscr{X}$ with cardinality K that maximizes f. Finding an optimal solution, X^*, is NP-hard for arbitrary submodular functions (Krause and Golovin 2014). The *greedy algorithm* constructs a set $\bar{X}_K = \{x_1, \ldots, x_K\}$ by iteratively adding an element x which maximizes the discrete derivative of f at the partial set already selected. In other words the ℓth element satisfies:

$$x_\ell \in \operatorname*{argmax}_{x \in \mathscr{X} \setminus \bar{X}_{\ell-1}} \Delta f(x \mid \bar{X}_{\ell-1}).$$

We refer to the optimization problem above as 'the greedy sub-problem' at step ℓ. A well-known theorem proven by (Nemhauser et al 1978) states that if f is a monotone, normalized, non-negative, and submodular set function, then $f(\bar{X}_K) \geq (1 - \frac{1}{e}) f(X^*)$. This is a powerful result, but if the set $\mathscr{X}$ 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. In particular, for some fixed $\alpha \leq 1$ the ℓth element $\hat{x}_\ell$ satisfies:

$$\Delta f(\hat{x}_\ell \mid \hat{X}_{\ell-1}) \geq \alpha \Delta f(x \mid \hat{X}_{\ell-1}) \qquad \forall x \in \mathscr{X} \setminus \hat{X}_{\ell-1}.$$

We provide a guarantee for the α–approximate greedy algorithm analogous to the guarantee for the greedy algorithm, thereby extending Theorem 4.2 of (Nemhauser et al 1978):

Theorem 1 (α–approximate greedy guarantee) *Let f be a monotone, normalized, non-negative, and submodular function with discrete derivative Δf. For $\alpha \in [0,1]$ and positive integer K, the output of any α–approximate greedy algorithm with $L \geq K$ elements, $\hat{X}_L$, satisfies the following inequality:*

$$f(\hat{X}_L) \geq \left(1 - e^{-\alpha L/K}\right) \max_{X \in 2^{\mathscr{X}} : |X| = K} f(X).$$

Proof The case where $L = K$ is a special case of Theorem 1 from (Wei et al 2014). To generalize to $L > K$ we extend the proof for the greedy algorithm in (Krause and Golovin 2014). Let $X^* \in 2^{\mathscr{X}}$ be a set which maximizes $f(X)$ subject to the cardinality constraint $|X| = K$. For $\ell < L$, we have:

$$\begin{aligned}
f(X^*) &\leq f(X^* \cup \hat{X}_\ell) \\
&= f(\hat{X}_\ell) + \sum_{k=1}^{K} \Delta f(x_k^* \mid \hat{X}_\ell \cup \{x_1^*, \ldots, x_{k-1}^*\}) \\
&\leq f(\hat{X}_\ell) + \sum_{k=1}^{K} \Delta f(x_k^* \mid \hat{X}_\ell) \\
&\leq f(\hat{X}_\ell) + \frac{1}{\alpha} \sum_{k=1}^{K} \Delta f(\hat{x}_{\ell+1} \mid \hat{X}_\ell) \\
&\leq f(\hat{X}_\ell) + \frac{K}{\alpha} (f(\hat{X}_{\ell+1}) - f(\hat{X}_\ell)).
\end{aligned}$$

The first line follows from the monotonicity of f, the second is a telescoping sum, and the third follows from the submodularity of f. The fourth line is due to the α–approximate greedy construction of $\hat{X}_L$, and the last is because all terms in the sum are equal. Now define $\delta_\ell = f(X^*) - f(\hat{X}_\ell)$. We can re-arrange the inequality above to yield:

$$\delta_{\ell+1} \leq \left(1 - \frac{\alpha}{K}\right) \delta_\ell \leq \left(1 - \frac{\alpha}{K}\right)^{\ell+1} \delta_0.$$

Since f is non-negative, $\delta_0 \leq f(X^*)$ and using the inequality $1 - x \leq e^{-x}$ we get

$$\delta_L \leq \left(1 - \frac{\alpha}{K}\right)^{L} \delta_0 \leq \left(e^{-\alpha L/K}\right) f(X^*).$$

Now substituting $\delta_L = f(X^*) - f(\hat{X}_L)$ and rearranging:

$$f(\hat{X}_L) \geq \left(1 - e^{-\alpha L/K}\right) f(X^*). \qquad \square$$

Remark: We can generalize this theorem to the case where each x_ℓ has guarantee α_ℓ. Using the same line of reasoning as in the proof for Theorem 1, we have

$$f(\hat{X}_L) \geq \left(1 - e^{-\sum_{\ell=1}^{L} \alpha_\ell / K}\right) f(X^*).$$

2.3 Graphs

Let $\mathscr{G}(\mathscr{V},\mathscr{E})$ denote an undirected graph, where $\mathscr{V}$ is the node set and $\mathscr{E} \subset \mathscr{V} \times \mathscr{V}$ 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, *complete* if there is an edge between each pair of nodes, and *planar* if nodes can be embedded in $\mathbb{R}^d$ in such a way that the edge weight is the distance between 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 denote the number of edges in a path by $|\rho|$. Under this notation, $\rho(|\rho|)$ is the last node in path ρ.

2.4 Poisson Binomial Distribution

The sum of K Bernoulli random variables with success probabilities $\{p_k\}_{k=1}^K$ follows the *Poisson binomial distribution*. Let $\mathscr{F}_m$ be the $\binom{K}{m}$ sets with m unique elements from $\{k\}_{k=1}^K$. For any $A \in \mathscr{F}_m$, its complement is denoted $A^c = \{k\}_{k=1}^K \setminus A$. The probability mass function for the Poisson binomial distribution is

$$f_{PB}\left(m;\{p_k\}_{k=1}^K\right) = \sum_{A\in\mathscr{F}_m} \prod_{i\in A} p_i \prod_{j\in A^c} (1-p_j),$$

which is the sum of the probabilities of each of the $\binom{K}{m}$ ways that exactly m variables are one and $K-m$ are zero. The special case where $p_k = p$ for all k, is referred to as the *binomial distribution* with parameters K and p. The binomial distribution has received much study because of its relatively simple form and extensive applications, but the Poisson binomial distribution is more difficult to analyze because each event has different probability. In the following lemma, we give a new result which gives sufficient conditions for the cumulative distribution function of a Poisson binomial random variable to be smaller than that of a specially crafted binomial random variable.

Lemma 1 (Bound for the Poisson Binomial Distribution) *For $K > 2$, let f_{PB} be a Poisson binomial probability mass function with parameters $\{p_k\}_{k=1}^K$, where $p_k \leq p_K$, and let f_B be a binomial probability mass function with parameters K and $p = \frac{1}{K}\sum_{k=1}^K p_k$. Then for $M \leq (1-p_K)\left((K-2)\frac{p}{1-p}\right) + p_K$,*

$$\sum_{m=0}^{M} f_{PB}(m) \leq \sum_{m=0}^{M} f_B(m).$$

Proof See the Appendix.

Although one could come up with a similar bound using a binomial distribution with parameters K and p_K, it would become quite loose if K becomes large or if p_K is very close to one but p is not. Lemma 1 is less susceptible to these effects since it uses the mean of $\{p_k\}_{k=1}^K$. We use this result later to derive performance bounds for our algorithms (by setting m as the number of robots which survive to the destination), but it has much broader uses outside the context of the TSO problem.

3 Problem Statement

In this section we give the formal problem statement for the static TSO problem and on-line TSO problem, provide an example, and give sufficient conditions for the objective function to be submodular.

3.1 Static TSO problem

Let $\mathscr{G}$ be a simple graph with $|\mathscr{V}| = V$ nodes. Edge weights $\omega : \mathscr{E} \to (0,1]$ correspond to the probability of survival for traversing an edge. Time is discretized into iterations $n = 0,1,\ldots,N$. At iteration $n \geq 1$ a robot following path ρ traverses edge $e_\rho^n = (\rho(n-1),\rho(n))$. Robots are indexed by variable k, and for each 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 and Table 1).

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

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

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

Variable	Description
e_ρ^n	The nth edge in path ρ, from node $\rho(n-1)$ to $\rho(n)$
$\omega(e)$	Probability robot survives edge e
p_s	Survival threshold for each robot
$s_n^k(\rho)$	One if robot k following path ρ survives edge e_ρ^n
$a_n^k(\rho)$	One if robot k following path ρ survives to iteration n
$z_j^k(\rho)$	One if robot k following path ρ visits node j
$p_j(m, X_K)$	Probability that m of the K robots following paths in set X_K visit node j

Table 1 Summary of notation for the TSO problem.

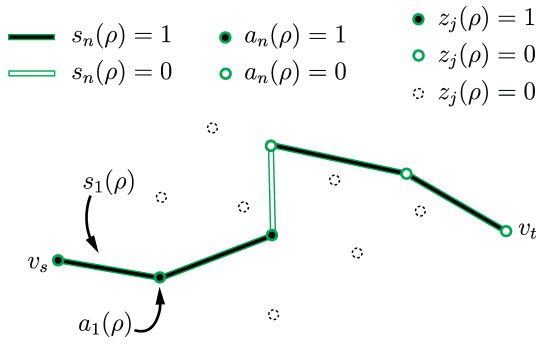


Fig. 2 Illustration of the notation used. Robot k plans to take path ρ, whose edges are represented by lines. The fill of the lines represent the value of $s_n^k(\rho)$. In this example $s_3^k(\rho) = 0$, which means that $a_3^k(\rho) = a_4^k(\rho) = a_5^k(\rho) = 0$. The variables $z_j^k(\rho)$ are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

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 $k \neq k'$, and the number of times that node j is visited by robots following the paths $\{\rho_k\}_{k=1}^K$ is given by $\sum_{k=1}^K z_j^k(\rho_k)$.

The number of robots which visit node j is distributed according to a Poisson binomial distribution. Given that K robots follow the paths $\{\rho_k\}_{k=1}^K$, we write the probability that exactly m robots visit node j as

$$p_j\left(m, \{\rho_k\}_{k=1}^K\right) := f_{PB}\left(m; \left\{\mathbb{E}[z_j^k(\rho_k)]\right\}_{k=1}^K\right).$$

Finally, let $h_j : \mathbb{Z}_+ \to \mathbb{R}_+$ be a function that maps the number of visits to node j to the reward accumulated at that node. Then the TSO problem is formally stated as:

Team Surviving Orienteers (TSO) Problem: Given a graph $\mathcal{G}$, edge weights ω, survival probability threshold p_s and team size K, maximize the expected reward of the node visits:

$$\begin{aligned}
\underset{\rho_1,\ldots,\rho_K}{\text{maximize}} \quad & \sum_{j=1}^{V} \mathbb{E}\left[h_j\left(\sum_{k=1}^{K} z_j^k(\rho_k)\right)\right] \\
\text{subject to} \quad & \mathbb{P}\left\{a_{|\rho_k|}^k(\rho_k) = 1\right\} \geq p_s \qquad k = 1,\ldots,K \\
& \rho_k(0) = v_s \qquad k = 1,\ldots,K \\
& \rho_k(|\rho_k|) = v_t \qquad k = 1,\ldots,K
\end{aligned}$$

Remarks – The objective represents the expected reward obtained by the K robots by visiting the nodes of the graph. The first set of constraints enforces the survival probability, the second and third sets of constraints enforce the initial and final node constraints. In particular, the survival probability threshold p_s serves two purposes: first, it requires that, on average, $p_s K$ robots will reach node v_t safely, and second, it enforces that risk is distributed fairly (i.e., no robot fails with too high a probability).

The model we use for risky traversal assumes that the survival random variables $s_n^k(\rho)$ are independent for all n and k. This is consistent with assumptions typical of navigation in unknown environments (e.g., SLAM applications where the environment is represented by occupancy grids), and navigation in adverse environments (e.g., due to piracy (Vaněk et al 2013) or storms (Zhang et al 2017)).

The TSO problem can be viewed as a set maximization problem with a cardinality constraint, where the domain of optimization is the set $\mathscr{X}$ containing K copies of *each* path in $\mathscr{X}(p_s, \omega)$. Crucially, if the objective function is a submodular function, then Theorem 1 guarantees that the greedily selected set of paths will achieve reward close to the optimum – a central result for this paper. Sufficient conditions for submodularity will be presented in Section 3.4. First, we state an online version of the TSO problem and provide an illustrative example.

3.2 On-line TSO problem

In the static TSO problem, the paths $\{\rho_k\}_{k=1}^K$ are computed at the beginning and then followed by the robots until the last iteration, with no path updates during execution. However at iteration n the variables $\{a_n^k(\rho_k)\}_{k=1}^K$ are observed, and this knowledge could be used to update the paths in order to account for the realized robot failures. Specifically, we seek to re-plan the paths surviving robots take such that the expected number of robots which reach node v_t safely is still $p_s K$ (consistent with the initial safety threshold), and that the risk is still distributed fairly. This can be accomplished by choosing a new survival probability threshold as follows.

Define the list of surviving robots at iteration n as $U_n := \{k \in \{1,\ldots,K\} : a_n^k(\rho_k) = 1\}$. Also, for robots $k \in U_n$, let the maximum probability that robot k can reach node v_t be denoted by ψ_k. The survival probability threshold at iteration n, denoted as p_s^n, is computed as the solution to the optimization problem:

$$\begin{aligned}
\underset{p \in (0,1)}{\text{minimize}} \quad & p \\
\text{subject to} \quad & p_s K \leq \sum_{k \in U_n} \min\{p, \psi_k\}.
\end{aligned}$$

Intuitively, p_s^n represent the smallest probability threshold p such that the average number of robots which reach v_t

safely will be no smaller then $p_s K$, while accounting for the fact that the maximum probability with which robot k can reach node v_t is ψ_k. If the minimization problem is infeasible, this means that for any set of paths, the expected number of robots that will reach node v_t safely is smaller than $p_s K$, and so p_s^n is simply set to one. We then define the on-line TSO problem as:

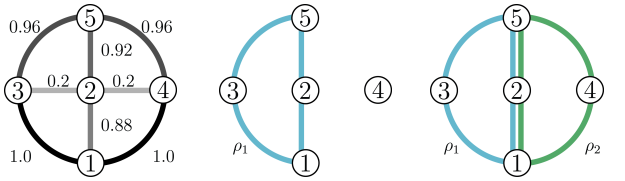
On-line Team Surviving Orienteers Problem: At iteration n, given a graph $\mathscr{G}$, edge weights ω, survival probability threshold p_s^n, paths $\{\rho_k^{n-1}\}_{k=1}^K$, and the list of surviving robots U_n, select new paths $\{\rho_k^n\}_{k=1}^K$ in order to maximize the cumulative rewards:

$$\begin{aligned}
\underset{\rho_1^n,\ldots,\rho_K^n}{\text{maximize}} \quad & \sum_{j=1}^{V} \mathbb{E}\left[h_j\left(\sum_{k\in U_n} z_j^k(\rho_k^n) \right) \mid a_n^k(\rho_k^n) = 1, \quad k \in U_n \right] \\
\text{subject to} \quad & \rho_k^n(n') = \rho_k^{n-1}(n') \qquad n' = 1,\ldots,n, \qquad k \in U_n \\
& \rho_k^n(|\rho_k^n|) = v_t \qquad\qquad\qquad\qquad\quad k \in U_n \\
& \mathbb{P}\left\{ a^k_{|\rho_k^n|}(\rho_k^n) = 1 \right\} \geq \min\{p_s^n, \psi_k\} \qquad k \in U_n
\end{aligned}$$

The objective is to maximize the expected cumulative reward conditioned on the set of surviving robots. The first constraint enforces continuity with actions taken up to iteration n, the second constraint enforces that each path ends at v_t, and the third constraint enforces the survival probability constraint. Note that if $p_s^n = 1$, this means that the number of robots which reach node v_t is expected to be less than $p_s K$ regardless of the paths chosen. If for any robot k, $p_s^n > \psi_k$, then this robot will take the one of the safest paths to v_t, and will reach v_t with probability ψ_k.

3.3 Example

An example of the TSO problem with a reward function that is one if the node is visited at least once and zero otherwise, is given in Figure 3(a). There are five nodes, and edge weights are shown next to their respective edges. Two robots start at node 1, and must end at virtual node $1'$ (which is a copy of node 1) with probability at least $p_s = 0.75$.



(a) Graph $\mathscr{G}$ (b) Path for one robot (c) Path for two robots

Fig. 3 (a) Example of a TSO problem. Robots start at the bottom (node 1) and darker lines correspond to safer edges. (b) A single robot can only visit four nodes safely. (c) Two robots can visit all nodes safely. It is easy to verify that adding more robots yields diminishing returns.

Path $\rho_1 = \{1,3,5,2,1'\}$ is shown in Figure 3(b), and path $\rho_2 = \{1,4,5,2,1'\}$ is shown alongside ρ_1 in Figure 3(c). Robot 1 visits node 3 with probability 1.0 and node 5 with probability 0.96. Robot 2 also visits node 5 with probability 0.96 and so the probability at least one robot visits node 5 is $\mathbb{E}[1 - p_5(0,\{\rho_1,\rho_2\})] = 0.9984$. The probability that robot 1 returns safely is $\mathbb{E}\left[a_4^1(\rho_1)\right] = 0.794$. For this simple problem, ρ_1 and ρ_2 are two of three possible paths (the third is $\{1,3,5,4,1'\}$). The expected number of nodes visited by the first robot following ρ_1 is 3.88, and for two robots following ρ_1 and ρ_2 it is 4.905. Since there are only five nodes, it is clear that adding more robots must yield diminishing returns.

3.4 Sufficient Conditions for Submodular Objective

The domain of optimization for the TSO problem with K robots is the set $\mathscr{X}$ that contains K copies of each element in $\mathscr{X}(p_s,\omega)$. With mild conditions on the functions $\{h_j\}_{j=1}^V$, the objective function for the TSO problem (and also for the on-line TSO problem) is submodular, as stated below.

Lemma 2 (Submodularity of the TSO problem objective) *Consider a set of paths $X_K := \{\rho_k\}_{k=1}^K$ and the objective function*

$$J(X_K) = \sum_{j=1}^{V} \mathbb{E}\left[h_j\left(\sum_{k=1}^{K} z_j^k(\rho_k) \right) \right].$$

For $L \geq 1$, the objective function has discrete derivative with respect to path ρ_L at partial solution $X_{L-1} = \{\rho_\ell\}_{\ell=1}^{L-1}$,

$$\Delta J(\rho_L \mid X_{L-1}) = \sum_{j=1}^{V} \mathbb{E}\left[z_j^L(\rho_L)\right] \delta_j(X_{L-1}),$$

where we define the set function,

$$\delta_j(X_K) := \mathbb{E}\left[h_j\left(1 + \sum_{k=1}^{K} z_j^k(\rho_k) \right) \right] - \mathbb{E}\left[h_j\left(\sum_{k=1}^{K} z_j^k(\rho_k) \right) \right].$$

Furthermore, the objective function is submodular if for all j, $-\delta_j(X)$ is a monotone function of X.

Proof Let $L \geq 1$. The random variable $z_j^L(\rho_L)$ is independent of the random variables $\{z_j^\ell(\rho_\ell)\}_{\ell=1}^{L-1}$. Hence from the definition of the discrete derivative and the tower property one has:

$$\begin{aligned}
\Delta J(\rho_L \mid X_{L-1}) = & \sum_{j=1}^{V} \mathbb{E}[z_j^L(\rho_L)] \mathbb{E}\left[h_j\left(1 + \sum_{\ell=1}^{L-1} z_j^\ell(\rho_\ell) \right) \right] \\
& + \left(1 - \mathbb{E}\left[z_j^L(\rho_L)\right] - 1\right) \mathbb{E}\left[h_j\left(\sum_{\ell=1}^{L-1} z_j^\ell(\rho_\ell) \right) \right],
\end{aligned}$$

which upon simplification yields the first statement of the lemma.

We now consider the second statement of the lemma. By definition, a set function is submodular if the negative of its discrete derivative is a monotone function. If $-\delta(j)$ is monotone, then the negative of the discrete derivative is also monotone (since $\mathbb{E}[z_j^L(\cdot)] \geq 0$ and the sum of monotone functions is monotone). Hence the objective function is submodular. □

Note that we can easily extend this result to the on-line case by conditioning on $\{a_n^k(\rho_k)\}_{k=1}^K$. For the remainder of this paper we will restrict our attention to TSO problems that fulfill the assumptions of Lemma 2. This class of problems is indeed large; we show several examples in the next section.

4 Applications and Examples

The TSO problem has many applications which have submodular reward functions, which means that a greedily selected set of paths will give near-optimal rewards (as discussed in Section 2). We provide some specific examples of such applications below.

4.1 Aid Delivery (single-visit rewards)

Consider an aid delivery problem where robots deliver a resource to sites with different demands. The reward accumulated for delivering resources to node j is the demand $d_j \geq 0$, and reward is only accumulated for the first visit. Formally, for $X_K = \{\rho_k\}_{k=1}^K$, the objective function is

$$\sum_{j=1}^{V} \mathbb{E}\left[h_j\left(\sum_{k=1}^{K} z_j^k(\rho_k)\right)\right] = \sum_{j=1}^{V} \mathbb{E}\left[d_j \mathbb{I}\left\{\sum_{k=1}^{K} z_j^k(\rho_k) > 0\right\}\right] = \sum_{j=1}^{V} d_j\,(1 - p_j(0, X_K)).$$

We refer to this form of objective function as an *single-visit reward function*, because reward is only accumulated for the first visit to a node. The following lemma shows that such reward functions are submodular:

Lemma 3 (Submodularity of single-visit rewards) *For $d_j \geq 0$, the single-visit reward function,*

$$\sum_{j=1}^{V} d_j(1 - p_j(0, X_K)),$$

is a normalized, non-negative, monotone, and submodular function with discrete derivative with respect to ρ_L at partial solution $X_{L-1} = \{\rho_\ell\}_{\ell=1}^{L-1}$

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

Proof Non-negativity follows from the fact that $d_j \geq 0$ and $p_j(\cdot) \geq 0$. The normalized property follows since $p_j(0, \emptyset) = 1$, and since $p_j(0, X)$ is a decreasing function of X, the objective function is monotone. For the single-visit reward function the quantity,

$$-\delta_j(X) = -d_j\, p_j(0, X),$$

is monotone, so using Lemma 2 we conclude that the objective function is submodular. □

4.2 Property Classification (multi-visit rewards)

Now consider a multiple visit reward function where reward $h_j(m) \geq 0$ is accumulated after m visits to node j. A concrete example is a classification scenario, where each robot measures a binary property of a node imperfectly, and the objective is to minimize the posterior variance of the property distribution. If one uses a Haldane prior, which is the $\beta(0,0)$ function (Haldane 1932), the posterior variance after m measurements is $\frac{1}{4(m+1)}$. Setting the node priorities to $h_j(m) = \left(\frac{1}{4} - \frac{1}{4(m+1)}\right)$ gives a multi-visit reward function. Maximizing the expected cumulative rewards $h_j(m)$ is equivalent to minimizing the expected posterior variance of the distribution of the feature probabilities.

With fairly mild conditions on the rewards h_j, the multi-visit reward function is submodular, as stated in the following lemma:

Lemma 4 (Submodularity of multi-visit rewards) *Let $h_j : \mathbb{Z}_+ \to \mathbb{R}_+$ be an increasing function with finite difference $\Delta_{h_j}(m) = h_j(m) - h_j(m-1)$ which satisfies the diminishing returns property*

$$\Delta_{h_j}(m+1) \leq \Delta_{h_j}(m), \qquad m \geq 1$$

and $h_j(0) = 0$. Then the reward function at the solution set $X_K = \{\rho_k\}_{k=1}^K$,

$$\sum_{j=1}^{V} \mathbb{E}\left[h_j\left(\sum_{k=1}^{K} z_j^k(\rho_k)\right)\right],$$

is a normalized, non-negative, monotone, and submodular function with discrete derivative with respect to ρ_L at partial solution $X_{L-1} = \{\rho_\ell\}_{\ell=1}^{L-1}$:

$$\sum_{j=1}^{V} \mathbb{E}[z_j^L(\rho_L)] \sum_{m=0}^{L-1} \Delta_{h_j}(m+1)\, p_j(m, X_{L-1}).$$

Proof The reward function is non-negative because $h_j(\cdot) \geq 0$ and normalized because $h_j(0) = 0$. The number of visits to a node, $\sum_{k=1}^K z_j^k(\rho_k)$, is a monotone function. Since h_j is an increasing function of the number of visits to a node, this

implies that the objective function is monotone. From the definition of the multi-visit reward function we have

$$\delta_j(X) = \sum_{m=0}^{|X|} \Delta_{h_j}(m+1)p_j(m,X).$$

Consider $Y = X \cup x$ and define $\gamma \in [0,1]$ such that $p_j(m,Y) = (1-\gamma)p_j(m,X) + \gamma p_j(m-1,X)$. From the definition of $\delta_j(X)$ and using the properties of h_j, we have

$$\begin{aligned}\delta_j(X) - \delta_j(Y) &= \sum_{m=0}^{|X|} \Delta_{h_j}(m+1)p_j(m,X) \\ &\quad - \sum_{m=0}^{|X|+1} \Delta_{h_j}(m+1)\left(\gamma p_j(m-1,X) + (1-\gamma)p_j(m,X)\right) \\ &= \gamma \sum_{m=1}^{|X|} \left(\Delta_{h_j}(m+1) - \Delta_{h_j}(m+2)\right)p_j(m,X) \\ &\geq 0.\end{aligned}$$

The first equality is derived by expressing $p_j(m,Y)$ in terms of $p_j(m,X)$, the second from simplification and the fact that $p_j(|X|+1,X) = 0$, and the inequality is due to the diminishing returns property of h_j.

This implies that $-\delta_j(X)$ is monotone and so from Lemma 2 we have that the multi-visit objective function is submodular with the stated discrete derivative. □

Note that the objective function of the feature classification example at the beginning of this subsection satisfies the conditions of Lemma 4, and hence it is a normalized, non-negative, monotone and submodular function.

4.3 Informative Path Planning

The multi-visit reward function can also model an informative path planning problem, where each node has a random variable Y_j, and the objective is to select measurements in order to minimize the entropy of the posterior distribution of $\{Y_j\}_{j=1}^V$ given the measurements. Setting $h_j(m)$ to be the information gained about Y_j after taking m measurements, the TSO problem becomes an informative path planning problem. It is easy to verify that information functions satisfy the conditions of Lemma 4: the information gained from taking zero measurements is zero, information is an increasing function, and it satisfies the diminishing returns property. Hence we have that the informative path planning application of the TSO problem has a submodular objective function.

4.4 Edge Rewards

Each of the formulations above can easily be extended to a scenario where the goal is to maximize a function of the *edges* traversed. Here we describe how to extend the single-visit case, but a similar approach can be used for any of the other reward functions described above. Define $z_{i,j}^k(\rho)$ to indicate whether robot k following path ρ takes edge (i,j), and for $(i,j) \in \mathscr{E}$ define $p_{i,j}(m, \{\rho_k\}_{k=1}^K)$ as in the single visit case with $z_j^k(\cdot)$ replaced by $z_{i,j}^k(\cdot)$ (if $(i,j) \notin \mathscr{E}$, then set $p_{i,j}(0,\cdot) = 1$). Instead of node rewards d_j, we now have edge rewards $d_{i,j}$ (with $d_{i,j} = 0$ if $(i,j) \notin \mathscr{E}$), and the objective function is

$$\sum_{i=1}^{V}\sum_{j=1}^{V} d_{i,j}\left(1 - p_{i,j}(0, \{\rho_k\}_{k=1}^K)\right).$$

This variant could be used to model a patrolling problem, where the goal is to inspect the maximum number of roads subject to the survival probability threshold. Such problems also occur when planning scientific missions (e.g., on Mars), where the objective is to execute the most important traversals.

4.5 Visit Risks

Consider a scenario where the action of visiting a node is risky: a robot visiting node j survives with probability $\upsilon(j)$ and fails with probability $1-\upsilon(j)$. We can easily incorporate this additional randomness into the TSO problem by using a directed graph to represent the traversals and directed edge weights, $\omega^d(i,j) = \omega(i,j)\upsilon(j)$, which incorporate the visit risk.

4.6 Non-homogeneous Traversal Time

The formal definitions of the static and on-line TSO problem above assume that exactly one edge is traversed per iteration by each robot. This greatly simplifies notation by allowing us to index paths with the time variable n, but is not a necessary assumption for our results.

For the static variant, homogeneous traversal time is purely a notational convenience and arbitrary times can be handled without modifying the problem statement. This is because the variables a_N^k are evaluated from the perspective of time zero and do not depend on the number of events between time zero and time N (and hence are independent of how 'time' is split).

The on-line variant is slightly more involved. At any given time t (now not necessarily the same as the index n for each path), each robot will either be at the terminal node, or be travelling to node $\rho_k^{t'}(n(\rho_k^{t'},t))$, where t' is the last time that robot k's path was updated, and $n(\rho_k^{t'},t)$ is defined as the index of the first node visited after time t in path $\rho_k^{t'}$. If we allow mid-course corrections (for example, with aerial vehicles), then the continuity constraints become

$$\rho_k^t(n') = \rho_k^{t'}(n') \qquad n' = 1,\ldots,n(\rho_k^{t'},t)-1, \quad k \in U_t,$$

and if we do not allow mid-course corrections (for example, on road networks), then the continuity constraints become

$$\rho_k^t(n') = \rho_k^{t'}(n') \qquad n' = 1,\ldots,n(\rho_k^{t'},t), \qquad k \in U_t.$$

In either case the fundamental structure of the problem remains the same as when traversal times are homogeneous.

5 Approximate Solution Approach to the Static TSO

As discussed in Section 3.4, we restrict our attention to TSO problems with objective functions that fulfill the assumptions of Lemma 2. Our approach to solving the TSO problem is then to exploit submodularity of the objective function using an α–approximate greedy algorithm (as defined in Section 2.2). In Section 5.1 we present a linearization of the greedy sub-problem, which in the context of the TSO problem entails finding a path which maximizes the discrete derivative of the objective function, at the partial set already constructed. We use this linearization to find a polynomial-time (p_s/λ)–approximate greedy algorithm for finding the best path given a partial solution. Leveraging this result, we describe our `GreedySurvivors` algorithm for the TSO problem in Section 5.2, discuss its approximation guarantee in Section 5.3, and characterize its computational complexity in Section 5.4.

5.1 Linear Relaxation for the Greedy Sub-problem

Given a previously selected set of paths, $X_{L-1} = \{\rho_\ell\}_{\ell=1}^{L-1}$, the greedy sub-problem for the TSO problem at step L requires us to find a path ρ_L from the set $\mathscr{X} \setminus X_{L-1}$ which maximizes the discrete derivative of the objective function at X_{L-1} with respect to ρ_L. Note that because we define $\mathscr{X}$ to have as many copies of each path as the maximum number of robots we plan for, the set $\mathscr{X} \setminus \{\rho_\ell\}_{\ell=1}^{L-1}$ always contains at least one copy of each path in $\mathscr{X}(p_s,\omega)$. Since the discrete derivative of the objective function at X_{L-1} with respect to any of the copies of a path $\rho \in \mathscr{X}(p_s,\omega)$ is the same, we can solve the greedy sub-problem by only considering elements in the set $\mathscr{X}(p_s,\omega)$. Even with this simplification, the greedy sub-problem is very difficult for the TSO problem: it requires finding a path which maximizes submodular node rewards subject to a budget constraint (this is the submodular orienteering problem). No polynomial-time constant-factor approximation algorithm is known for general submodular orienteering problems (Chekuri and Pál 2005), and so in this section we design one specifically for the greedy sub-problem for the TSO problem.

Under the assumptions of Lemma 2, the discrete derivative is of the form $\sum_{j=1}^{V} \mathbb{E}[z_j^L(\rho_L)]\delta_j(X_{L-1})$, for $\delta_j(X_{L-1}) \geq 0$. We relax the problem of maximizing the discrete derivative by replacing the probability that robot L traversing path ρ visits node j, $\mathbb{E}[z_j^L(\rho)]$, with the maximum probability that any robot following a feasible path can visit node j, ζ_j:

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

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

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

which represents an *optimistic* estimate of the actual discrete derivative of our objective function at X_{L-1} with respect to ρ. We can find the (approximately) best path by solving an orienteering problem as follows. Recall that for the orienteering problem we provide node weights and a constraint on the sum of edge weights (referred to as a budget), and find the path which maximizes the node rewards along the path while guaranteeing that the sum of edge weights along the path is below the budget.

We use the graph $\mathscr{G}_O$, which has the same edges and nodes as $\mathscr{G}$ but has edge weights $\omega_O(e)$ and node rewards $v_L(j) = \zeta_j\delta_j(X_{L-1})$. Solving the orienteering problem on $\mathscr{G}_O$ with budget $-\log(p_s)$ will return a path that maximizes the sum of node rewards (which is $\Delta\bar{J}(\rho \mid X_{L-1})$), and satisfies $\sum_{e\in\rho} -\log(\omega(e)) \leq -\log(p_s)$, which is equivalent to $\mathbb{P}\{a^L_{|\rho|}(\rho) = 1\} \geq p_s$.

Although solving the orienteering problem is NP-hard, several polynomial-time constant-factor approximation algorithms exist which guarantee that the returned objective is lower bounded by a factor of $1/\lambda \leq 1$ of the optimal objective. For undirected graphs (Chekuri et al 2012) gives a guarantee $\lambda = (2+\varepsilon)$ with complexity $O(V^{1/\varepsilon^2})$, and for directed graphs (Chekuri and Pál 2005) gives a guarantee in terms of the number of nodes. An important special case is when the nodes represent points in $\mathbb{R}^d$ and for all edges in $\mathscr{E}$ the edge weights $\omega_O(e)$ is proportional to the distance between the source and sink nodes of the edge. In this case, the graph $\mathscr{G}_O$ is called *planar*, and the underlying orienteering problem is significantly easier to solve. For undirected planar graphs (Chen and Har-Peled 2006) gives a guarantee $\lambda = (1+\varepsilon)$ with complexity $O(V^{1/\varepsilon})$. Using such an oracle, we have the following guarantee:

Lemma 5 (Single robot constant-factor guarantee) *Let* `Orienteering` *be a routine that solves the orienteering problem within constant-factor* $1/\lambda$*, that is for node weights* $v(j) = \zeta_j c_j$*, path* $\hat{\rho}$ *output by the routine and any path* $\rho \in \mathscr{X}(p_s,\omega)$*,*

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

Then for any $c_j > 0$ and any $\rho \in \mathscr{X}(p_s, \omega)$, the cumulative rewards for a robot following path $\hat{\rho}$ satisfies

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

Proof By definition of ζ_j and the `Orienteering` routine, we have:

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

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

This is a remarkable statement because it guarantees that, if we solve the orienteering problem near-optimally, choose $c_j = \delta_j(X_{L-1})$ and p_s is not too small, the solution to the linear relaxation will give nearly the same result as the optimal solution to the greedy sub-problem at step L for the TSO problem. The intuition is that for p_s close to unity no feasible path can be very risky and so the probability that a robot *actually* reaches a node will not be too far from the maximum probability that it *could* reach the node.

5.2 Greedy Approximation for the TSO Problem

Using this relaxation with $c_j = \delta_j(X_{L-1})$ we have an p_s/λ–approximate algorithm for the greedy sub-problem at step L. This gives us a $(1 - e^{-p_s/\lambda})$–approximate greedy algorithm for maximizing the discrete derivative of the objective function for the variants discussed in Section 4, as detailed next.

Define the method `Dijkstra`($\mathscr{G}$, i, j), which returns the length of the shortest path from i to j on the edge weighted graph $\mathscr{G}$ using Dijkstra's algorithm. Given an edge weighted graph $\mathscr{G}$ and node rewards v, the `Orienteering`$(\mathscr{G}, v)$ routine solves the orienteering problem (assuming $v_s = 1$, $v_t = V$ and budget $-\log(p_s)$) within factor $1/\lambda$, and returns the best path. Pseudocode for our algorithm is given in Figure 4. We begin by forming the graph $\mathscr{G}_O$ with log-transformed edge weights $\omega_O(e)$, and then use Dijkstra's algorithm to compute the maximum probability that a node can be reached. For each robot $k = 1, \ldots, K$, we solve an orienteering problem to greedily choose the path that maximizes the discrete derivative of $\bar{J}$.

Given a node index j and set of paths X, the `update`(j, X) routine returns the value of $\delta_j(X)$ as detailed below.

Updates for single-visit reward functions– Recall from Lemma 3 that for the single-visit reward function

$$\delta_j(X_L) = d_j\, p_j(0, X_L),$$

```
1: procedure GreedySurvivors(𝒢, K)
2:     Form 𝒢_O from 𝒢, such that v_s = 1, v_t = V
3:     for j = 1, ..., V do
4:         ζ_j ← exp(−Dijkstra(𝒢_O, 1, j))
5:     end for
6:     for k = 1, ..., K do
7:         for j = 1, ..., V do
8:             c_j ← Update(j, {ρ_ℓ}_{ℓ=1}^{k−1})
9:             v_k(j) ← ζ_j c_j
10:        end for
11:        ρ_k ← Orienteering(𝒢_O, v_k)
12:    end for
13: end procedure
```

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

which can be computed efficiently. Initially, $\delta_j(\emptyset) = d_j$. When adding ρ_L, $\delta_j(X_L)$ updates to

$$\delta_j(X_L) \leftarrow \left(1 - \mathbb{E}[z_j^L(\rho_L)]\right) \delta_j(X_{L-1}),$$

which can be interpreted as the value of the node times the probability that none of the first L robots visit node j. The complexity of updating the node weights is $O(V)$.

Updates for multi-visit reward functions– Recall from Lemma 4 that for the multi-visit reward function

$$\delta_j(X_L) = \sum_{m=0}^{|X_L|} \Delta_{h_j}(m+1)\, p_j(m, X_L),$$

which can be updated by tracking the probability distribution of the number of visits to each node. The probability $p_j(m, X_L)$ can be computed recursively, since we have

$$\begin{aligned} p_j(m, X_L) = {} & \mathbb{E}[z_j^L(\rho_L)]\, p_j(m-1, X_{L-1}) \\ & + (1 - \mathbb{E}[z_j^L(\rho_L)])\, p_j(m, X_{L-1}). \end{aligned}$$

Updating the node weights requires $O(L+1) \leq O(K)$ computations.

5.3 Approximation Guarantees

In this section we combine the results from Section 2.2 and 5.1 to prove that the output of the `GreedySurvivors` algorithm is close to the optimal solution to the TSO problem. Specifically, we compare a team with $L \geq K$ robots using greedily selected paths to a team with K optimally selected paths, because this gives us a way to compute tighter bounds on the performance of our algorithm.

Theorem 2 (Multi-robot constant-factor guarantee) *Given an `Orienteering` routine with constant-factor guarantee $1/\lambda$ as in Lemma 5, assign robot ℓ path $\hat{\rho}_\ell$ corresponding to the path returned by the `Orienteering` routine on graph $\mathscr{G}_O$ with node weights $v_j = \zeta_j \delta_j(\{\hat{\rho}_k\}_{k=1}^{\ell-1})$.*

Let $X_K^* = \{\rho_k^*\}_{k=1}^K$ be an optimal solution to the TSO problem with K robots. For some $L \geq K$ and $1 \leq \ell \leq L$, suppose the objective is a normalized, non-negative, monotone, and submodular function with discrete derivative of the form

$$\Delta J(\rho_\ell \mid X_{\ell-1}) = \sum_{j=1}^{V} \mathbb{E}[z_j^\ell(\rho_\ell)]\delta_j(X_{\ell-1}).$$

Then the expected cumulative reward gathered by a team of L robots with types and paths $\hat{X}_L = \{\hat{\rho}_\ell\}_{\ell=1}^L$ is at least a fraction $\gamma = \left(1 - e^{-\frac{p_s L}{\lambda K}}\right)$ of the optimal:

$$\sum_{j=1}^{V} \mathbb{E}\left[h_j\left(\sum_{\ell=1}^{L} z_j^\ell(\hat{\rho}_\ell)\right)\right] \geq \gamma \sum_{j=1}^{V} \mathbb{E}\left[h_j\left(\sum_{k=1}^{K} z_j^k(\rho_k^*)\right)\right].$$

Proof The objective is a set function with domain $\mathscr{X}$, which has L copies of each feasible path. Hence for $1 \leq \ell < L$, the set $\mathscr{X} \setminus \{\hat{\rho}_k\}_{k=1}^{\ell-1}$ will always contain at least one copy of each path in $\mathscr{X}(p_s, \omega)$, and since the discrete derivative evaluated at any of the copies of the same path is the same, we can solve the greedy sub-problem by only considering elements in $\mathscr{X}(p_s, \omega)$. Using Lemma 5 with c_j chosen appropriately for the objective function, we have a constant-factor guarantee $\alpha = p_s/\lambda$ for the problem of finding the path from $\mathscr{X}(p_s, \omega)$ that maximizes the discrete derivative of our objective function. Now applying Theorem 1 to our objective function (which by assumption is normalized, non-negative, monotone, and submodular) we have the desired result. □

In many scenarios of interest p_s is quite close to 1, since robots are typically valuable or difficult to replace. For $L = K$ this theorem gives an $1 - e^{-p_s/\lambda}$ guarantee for the output of our algorithm. This bound holds for any team size, and guarantees that the output of the (polynomial-time) linearized greedy algorithm will have a similar reward to the output of the (exponential time) optimal algorithm.

Taking $L > K$ gives a practical way of testing how much more efficient the allocation for K robots could be. For example, if $L\frac{p_s}{\lambda} = 6K$ we have a $(1 - 1/e^6) \simeq 0.997$ factor approximation for the optimal value achieved by K robots. We use this approach to generate tight upper bounds for our experimental results. Note that this theorem also guarantees that as $L \to \infty$, the output of our algorithm has at least the same value as the optimum, which emphasizes the importance of guarantees for *small* teams.

Next we use the Poisson binomial bound from Section 2.4 to bound the probability of worst-case events, namely that a small number of robots reach node v_t safely.

Lemma 6 (Worst-case Probability Bounds) *For $K > 2$, let $X_K = \{\rho_k\}_{k=1}^K$ be a set of paths which is a feasible solution to the TSO problem. Denote $p_K := \max_k \mathbb{E}[z_{v_t}^k(\rho_k)]$ and let $\mu := \frac{1}{K}\sum_{k=1}^K \mathbb{E}[z_{v_t}^k(\rho_k)] \geq p_s$ be the expected fraction of robots which will reach node v_t. Then for $M \leq \lfloor (1 - p_K)(K-2)\frac{\mu}{1-\mu} + p_K \rfloor$, the probability that M or fewer robots reach node v_t decreases exponentially as M decreases:*

$$\sum_{m=0}^{M} p_{v_t}(m, X_K) \leq \exp\left(-2K(\mu - M/K)^2\right).$$

Proof Recall that if robots follow paths X_K, the probability that m robots reach node v_t is $p_{v_t}(m, X_K)$, which is the Poisson binomial probability mass function evaluated at m with parameters $\left\{\mathbb{E}[z_{v_t}^k(\rho_k)]\right\}_{k=1}^K$. Using Lemma 1, we have that the Poisson binomial cumulative distribution function is bounded by the binomial cumulative distribution function with parameters K and μ. Applying Hoeffding's inequality,

$$\begin{aligned}\sum_{m=0}^{M} p_{v_t}(m, X_K) &\leq \sum_{m=0}^{M} \binom{K}{m} \mu^m (1-\mu)^{K-m} \\ &\leq \exp\left(-2\frac{(K\mu - M)^2}{K}\right)\end{aligned}$$

which after simplification is the stated result. □

This statement gives a very strong guarantee that the number of surviving robots will not be significantly below $p_s K$. For example, if $K = 25$, $\mu = 0.85$ and $p_K \leq 0.89$, then the probability that 15 or fewer robots reach v_t is less than 0.044, but the probability that 13 or fewer robots reach v_t is less than 0.0043.

5.4 Computational Complexity

Suppose that the complexity of the `Orienteering` oracle is C_O, and the complexity of the update step is C_U. Then the complexity of our algorithm is:

$$O(V^2 \log(V)) + O(KC_U) + O(KC_O).$$

The first term is the complexity of running Dijkstra's to calculate ζ_j for all nodes, the second term is the complexity of updating the weights K times, and the final term is the complexity of solving the K orienteering problems. Generally $C_U = O(V)$ and is dominated by C_O so the asymptotic complexity of our algorithm is KC_O. Relying on an oracle routine makes the `GreedySurvivors` routine applicable for several diverse communities of researchers.

Complexity theory – From a theoretical standpoint, if a polynomial-time approximation scheme (PTAS) for the orienteering problem is used, then our algorithm is a PTAS for the TSO problem. This is a meaningful result on the complexity of the TSO problem: although the TSO is NP-hard, it can be approximated within a constant factor in polynomial time. 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 $\lambda = 1 + \varepsilon$ PTAS for the planar case which runs in

$O(V^{16d^{3/2}/\varepsilon})$ time, where d in this context is the dimension of the plane that nodes are embedded in. Even for $\varepsilon = 1$ and $d = 2$, this is $O(V^{46})$, which is not suitable for real-world applications.

Certifiable performance applications – Practitioners who require guarantees on the quality of the solution can use mixed integer linear programming (MILP) formulations of the orienteering problem (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 $1/\lambda$.

Time-critical applications – 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.

5.5 Modifications for Variants

Edge rewards – The `GreedySurvivors` routine is easily modified for the edge rewards variant. After re-defining the variables as described in Section 4.4, define $\zeta_{i,j} = \zeta_i \omega(i,j)$, which is the largest probability that edge (i,j) is successfully taken. The linearized greedy algorithm will still have a constant-factor guarantee, but now requires solving an *arc* orienteering problem. Constant-factor approximations for the arc orienteering problem can be found using algorithms for the OP as demonstrated in (Gavalas et al 2015): for an undirected graph $\lambda = 6 + \varepsilon + o(1)$ in polynomial-time $V^{O(1/\varepsilon)}$. The arguments for Theorem 2 are the same as in the node reward case.

Walks – We can also consider *walks*, which are like a path but allow nodes and edges to be visited more than once. In this setting, $z_j^k(\rho)$ is no longer binary, and so the proofs for submodularity of the various reward functions must be updated. The argument used for Lemma 5 can be extended to walks by using an oracle which maximizes $\sum_{j=1}^{V} z_j(\rho) c_j$. If $\bar{m}$ is the maximum number of visits to a node, then this approach would give the constant factor guarantee for the greedy sub-problem as $\alpha = \frac{p_s}{\lambda \bar{m}}$. While this model does not have orienteering PTAS, it is straightforward to modify the MILP and heuristic formulations to allow for walks in in this way.

On the other hand, if we define $\mathbb{I}_j(\rho, m) := \mathbb{I}\{z_j(\rho) = m\}$, and the oracle maximizes $\sum_{j=1}^{V} \mathbb{I}_j(\rho, m) c_j(m)$, then we recover the $\frac{p_s}{\lambda}$ guarantee from Lemma 5. It is unclear whether there is an efficient MILP formulation which can act as such an oracle, though it can be posed as a Mixed Integer Program (which is generally much more difficult to solve than a MILP).

6 Approximate Solution Approach to the On-line TSO

Information gathered on-line can be incorporated to solve the on-line TSO problem in a manner similar to the static case. There are two main structural differences between the static and the on-line planning problems: the space of feasible paths for each robot might be different (since nodes cannot be re-visited, due to the definition of paths in Section 2.3), and the survival constraint must be updated appropriately. These changes are handled by modifying the pre-processing step and solving a minimization problem to find p_s^n. In Section 6.1 we outline the on-line algorithm, discuss guarantees on performance in Section 6.3, and characterize the worst-case complexity in Section 6.4.

6.1 On-line Algorithm

At iteration n, the on-line algorithm re-plans paths given a list of surviving robots U_n and the planned paths at the previous iteration, $X_K^{n-1} = \{\rho_k^{n-1}\}_{k=1}^{K}$. The first constraint of the on-line TSO problem requires that the first $n-1$ steps of a new plan be consistent with the past, that is $\rho_k^n(n') = \rho_k^{n-1}(n')$ for $n' \leq n-1$, which implies that the rest of the path cannot contain these nodes. We focus on finding sub-paths which do not contain any nodes already visited, start at $\rho_k^{n-1}(n)$, and end at v_t. Our algorithm consists of three stages: the first is a pre-processing stage which identifies the safest paths for every robot to reach the remaining nodes, the second stage computes the updated survival probability threshold, p_s^n, and the third stage runs the greedy algorithm to select new sub-paths.

6.1.1 Pre-processing

Due to the strict definition of paths in Section 2, robots are not permitted to re-visit nodes. Hence for each robot $k \in U_n$, we must update the maximum probability that robot k can visit each node j in $\mathscr{V}_k^n := \mathscr{V} \setminus \{\rho_k^{n-1}(n')\}_{n'=1}^{n-1}$ given that it starts from node $\rho_k^{n-1}(n)$ and cannot travel through nodes in $\{\rho_k^{n-1}(n')\}_{n'=1}^{n-1}$. We denote this probability as $\zeta_j^{k,n}$, and compute it using Dijkstra's algorithm on the graph $\mathscr{G}_k^n$ which has node set $\mathscr{V}_k^n$, edges in $\mathscr{E}$ with both the source and sink nodes in $\mathscr{V}_k^n$, and each edge given weight $-\log(\omega(e))$. The maximum probability that robot k can reach node v_t is given by $\psi_k := \zeta_{v_t}^{k,n}$.

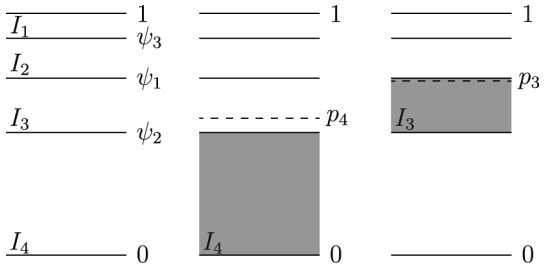


Fig. 5 Illustration of the algorithm for updating the survival probability threshold. The maximum survival probabilities ψ_k and intervals are shown on the left. At the first step, we assume the optimum is in the interval I_4 which has the smallest upper bound (ψ_2), but this assumption is false since $p_4 > \psi_2$. At the second step we proceed to the interval with the next smallest upper bound, I_3, and find that $p_3 \in I_3$. Since the assumption is correct, we know p_3 is the optimum.

6.1.2 Survival threshold update

The on-line version of the TSO problem requires updating the survival probability threshold p_s^n in order to guarantee that, if possible, the risk is distributed fairly and the expected number of robots which reach node v_t safely is $p_s K$. Recall from Section 3.2 that p_s^n is defined as the solution to a minimization problem, and set to one if the problem is infeasible.

If $\psi_k \leq p_s^n$ for any k, this means that there is no path which satisfies the desired survival probability threshold for robot k. In this case, $\{k\}$ is removed from U_n, and ρ_k^n is set to the safest path for robot k to reach node v_t.

Solving for p_s^n is straightforward, as illustrated in Figure 5. The survival probability threshold p_s^n lies in one of at most $|U_n|+1$ intervals between the maximum survival probabilities $\{\psi_k\}_{k\in U_n}$. We begin by sorting the survival probabilities and guessing that the solution is in the interval with the smallest upper bound, evaluating the 'min' operator in the constraint, and then finding the value of p which makes the constraint active. If the result is in the interval we guessed, then we are done and return the result. Otherwise we move to the interval with the next smallest upper bound and repeat. If p is in none of these intervals then the problem is infeasible and we return 1. The complexity of this algorithm is $O(|U_n|(1+\log(|U_n|))) \leq O(K\log(K))$.

6.1.3 Greedy selection

The greedy selection step is quite similar to the static TSO problem, except the survival probability threshold is now p_s^n when solving for the best path for robot k to take (the case $\psi_k \leq p_s^n$ is handled in the previous step). Because each robot has a different graph, we must solve $O(K)$ orienteering problems when selecting each path (one for each robot in U_n), which means the oracle routine is called $O(K^2)$ times during the greedy selection step. While U_n is not empty, we set the node weights appropriately (by choosing the appropriate value for $\delta_j(X_\ell)$ conditioned on U_n and accounting for already selected paths) and find the maximum weight path with survival probability at least p_s^n for each $k \in U_n$. The most valuable path is assigned to its respective robot, that robot is removed from U_n and the loop continues.

6.2 Decentralized Implementations

The presentation above is from a centralized perspective, where a single processing node runs all computations and sends the paths to each robot. In practice, especially for the on-line version of this problem, the robots may not be able to communicate with every other member of the team and may have noisy communications. Greedy algorithms can be decentralized by using 'iterative assignment' (e.g., as used by (Atanasov et al 2015)). In this approach, each robot solves a single-robot sub-problem over its own sub-graph. A leader election is then held to determine which path has the highest discrete derivative. The winner of the election updates its plan and is removed from the pool. Remaining robots repeat the process of planning and determining a leader until every robot has a plan. If the communications graph is connected (meaning there is a way for every robot to communicate with any other robot), then this routine will yield the same result as the centralized counterpart. The communications complexity is (loosely) bounded by K^3 messages containing a path and the value of the path (Lynch 1997). Finally, since each message is small (a path can be represented by $V\log_2(V)$ bits), noisy communications can be mitigated by adding strong error correction and repeated transmission.

In the case where K_d robots cannot communicate with the rest of the team, submodularity implies that the performance degrades by a factor of at most $(K-K_d)/K$. If robots not heard from are presumed 'failed', then our algorithm will make conservative choices, causing the robots to return to the terminal node sooner than if the communications were perfect. This has the added benefit that, for disk connected communications graphs, the communications network will get stronger as robots converge to v_t. This enables the list of surviving robots to be updated and the correct survival probability thresholds computed, so in a sense the communications network will be self-healing.

6.3 On-line Performance Guarantees

Both of the guarantees from the static TSO problem can be extended to the on-line case. The approximation guarantee can be applied because the objective function of the on-line problem inherits submodularity from the objective function of the static problem. Conditioning on U_n will change the value of the constants $\delta_k(X_{L-1})$, but not the basic form of the discrete derivative (in the sense of Lemma 2). The proofs for

Lemma 5 and Theorem 2 depend only on the form of the discrete derivative, which means that we can immediately apply them by exchanging p_s with p_s^n. This means that robots following the paths output by the on-line algorithm will accumulate at least a constant factor $1-\exp(-p_s^n/\lambda)$ of the reward accumulated by the optimal solution to the on-line TSO problem.

The on-line algorithm adapts the survival probability threshold in order to keep the expected number of robots that *actually* reach node v_t as close to p_sK as possible. When it increases p_s^n in response to robot failures, the guarantees that few robots fail become much stronger. As discussed after Lemma 6, the probability that m or fewer robots reach node v_t decreases exponentially as m decreases. So by adapting p_s^n, the on-line algorithm ensures that it is very unlikely for $|U_N|$ to be much smaller than p_sK.

6.4 Complexity

The computational complexity consists of four factors: preprocessing, updating the survival constraints, running the oracle, and updating the node weights. Preprocessing requires running Dijkstra's algorithm for each node and robot which has complexity $O(KV^2\log(V))$. Updating the survival constraints requires sorting at most K elements and running at most K multiplications for the optimization routine, hence has complexity $O(K\log(K)+K)$. The oracle routine is called at most $O(K^2)$ times, since each remaining robot re-plans at every planning step. Finally, the update routine is called after each robot is selected with complexity described in Section 5. The total complexity is then

$$O(KV^2\log(V))+O(K(\log(K)+1))+O(K^2C_O)+O(KC_U).$$

For most applications, the complexity of the oracle will dominate, and so the asymptotic complexity will typically be $O(K^2C_O)$.

This complexity can be improved by using the *accelerated greedy* algorithm, as discussed in (Krause and Golovin 2014). The basic idea is to use the non-increasing property of the discrete derivative to quickly determine whether a given orienteering problem is worth solving. If the marginal benefit of best path for robot k at iteration n is less than the marginal benefit of some robot k' already calculated for iteration $n+1$, then we can skip re-calculating the path for robot k at iteration $n+1$. In the worst-case, this acceleration will not improve the run-time complexity, but in practice it can yield a significant improvements – in the best case, the complexity becomes $O(KC_O)$. Note that this accelerated greedy algorithm only helps when because each robot has a different feasible set, hence we cannot use it for the static algorithm.

Variable	Description
$\omega_r(e)$	Probability robot of type r survives edge e
$p_s(r)$	Survival threshold for each type of robot
$s_n^k(r,\rho)$	One if robot k of type r following path ρ survives edge $(\rho(n-1),\rho(n))$.
$a_n^k(r,\rho)$	One if robot k of type r following path ρ survives to iteration n
$z_j^k(r,\rho)$	One if robot k of type r following path ρ visits node j
$p_j^r(m,X_K)$	Probability of m robots of type r following paths in set X_K visiting node j

Table 2 Summary of notation for the HTSO problem.

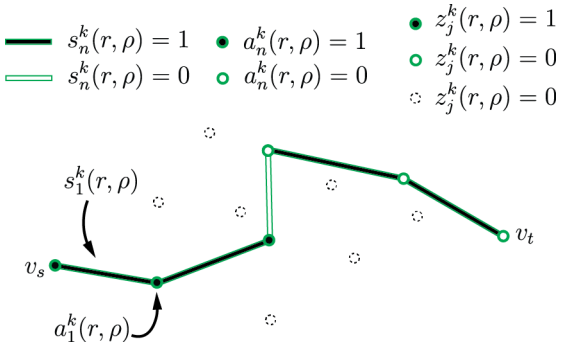


Fig. 6 Illustration of the notation used for the HTSO (note that this is similar to Figure 2, except variables are now indexed by r). Robot k has type r and plans to take path ρ, whose edges are represented by lines. The fill of the lines represent the value of $s_n^k(r,\rho)$. In this example $s_3^k(r,\rho)=0$, which means that $a_3^k(r,\rho)=a_4^k(r,\rho)=a_5^k(r,\rho)=0$. The variables $z_j^k(r,\rho)$ are zero if either the robot fails before reaching node j or if node j is not on the planned path.

7 Heterogeneous Teams

The TSO problem and our algorithm can be readily extended to a heterogeneous setting, where there are R types of robots, and we are given the co-design problem of optimizing over both paths and robot types. In Section 7.1 we outline the problem statement and necessary modifications to notation, in Section 7.2 we give sufficient conditions for the objective function to be submodular and provide an application, and in Section 7.3 we outline the static algorithm and guarantees for the heterogeneous case. In Section 7.4 we describe the on-line HTSO problem and its relationship to the on-line TSO problem.

7.1 Static HTSO Problem

The problem statement for the heterogeneous case is quite similar to the TSO problem, except that there are R edge weight functions and survival constraints, and any variables which previously were a function of path (e.g. s_n^k, z_j^k, and a_n^k) are now a function of path and robot type. The notation for the HTSO problem is summarized in Table 2 and Figure 6.

Given a set $X_K = \{r_k, \rho_k\}_{k=1}^K$, define the R element vector $V_j(X_K)$ element-wise as the number of robots of type r which visit node j by iteration N:

$$[V_j(X_K)]_r := \sum_{k=1}^{K} z_j^k(r, \rho_k).$$

Let the value of visiting node j with V_j visits be given by the function $H_j : \mathbb{Z}_+^R \to \mathbb{R}_+$. The HTSO problem is defined formally as:

Heterogeneous Team Surviving Orienteers Problem: Given a graph $\mathscr{G}$, edge weights ω_r, survival probability thresholds $\{p_s(r)\}_{r=1}^R$ and team size K, choose robot types and paths in order to maximize the expected reward accumulated by the team:

$$\begin{aligned}
\underset{r_1,\rho_1,\ldots,r_K,\rho_K}{\text{maximize}} \quad & \sum_{j=1}^{V} \mathbb{E}\left[H_j\left(V_j\left(\{r_k,\rho_k\}_{k=1}^K\right)\right)\right] \\
\text{subject to} \quad & \mathbb{P}\{a^k_{|\rho_k|}(r_k,\rho_k) = 1\} \geq p_s(r_k) \qquad k = 1,\ldots,K \\
& \rho_k(0) = v_s \qquad k = 1,\ldots,K \\
& \rho_k(|\rho_k|) = v_t \qquad k = 1,\ldots,K
\end{aligned}$$

The objective is to choose a team of K feasible type/path pairs which maximize the expected cumulative rewards. The first constraint enforces the survival probability constraint for each path, the second and third constraints enforce that each path starts at v_s and end at v_t. Note that the reward function H_j maps a *vector* number of visits to a reward, rather than in the TSO problem, where the reward function h_j maps a scalar number of visits to a reward.

7.2 Submodularity and Applications

We begin by characterizing when the objective function is submodular:

Lemma 7 (Submodularity of the HTSO problem objective) *Let e_r be r^{th} canonical basis vector of $\mathbb{R}^R$. Given an objective function*

$$J(X_K) = \sum_{j=1}^{V} \mathbb{E}\left[H_j\left(V_j(X_K)\right)\right],$$

define the set function

$$\delta_j^r(X) = \mathbb{E}\left[H_j\left(V_j(X) + e_r\right) - H_j\left(V_j(X)\right)\right].$$

The objective function has discrete derivative with respect to (r_L, ρ_L) at partial solution $X_{L-1} = \{r_\ell, \rho_\ell\}_{\ell=1}^{L-1}$

$$\Delta J((r_L,\rho_L) \mid X_{L-1}) \sum_{j=1}^{V} \mathbb{E}\left[z_j^L(r_L,\rho_L)\right] \delta_j^{r_L}(X_{L-1}),$$

and is submodular $-\delta_j^r(X)$ is a monotone function of X for all j, r.

Proof The random variable $z_j^L(r_L,\rho_L)$ is independent of each element of the random vector $V_j(X_{L-1})$. Hence from the definition of the discrete derivative and the tower property we have for $x_L = (r_L, \rho_L)$

$$\begin{aligned}
\Delta J(x_L \mid X_{L-1}) &= \sum_{j=1}^{V} \mathbb{E}\left[H_j\left(V_j(X_{L-1} \cup (r_L,\rho_L))\right)\right] \\
&\quad - \mathbb{E}\left[H_j\left(V_j(X_{L-1})\right)\right] \\
&= \sum_{j=1}^{V} \mathbb{E}[z_j^L(r_L,\rho_L)]\mathbb{E}\left[H_j\left(V_j(X_{L-1}) + e_{r_L}\right)\right] \\
&\quad - \left(\mathbb{E}[z_j^L(r_L,\rho_L)]\right)\mathbb{E}\left[H_j\left(V_j(X_{L-1})\right)\right],
\end{aligned}$$

which upon simplification yields the first statement of the lemma. By definition, a set function is submodular if the negative of its discrete derivative is a monotone function. Since $\mathbb{E}[z_j^L(\cdot)] \geq 0$ and the sum of monotone functions is monotone, we have that the negative of the discrete derivative is monotone (hence the objective function is submodular). □

We can use Lemma 7 to immediately extend the settings described in Section 4 to their *uncoupled* analogues, where each robot type has its own single or multi-visit reward function, and the total reward is the sum of the rewards accumulated by each type. We can also consider *coupled* reward functions, as described next.

Consider a scenario where robot types correspond to sensor resolutions, and the information gained about a node is determined by only the highest resolution data recorded about the node. Let d_j^r the information gained about node variable j by a sensor of type r. The *best-visit reward* function is:

$$\sum_{j=1}^{V} \mathbb{E}\left[H_j(V_j(X_K))\right] = \sum_{j=1}^{V} \mathbb{E}\left[\max_k d_j^{r_k} z_j^k(r_k,\rho_k)\right]$$

Given a partial solution $X_{L-1} = \{r_\ell, \rho_\ell\}_{\ell=1}^{L-1}$, we write the probability that at least one robot of type r will visit node j as

$$p_{j,r}(X_{L-1}) = 1 - p_j^r(0, X_{L-1}),$$

and write the probability no robot of type r or less visits node j as

$$\bar{p}_{j,r}^{+}(X_{L-1}) = \prod_{r'=1}^{r} \left(1 - p_{j,r'}(X_{L-1})\right).$$

Without loss of generality, we assume that sensors with smaller type have superior resolution. The reward function depends only on the first visit for robots of a given type:

$$\sum_{j=1}^{V} \mathbb{E}\left[H_j\left(V_j(X_K)\right)\right] = \sum_{j=1}^{V} \sum_{r=1}^{R} d_j^r p_{j,r}(X_K) \bar{p}_{j,r-1}^{+}(X_K).$$

Lemma 8 (Submodularity of best-visit rewards) *Let* $d_j^r \geq \sum_{\hat{r}=r+1}^{R} d_j^{\hat{r}} \geq 0$. *Then the reward function at the solution set* $X_K = \{r_k, \rho_k\}_{k=1}^{K}$,

$$\sum_{j=1}^{V} \mathbb{E}\left[H_j(V_j(X_K))\right] = \sum_{j=1}^{V} \mathbb{E}\left[\max_k d_j^{r_k} z_j^k(r_k, \rho_k)\right],$$

is a normalized, non-negative, monotone, and submodular function with discrete derivative with respect to (r_L, ρ_L) *at partial solution* $X_{L-1} = \{r_\ell, \rho_\ell\}_{\ell=1}^{L-1}$:

$$\sum_{j=1}^{V} \mathbb{E}[z_j^L(r_L, \rho_L)] \left(d_j^{r_L} \bar{p}_{j,r_L}^{+}(X_{L-1}) - \sum_{r=r_L+1}^{R} d_j^r p_{j,r}(X_{L-1}) \bar{p}_{j,r-1}^{+}(X_{L-1}) \right).$$

Proof The normalized, non-negative and monotone properties follow immediately from the positivity of d_j^r, z_j^r and the fact that the maximum function is monotone. From the definition of the best-visit reward function we have

$$\delta_j^{r_L}(X) = d_j^{r_L} \bar{p}_{j,r_L}^{+}(X) - \sum_{r=r_L+1}^{R} d_j^r p_{j,r}(X) \bar{p}_{j,r-1}^{+}(X).$$

The first term can be interpreted as the negative probability that a robot of type r_L following path ρ_L is the best robot to visit the nodes in path ρ_L, and the second term is the reduction in the probability that robots in X with type $r > r_L$ will be the best type to visit nodes in path ρ_L. Consider two sets X and $Y = X \cup (\tilde{r}, \tilde{\rho})$. If $p_{j,\tilde{r}}(X) = 1$, then trivially $\delta_j^{r_L}(X) = \delta_j^{r_L}(Y)$. Otherwise, $p_{j,r}(Y) \geq p_{j,r}(X)$ with equality if $r \neq \tilde{r}$. For $r \geq \tilde{r}$, we have $\bar{p}_{j,r}^{+}(X)(1 - p_{j,\tilde{r}}(Y))/(1 - p_{j,\tilde{r}}(X)) = \bar{p}_{j,r}^{+}(Y)$, and otherwise $\bar{p}_{j,r}^{+}(X) = \bar{p}_{j,r}^{+}(Y)$. Now we show that $\delta_j^{r_L}(X) \geq \delta_j^{r_L}(Y)$ by considering three cases:

1. $(\tilde{r} \leq r_L)$: From the definition of $\delta_j^{r_L}(X)$ we have

   $$\delta_j^{r_L}(X) \geq \frac{1 - p_{j,\tilde{r}}(Y)}{1 - p_{j,\tilde{r}}(X)} \delta_j^{r_L}(X) = \delta_j^{r_L}(Y)$$

   The inequality is due to the fact that $\frac{1-p_{j,\tilde{r}}(Y)}{1-p_{j,\tilde{r}}(X)} \leq 1$, and the equality because $r_L \geq \tilde{r}$.
2. $(\tilde{r} > r_L, p_{j,\tilde{r}}(X) = 0)$: We have from the definition of $\delta_j^{r_L}(X)$:

   $$\begin{aligned}
   \delta_j^{r_L}(X) &= d_j^{r_L} \bar{p}_{j,r_L}^{+}(X) - \sum_{r=r_L+1}^{\tilde{r}-1} d_j^r p_{j,r}(X) \bar{p}_{j,r-1}^{+}(X) - 0 \\
   &= d_j^{r_L} \bar{p}_{j,r_L}^{+}(Y) - \sum_{r=r_L+1}^{\tilde{r}-1} d_j^r p_{j,r}(Y) \bar{p}_{j,r-1}^{+}(Y) - 0 \\
   &\geq d_j^{r_L} \bar{p}_{j,r_L}^{+}(Y) - \sum_{r=r_L+1}^{R} d_j^r p_{j,r}(Y) \bar{p}_{j,r-1}^{+}(Y) \\
   &= \delta_j^{r_L}(Y)
   \end{aligned}$$

   where we use the properties introduced above for each line.
3. $(\tilde{r} > r_L, p_{j,\tilde{r}}(X) > 0)$: Define $\gamma \geq 0$ such that $p_{j,\tilde{r}}(X)(1+\gamma) = p_{j,\tilde{r}}(Y)$. Then we have

   $$\begin{aligned}
   \delta_j^{r_L}(X) - \delta_j^{r_L}(Y) &= \gamma d_j^{\tilde{r}} p_{j,\tilde{r}}(X) p_{j,\tilde{r}-1}(X) \\
   &\quad - \sum_{r=\tilde{r}+1}^{R} d_j^r p_{j,r}(X) \bar{p}_{j,r-1}^{+}(X) \left(\frac{\gamma p_{j,\tilde{r}}(X)}{1 - p_{j,\tilde{r}}(X)} \right) \\
   &\geq \gamma p_{j,\tilde{r}}(X) \bar{p}_{j,\tilde{r}}^{+}(X) \left(d_j^{\tilde{r}} - \sum_{r=\tilde{r}+1}^{R} d_j^r p_{j,r}(X) \right) \\
   &\geq 0
   \end{aligned}$$

   The first and second statements are due to the definition of γ and the given identities, and the final inequality follows from the definition of $d_j^{\tilde{r}}$.

This implies that $-\delta_j^{r_L}(X)$ is a monotone function of X, which implies that the best-reward objective function is submodular. □

An example which satisfies the requirement that $d_j^r \geq \sum_{\hat{r}=r+1}^{R} d_j^{\hat{r}}$ is an imaging scenario, where r corresponds to observation distance. The area covered by a picture is proportional to the square of distance, and so a small distance implies a high density of pixels (i.e. high resolution). Another example of a coupled reward function is informative path planning where each robot has a different sensor quality, and the goal is to minimize the entropy of the posterior distribution of node variables Y_j, similar to the multi-visit example from Section 4.

7.3 Algorithm

The algorithm for the static HTSO problem proceeds in an identical manner as for the TSO problem, except that at each step we must consider each of the R types of robots. We begin by computing the maximum probability that a node can be visited by a robot of type r, which we denote ζ_j^r. Then we solve R orienteering problems to find the (approximate) best type/path pair to add. Using Lemma 5 we can guarantee that each path is within constant factor $p_s(r)/\lambda$ of the optimal path (for a fixed robot of type r), and hence the best path/pair is within constant factor $\min_r p_s(r)/\lambda$ of the optimal path/type pair for the greedy step. After choosing the path/type pair to add, we update the reward function appropriately and continue on to the next iteration.

Updates for best-visit reward functions – Recall from Lemma 8 that for the best-visit reward function

$$\delta_j^{r_L}(X_{L-1}) = d_j^{r_L} \bar{p}_{j,r_L}^{+}(X_{L-1}) - \sum_{r=r_L+1}^{R} d_j^r p_{j,r}(X_{L-1}) \bar{p}_{j,r-1}^{+}(X_{L-1}),$$

which can be computed recursively by updating the visit and non-visit probabilities $p_{j,r}$ and $\bar{p}^{+}_{j,r}$. When (r_L, ρ_L) is added to X_{L-1}, we update the visit probabilities for $j \in \rho_L$ as

$$p_{j,r_L}(X_L) = 1 - (1 - p_{j,r_L}(X_{L-1}))(1 - \mathbb{E}[z^L_j(r_L, \rho_L)]),$$

and the non-visit probabilities for $j \in \rho_L$, $r \geq r_L$ as

$$\bar{p}^{+}_{j,r}(X_L) = \bar{p}^{+}_{j,r}(X_{L-1})(1 - \mathbb{E}[z^L_j(r_L, \rho_L)]).$$

The complexity of updating the probabilities is $O(VR)$, and updating the node weights is $O(VR^2)$.

Guarantees We can easily get a $1 - e^{-\min_r p_s(r)/\lambda}$ constant factor guarantee by using the same approach as was used for Theorem 2. Using the remark following Theorem 1, we can provide a tighter guarantee at run-time by computing the approximation factors α_ℓ for each step of the greedy algorithm as follows. If $\hat{\rho}^r_\ell$ is the best path found for type r at step ℓ, the optimum is bounded by

$$J^{UB}_\ell = \max_r \left(1 - e^{-p_s(r)/\lambda}\right)^{-1} \sum_{j=1}^{V} \mathbb{E}\left[H_j(V_j(\hat{X}_{\ell-1} \cup \hat{\rho}^r_\ell))\right],$$

and so the approximation factor for step ℓ is bounded by the ratio of the upper bound on the optimum to the value of the approximate greedy set $\hat{X}_\ell$.

$$\alpha_\ell \geq \sum_{j=1}^{V} \mathbb{E}\left[H_j(V_j(\hat{X}_{\ell-1} \cup \rho^{r_\ell}_\ell))\right] / J^{UB}_\ell,$$

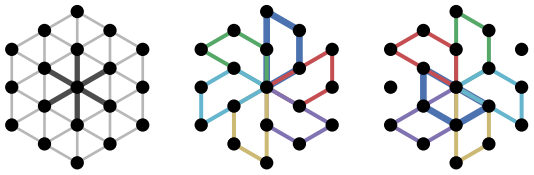
which in practice will be tighter than the $1 - e^{-\min_r p_s(r)/\lambda}$ guarantee.

7.4 On-line Heterogeneous TSO problem

We can also consider the on-line heterogeneous TSO problem, where the robot types remain fixed, but the paths can be updated. We compute an updated survival probability threshold for each robot type in a manner identical to that described in Section 6, where U_n is replaced by $U_n(r)$, the list of surviving robots of type r. Because robot types are not recomputed on-line (they are fixed after iteration $n = 0$), after iteration 0 the computational complexity and algorithm will be the same as for the on-line TSO problem.

8 Numerical Experiments

In this section we provide numerous numerical experiments over a variety of synthetic and real-world graphs to characterize the performance of our algorithm in the settings described above. In Section 8.1 we verify that the theoretical bounds hold for a highly structured problem (where we have access to the optimal solution). In Section 8.2 we characterize the empirical approximation factor over a wide range of survival probability thresholds. In Sections 8.3 and 8.4 we consider real-world scenarios involving classification during a storm and information gathering in hostile environments. We demonstrate the effectiveness of simple heuristics for the orienteering problem for very large problems in Section 8.5. Finally we consider the on-line and heterogeneous variants in Sections 8.6 and 8.7, respectively. Unless otherwise stated, we pose the orienteering problem as a MILP and use the Gurobi solver with tolerance 10^{-4} as the oracle routine.



(a) Graph $\mathscr{G}$ (b) Optimal X^*_6 (c) Greedy $\bar{X}_6$

Fig. 7 (a) Example of a team surviving orienteers problem with depot in the center. Thick edges correspond to survival probability 0.98, light edges have survival probability 0.91. (b) Optimal paths for survival threshold $p_s = 0.70$ and $K = 6$. (c) Greedy paths for the same problem.

8.1 Verification of Bounds

We consider a TSO problem (where we seek to maximize the expected number of nodes visited by a homogeneous team) on the graph shown in Figure 7(a): the central starting node has 'safe' transitions to six nodes, which have 'unsafe' transitions to the remaining twelve nodes. Due to the symmetry of the problem we can compute an optimal policy for a team of six robots, which is shown in Figure 7(b). The output of the greedy algorithm is shown in Figure 7(c). The `GreedySurvivors` solution comes close to the optimal, although the initial path planned (shown by the thick dark blue line) does not anticipate its impact on later paths. The expected number of nodes visited by robots following optimal paths, greedy paths, and the upper bound are shown in Figure 8. Note that the upper bound is close to the optimal, even for small teams, and that the `GreedySurvivors` performance is nearly optimal.

8.2 Empirical Approximation Factor

We compare our algorithm's performance against an upper bound on the optimal value to get a sense of the empirical versus theoretical approximation ratios. We use an exact solver for the orienteering problem, and generate instances on a complete undirected graph (meaning there is an edge between every pair of nodes) with $V = 65$ nodes and uniformly distributed edge weights in the interval $[0.3, 1)$. The

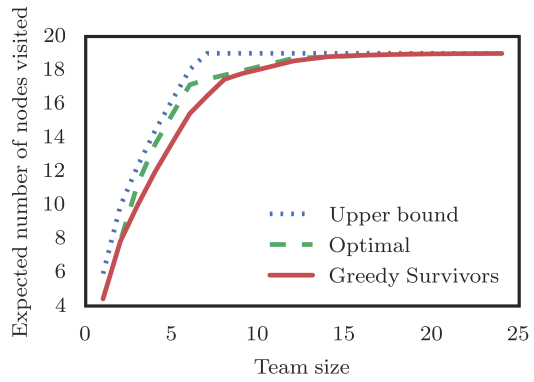


Fig. 8 Performance comparison for the example in Figure 7(a). The optimal value is shown in green and the GreedySurvivors value is shown in red. The upper bound on the optimum from Theorem 2 is shown by the dotted line.

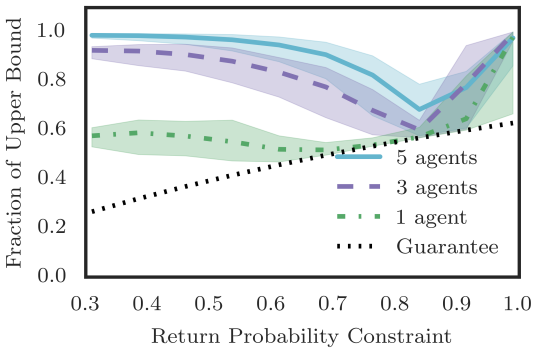


Fig. 9 Ratio of actual result to upper bound for a 65 node complete graph. The team size ranges from 1 (at the bottom) to 5 (at the top), and in all cases a significant fraction of the possible reward is accumulated even for small p_s.

upper bound used for comparison is the smallest of 1) the number of nodes which can be reached within the budget, 2) the constant-factor guarantee times our approximate solution, and 3) the guarantee from solving the problem with an oversized team (from Theorem 2). The average performance (relative to the upper bound) along with the total range of results are shown in Figure 9, with the function $1-e^{-p_s/\lambda}$ drawn as a dashed line. As shown, the approximation factor converges to the optimal as the team size grows. The dip around $p_s = 0.85$ is due to looseness in the bound and the fact that the optimum is not yet reached by the greedy routine.

8.3 Information Gathering

Consider a setting where robotic sensors are used to gather information about a physical phenomena (e.g., health of coral reefs, algae blooms) in the Coral Triangle, an ecologically significant region surrounding Indonesia. Figure 10 shows 108 marine protected areas listed by (Cros et al 2014). Each area is marked by an 'X', and areas are contained in larger regions highlighted by boxes (corresponding to relatively similar environments). One commonly proposed platform for long-duration environmental monitoring are underwater gliders (**?**), which have limited communication. Hence we consider the off-line TSO problem, as uncertain communications would otherwise lead to overly conservative actions (as discussed in Section 6.2).

We represent this environment using a graph with nodes corresponding to a fine uniform grid (with respect to distance). Neighboring nodes are connected (including diagonals) with edges, and we use piracy incident data (International Chamber of Commerce: Commercial Crime Services 2017) and a Poisson model similar to (Vaněk et al 2013) to calculate the risk of traversing along the edge. Since we are not running the on-line algorithm we can simplify the problem by only considering the graph induced by the nodes corresponding to marine protected regions and edges corresponding to the shortest (maximum survival probability) path between these nodes. We assume a Gaussian measurement model, where the marginal information gain of the mth visit to node j is $\frac{1}{2}\log(1+\sigma_j^{-2}(1+m)^{-1})$, where σ_j^2 is the noise variance for measurements at node j.

Figure 11 compares the performance of a team of 25 robots with survival probability threshold 0.64 when using paths computed by a MILP formulation (with $\lambda = 1.5$) and using the Variable Neighborhood Search (VNS) heuristic with depth 10 (implemented as part of HeuristicLab (Wagner and Affenzeller 2005)). The two approaches are compared over 30 scenarios with σ_j^2 drawn from the uniform distribution over $[0.1, 1.0]$, and objective is plotted relative to the information gained if every robot visited every node. The mean is shown as a solid line, and total range is shown using the dotted lines. The two approaches provide similar quality answers, though for these problems the VNS approach performs 8% better on average. The MILP formulation takes between 2.59 seconds and 71.01 seconds to find a path, with an average of 14.9 seconds and standard deviation of 13.6 seconds. The VNS approach takes between 17.7 and 26.8 seconds to find a path, with an average of 21.0 seconds and standard deviation of 1.52 seconds. Hence the VNS heuristic (with the given parameters) provides a higher quality solution in a more consistent, though longer amount of time compared to the MILP approach.

8.4 Classification During a Storm

As the problems become large, solving MILP using personal computers becomes impractical. However cloud computing offers a low-cost way of solving even moderately large MILP problems. In this scenario, we demonstrate the effectiveness of a 64-core cluster with 200GB of RAM in solving a problem with 225 nodes and 25 robots.

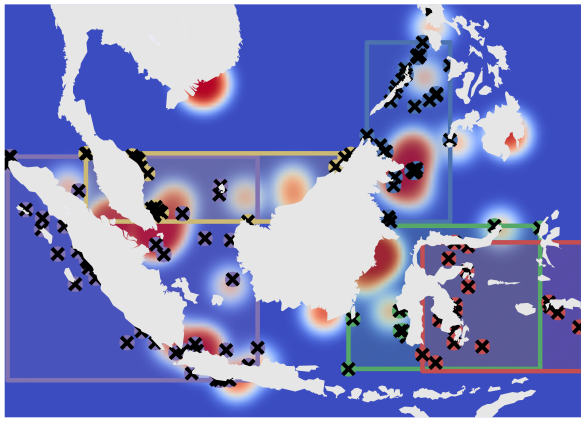


Fig. 10 Illustration of an ocean monitoring scenario. Various regions in the Coral Triangle are outlined by boxes, sites to visit within each region are marked by 'X', and the heatmap indicates the risk of robot failure inferred from piracy incidents. Data is from the Coral Triangle Atlas (Cros et al 2014) and IMB Piracy Reporting Centre (International Chamber of Commerce: Commercial Crime Services 2017).

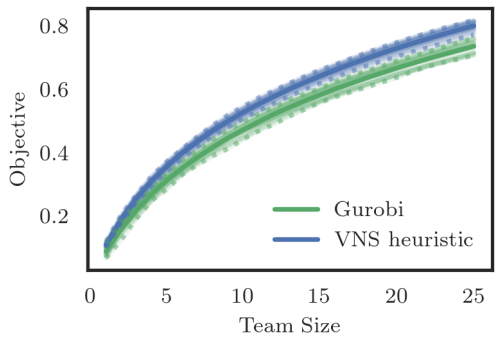


Fig. 11 Normalized information gained by team of 25 robots when using a VNS heuristic and the MILP formulation. The mean and range are shown as solid and dotted lines, respectively. Note that depending on the parameters given, the VNS heuristic quickly produces quality solutions.

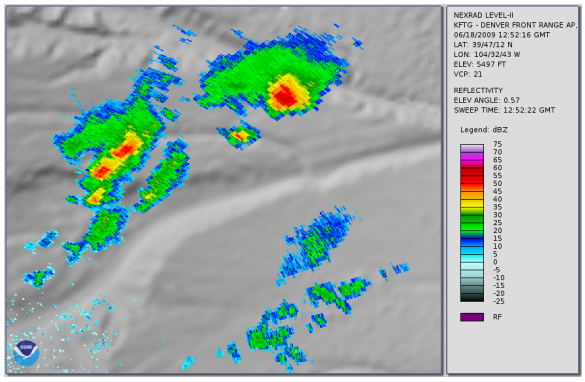


Fig. 12 Illustration of the "base reflectivity" of a storm, which can be used to infer the danger to robots. Data from the NOAA NEXRAD level II dataset, visualization courtesy the Weather and Climate Toolkit (NOAA National Weather Service Radar Operations Center 1991).

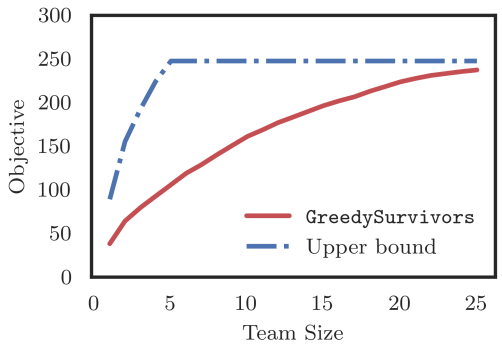


Fig. 13 Reduction in posterior variance (using the Haldane prior) for the storm classification scenario. Note that the team of 25 robots achieves 95.8% of the maximum award available, essentially solving the problem.

Consider a setting where robots travel through a storm and must classify some property of each node (e.g., whether a piece of infrastructure is operational or if there are people present). We use data from the NOAA NEXRAD system (NOAA National Weather Service Radar Operations Center 1991), shown in Figure 12. We assume a team of relatively simple robots is used, which in turn limits the communications available. This is because (1) wireless quality degrades with precipitation, and (2) turbulence limits the ability to steer directional antenna. Accordingly, we consider the offline TSO problem in this setting.

Robots seek to classify a single property for each node using a Haldane prior (as discussed in Section 4.2). Recall that this means the value for the mth visit to node j is $h_j(m) = \left(\frac{1}{4} - \frac{1}{4(m+1)}\right)$.

The risk a storm poses to robots was analyzed by (Zhang et al 2017) with a survival model very similar to ours (i.e. the product of Bernoulli random variables over each edge in a graph). For simplicity we assume that survival probability is inversely proportional to "base reflectivity" (the proportion of radar energy reflected by the weather system), though in practice one would want to use a more sophisticated approach such as the ensemble models used in (Zhang et al 2017).

We place 225 sites in a grid and compute edge weights across the straight-line connection between sites. We use a team of 25 robots with $p_s = 0.8$. Paths are found using the Gurobi solver with tolerance $\lambda = 1.5$, and each path takes an average of 55.4 seconds to find. Figure 13 shows the objective achieved by our team along with an upper bound (which is the smaller of the constant factor guarantee and the upper bound computed using ζ_j and k only). The team of 25 robots achieves 95.8% of the maximum reduction in posterior variance possible.

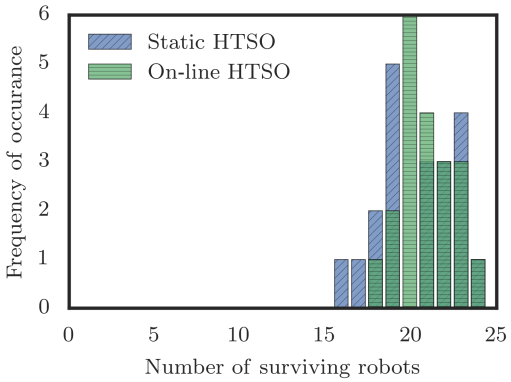


Fig. 14 Histogram comparing surviving robots with and without re-planning for 20 trials with $K = 25$ and $p_s = 0.8$. Note that the expected number of surviving robots at the initial iteration is 20.

8.5 Large Scale Performance

As the problems become very large, it becomes impractical to solve them using a MILP approach. We demonstrate the usefulness of simple heuristics in solving such large problems by planning $K = 25$ paths for synthetic complete undirected graphs of various sizes. We use two `Orienteering` routines: the mixed integer formulation from (Kara et al 2016) with Gurobi's MILP solver, and an adapted version of the open source heuristic developed by the authors of (Vansteenwegen et al 2009). For the cases where we have comparison data (up to $V = 100$ nodes) the team using paths computed using the heuristic achieves an average of 98.2% the reward of the MILP algorithm. Even very large problems, e.g. 25 robots on a 900 node graph, can be solved in approximately an hour with the heuristic on a machine that has a 3GHz i7 processor using 8 cores and 64GB of RAM.

8.6 Benefits of Re-planning

We demonstrate the benefits of an on-line approach using a synthetic complete undirected planar graph with 50 nodes and 25 homogeneous robots that have survival probability threshold $p_s = 0.8$. Figure 14 shows the histogram of the number of surviving robots at iteration N for 20 trials. Without re-planning, 45% of trials finished with fewer than 20 survivors, versus 15% when paths were re-planned. Conditioned on the event that the final budget is negative, the expected value of the overrun is reduced by 25% from -1.78 (without re-planning) to -1.33. A side effect of re-planning in the worst-case regime is that the average reward is reduced, but the effect is not significant. For the same set of trials discussed above, the average reward with re-planning was 93% of the reward gained without re-planning.

8.7 Heterogeneous Team with Best-visit Rewards

Using heterogeneous teams can give substantial advantages over homogeneous teams of the same size. We demonstrate this by comparing teams of 20 robots on a complete undirected graph with 35 nodes in a setting where robots with higher sensor qualities have more stringent survival probability thresholds. Namely, the sensing qualities are (1, 2, 4, 8, 16), and the survival constraints are respectively (0.30, 0.45, 0.6, 0.75, 0.90). Figure 15 shows the expected cumulative reward as a function of the team size for the heterogeneous team (solid line) and best homogeneous teams among the five robot types (dashed line). For teams of 20 robots, the heterogeneous team vastly outperforms the homogeneous teams, in this example it is expected to achieve 186% the reward of the best homogeneous team (and 851% of the worst). We note that this is for the static HTSO problem, meaning that the paths are not re-planned based on survival events.

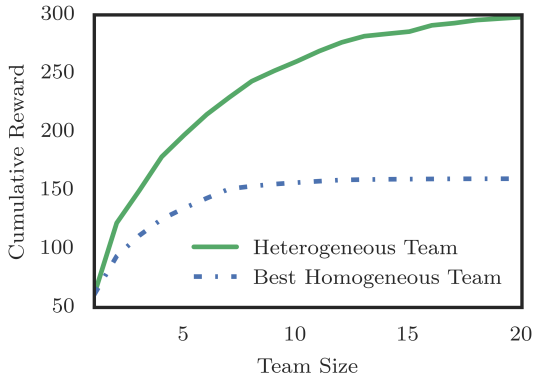


Fig. 15 Cumulative reward of a heterogeneous team versus a homogeneous team for best-visit rewards. The graph has $V = 35$.

9 Conclusions

In this paper we formulate the *Team Surviving Orienteers problem*, where we are asked to plan a set of paths that maximizes the expected cumulative reward at nodes visited while guaranteeing that every robot survives with probability at least p_s. What sets this problem apart from previous work is the notion of risky traversal, where a robot might not complete its planned path. This creates a complex, history-dependent coupling between the edges chosen and the distribution of nodes visited, which precludes the application of existing approaches available for the traditional orienteering problem. We give numerous applications where the objective function is submodular in the paths chosen, present a linearization for a class of submodular functions, and use

it to develop the `GreedySurvivors` algorithm which has polynomial-time complexity with a constant-factor guarantee that the returned objective is lower bounded by $(1 - e^{-p_s/\lambda})$OPT, where OPT is the optimum.

We then provide an algorithm for the on-line TSO problem, where at iteration n the list of surviving robots and edges traversed is given and we are asked to re-plan the paths in order to maximize the expected cumulative rewards, while observing an updated survival probability thresholds. We give an algorithm which finds a near-optimal set of paths which satisfy the constraints in polynomial-time. Third, we outline how to extend the TSO to the co-design problem where we are asked to assemble a heterogeneous team of robots. We give an algorithm for the HTSO with complexity that grows only linearly in the number of robots types relative to the complexity of the TSO. Finally we demonstrate the effectiveness of our approach using numerical experiments for a variety of settings. Our experiments support the theoretical performance guarantees, and we demonstrate the efficiency of our algorithm for large graphs by solving a TSO with 25 robots and 900 nodes. Finally we illustrate the performance gains of running the on-line version of the TSO and of using heterogeneous teams.

Future Work There are numerous directions for future work. First, we are interested in more general models for the reward functions, such as time-varying rewards, or when the reward functions between nodes are correlated. Our analysis assumes that the reward functions at each node are independent, which does not capture applications such as informative path planning with covariance between node variables.

Second, we are interested in more general models for traversal, such as allowing walks or more complicated survival models. As mentioned in Section 5.5 it is possible to recover the $1-\exp(-p_s/\lambda)$ guarantee if we use a solver that can assign different rewards based on the number of visits to a node, but this becomes a MIP rather than a MILP, which is significantly more difficult to solve. We leave a more detailed investigation of this to future work. More general survival models (e.g., non-binary survival states) should also be considered. The approach taken by (Campbell et al 2011) may be a good starting point and may also allow for dependence between the survival variables s_n^k.

Third, we are interested in applying the notion of *adaptive submodularity* (Golovin and Krause 2011), to the on-line TSO problem. In this case, we would optimize over *policies* of paths, which would specify which path to take based on all of the survival outcomes. Using such an approach will require extending the setting in (Golovin and Krause 2011) to allow for sequences of outcomes (the s_n^k variables) to be associated with each 'item' in the feasible set.

Finally, we are interested in investigating numerous related problems, such as the dual problem (where the survival probabilities are given as the objective, and the visit probabilities as the constraint), or the knapsack variant where we are given a cost function and must assemble a team that satisfies the cost constraint (rather than the cardinality constraint used in the present work).

Acknowledgements The authors would like to thank Federico Rossi, Edward Schmerling, and Sumeet Singh for their comments and insights which led to tighter analysis.

References

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

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

Chao IM, Golden BL, Wasil EA (1996) The team orienteering problem. European Journal of Operational Research 88(3):464–474

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

Chekuri C, Korula N, Pál M (2012) Improved algorithms for orienteering and related problems. ACM Transactions on Algorithms 8(3):23:1–23:27

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

Chen XH, Dempster AP, Liu JS (1994) Weighted finite population sampling to maximize entropy. Biometrika 81(3):457–469

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, Beare D (2014) The Coral Triangle Atlas: An integrated online spatial database system for improving coral reef management. PLoS ONE 9(6):1–7

Gavalas D, Konstantopoulos C, Mastakas K, Pantziou G, Vathis N (2015) Approximation algorithms for the arc orienteering problem. Information Processing Letters 115(2):313–315

Golden BL, Yee JR (1979) A framework for probabilistic vehicle routing. AIIE Transactions 11(2):109–112

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

Golovin D, Krause A (2011) Adaptive submodularity: Theory and applications in active learning and stochastic optimization. Journal of Artificial Intelligence Research 42:427–486

Gunawan A, Lau HC, 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, Ravi R (2012) Approximation algorithms for stochastic orienteering. In: ACM-SIAM Symp. on Discrete Algorithms

Haldane JBS (1932) A note on inverse probability. Mathematical Proceedings of the Cambridge Philosophical Society 28(1):55–61

Hollinger GA, Sukhatme GS (2014) Sampling-based robotic information gathering algorithms. Int Journal of Robotics Research 33(9):1271–1287

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

Jorgensen S, Chen RH, Milam MB, Pavone M (2017) The team surviving orienteers problem: Routing robots in uncertain environments

with survival constraints. In: IEEE Int. Conf. on Robotic Computing
Kara I, Biçakci PS, Derya T (2016) New formulations for the orienteering problem. Procedia Economics and Finance 39:849–854
Krause A, Golovin D (2014) Submodular function maximization. In: Tractability: Practical approaches to hard problems, Cambridge University Press
Laporte G, Louveaux F, Mercure H (1989) Models and exact solutions for a class of stochastic location-routing problems. European Journal of Operational Research 39(1):71–78
Lynch NA (1997) Distributed Algorithms, 1st edn. Morgan Kaufmann
Nemhauser GL, Wolsey LA, Fisher ML (1978) An analysis of approximations for maximizing submodular set functions–I. Mathematical Programming 14(1):265–294
NOAA National Weather Service Radar Operations Center (1991) NOAA next generation radar (NEXRAD) level II base data
Pillac V, Gendreau M, Guéret C, Medaglia AL (2013) A review of dynamic vehicle routing problems. European Journal of Operational Research 225(1):1–11
Psaraftis HN, Wen M, Kontovas CA (2016) Dynamic vehicle routing problems: Three decades and counting. Networks 67(1):3–31
Singh A, Krause A, Guestrin C, Kaiser WJ (2009) Efficient informative sensing using multiple robots. Journal of Artificial Intelligence Research 34:707–755
Stewart WR, Golden BL (1983) Stochastic vehicle routing: A comprehensive approach. European Journal of Operational Research 14(4):371–385
Vaněk O, Jakob M, Hrstka O, 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, Van Oudheusden D (2009) Iterated local search for the team orienteering problem with time windows. Computers & Operations Research 36(12):3281–3290
Vansteenwegen P, Souffriau W, Van Oudheusden D (2011) The orienteering problem: A survey. European Journal of Operational Research 209(1):1–10
Varakantham P, Kumar A (2013) Optimization approaches for solving chance constrained stocahstic orienteering problems. In: Proc. Int. Conf. on Algorithmic Decision Theory
Wagner S, Affenzeller M (2005) HeuristicLab: A generic and extensible optimization environment. In: Adaptive and Natural Computing Algorithms, Springer
Wei K, Iyer RK, Bilmes JA (2014) Fast multi-stage submodular maximization. In: Int. Conf. on Machine Learning
Zhang B, Tang L, Roemer M (2017) Probabilistic planning and risk evaluation based on ensemble weather forecasting. IEEE Transactions on Automation Sciences and Engineering PP(99):1–11
Zhang H, Vorobeychik Y (2016) Submodular optimization with routing constraints. In: Proc. AAAI Conf. on Artificial Intelligence

Appendix

In the following we prove the technical lemma stated in the background section. We start the proof by considering a sequence of Poisson binomial distributions $f_0, f_1, \ldots$. The parameters of the nth distribution are denoted as $\{p_{n,k}\}_{k=1}^{K}$, with $p_{n,1} \leq p_{n,2} \leq \cdots \leq p_{n,K}$. The parameters of the $n+1$st distribution are

$$\{p_{n+1,k}\}_{k=1}^{K} = \left\{ \frac{p_{n,1}+p_{n,K}}{2}, \{p_{n,k}\}_{k=2}^{K-1}, \frac{p_{n,1}+p_{n,K}}{2} \right\},$$

that is, the largest and smallest event probabilities of the nth distribution are averaged to form the $n+1$st distribution. Note that we re-sort the parameters after constructing them from the nth distribution, so it is still true that $p_{n+1,j} \leq p_{n+1,k}$ for $j \leq k$.

It is easy to verify that this sequence converges to the binomial distribution with parameters K and $p = \frac{1}{K}\sum_{k=1}^{K} p_k$. We are interested in showing that the tails of the sequence become heavier as n increases. We begin by making some basic observations:

Lemma 9 *Define*

$$\varepsilon_n := \frac{1}{2}(p_{n,K} - p_{n,1})$$

and

$$\bar{p}_n := \frac{1}{2}(p_{n,K} + p_{n,1}).$$

Then

$$p_{n,1}p_{n,K} = \bar{p}_n^2 - \varepsilon_n^2,$$

$$(1-p_{n,1})(1-p_{n,K}) = (1-\bar{p}_n)^2 - \varepsilon_n^2,$$

and

$$p_{n,1}(1-p_{n,K}) + p_{n,K}(1-p_{n,1}) = 2\bar{p}_n(1-\bar{p}_n) + 2\varepsilon_n^2.$$

Proof Each of these statements follows from straightforward algebra:

$$\begin{aligned}\bar{p}_n^2 - \varepsilon_n^2 &= \frac{1}{4}\left(p_{n,1}^2 + 2p_{n,1}p_{n,K} + p_{n,K}^2\right) \\ &\quad - \frac{1}{4}\left((p_{n,K}^2 - 2p_{n,1}p_{n,K} + p_{n,1}^2)\right) \\ &= \frac{1}{4}(4p_{n,1}p_{n,K}) = p_{n,1}p_{n,K}\end{aligned}$$

$$\begin{aligned}(1-\bar{p}_n)^2 - \varepsilon_n^2 &= 1 - 2\bar{p}_n + \bar{p}_n^2 - \varepsilon_n^2 \\ &= 1 - (p_{n,1}+p_{n,K}) + p_{n,1}p_{n,K} \\ &= (1-p_{n,1})(1-p_{n,K})\end{aligned}$$

$$\begin{aligned}2\bar{p}_n(1-\bar{p}_n) + 2\varepsilon_n^2 &= -2(\bar{p}_n^2 - \varepsilon_n^2) + 2\bar{p}_n \\ &= -2(p_{n,1}p_{n,K}) + (p_{n,1}+p_{n,K}) \\ &= p_{n,1}(1-p_{n,K}) + pn,K(1-p_{n,1})\end{aligned}$$

□

It is useful to define an auxiliary sequence of Poisson binomial probability mass functions

$$g_n(m) = f(m; \{p_{n,k}\}_{k=2}^{K-1}),$$

which correspond to the probabilities of m successes *excluding* the most and least likely "events" of the nth distribution. Note that by definition $g_n(m) = 0$ if $m < 0$ or $m > K-2$.

We also define notation for the first and second-order finite difference of $g_n(m)$, which are crucial quantities in our inequalities below.

$$\Delta_{1,n}(m) := g_n(m) - g_n(m-1),$$
and
$$\Delta_{2,n}(m) := \Delta_{1,n}(m) - \Delta_{1,n}(m-1) = g_n(m) - 2g_n(m-1) + g_n(m-2).$$

Using these relationships, we can form a succinct recursive description of $f_n(m)$:

Lemma 10 *For the sequence of probability mass functions above, we have*

$$f_n(m) = f_{n+1}(m) - \varepsilon_n^2 \Delta_{2,n}(m),$$

and for $F_n(m') := \sum_{m=0}^{m'} f_n(m)$,

$$F_n(m') = F_{n+1}(m') - \varepsilon_n^2 \Delta_{1,n}(m').$$

Proof By definition of the probability mass function and $g_n(m)$,

$$\begin{aligned} f_n(m) &= p_{n,1}p_{n,K}g_n(m) \\ &\quad + (p_{n,1}(1-p_{n,K}) + p_{n,K}(1-p_{n,1}))g_n(m-1) \\ &\quad + (1-p_{n,1})(1-p_{n,K})g_n(m-2) \\ &= \bar{p}_n^2 g_n(m) + \bar{p}_n(1-\bar{p}_n)g_n(m-1) + (1-\bar{p}_n)^2 g_n(m-2) \\ &\quad - \varepsilon_n^2(g_n(m) - 2g_n(m-1) + g_n(m-2)) \\ &= f_{n+1}(m) - \varepsilon_n^2 \Delta 2, n(m). \end{aligned}$$

The second equality follows from the identities in Lemma 9, and the second equality follows by definition of $f_{n+1}(m)$ and $\Delta_{2,n}$.

Now taking the summation gives us the second statement:

$$\begin{aligned} F_n(m') &= \sum_{m=0}^{m'} f_{n+1}(m) - \varepsilon_n^2(g_n(m) - 2g_n(m-1) + g_n(m-2)) \\ &= F_{n+1}(m') - \varepsilon_n^2 \left(\sum_{m=0}^{m'} g_n(m) - 2\sum_{m=0}^{m'-1} g_n(m) + \sum_{m=0}^{m'-2} g_n(m) \right) \\ &= F_{n+1}(m') - \varepsilon_n^2(g_n(m') - g_n(m'-1)) \\ &= F_{n+1}(m') - \varepsilon_n^2 \Delta_{1,n}(m'). \end{aligned}$$

□

This lemma gives us an *exact* characterization of the difference between successive distributions in our sequence. Specifically, $f_n(m) \leq f_{n+1}(m)$ if and only if the second order finite difference, $\Delta 2, n(m)$, is non-negative, and $F_n(m') \leq F_{n+1}(m')$ if and only if the first order finite difference, $\Delta_n^1(m')$ is non-negative. In the following lemma, we give a sufficient condition on m to ensure that $\Delta_{1,n}(m) \geq 0$.

Lemma 11 *Let* $\{p_{n,k}\}_{k=1}^K$ *and* μ *be defined as in Lemma 1, and* $\Delta_{1,n}(m)$ *defined as the first order finite difference of* $g_n(m)$*, the Poisson binomial distribution with parameters* $\{p_{n,k}\}_{k=2}^{K-1}$*. Then for* $m \leq (1-p_{1,K})(K-2)\frac{\mu}{1-\mu} + p_{1,K}$, $\Delta_{1,n}(m) \geq 0$.

Proof We start by expressing $g_n(m)$ using the recursive characterization of the Poisson binomial probability mass function given by (Chen et al 1994):

$$g_n(m) = \frac{1}{m}\sum_{i=1}^{m}(-1)^{i-1} g_n(m-i)T_n(i),$$

where $T_n(i) = \sum_{k=2}^{K-1}\left(\frac{p_{n,k}}{1-p_{n,k}}\right)^i$. Note that for $i \geq 2$, we have $T_n(i) \leq T_n(i-1)\frac{p_{1,K}}{1-p_{1,K}}$. The case $\Delta_{1,n}(m+1) \geq 0$ is equivalent to saying that $\frac{g_n(m+1)}{g_n(m)} \geq 1$. Using the recursive expression above,

$$\begin{aligned} \frac{g_n(m+1)}{g_n(m)} &= \frac{T_n(1)}{m+1} - \frac{\sum_{i=1}^m (-1)^{i-1} g_n(m-i)T_n(i+1)}{(m+1)g_n(m)} \\ &\geq \frac{T_n(1)}{m+1} - \frac{\sum_{i=1}^m (-1)^{i-1} g_n(m-i)T_n(i)\left(\frac{p_{1,K}}{1-p_{1,K}}\right)}{(m+1)g_n(m)} \\ &\geq \frac{T_n(1)}{m+1} - \frac{m}{m+1}\left(\frac{p_{1,K}}{1-p_{1,K}}\right) \\ &\geq \frac{K-2}{m+1}\frac{\mu}{1-\mu} - \frac{m}{m+1}\frac{p_{1,K}}{1-p_{1,K}}. \end{aligned}$$

Solving for $m+1$ we have that $\Delta_{1,n}(m+1) \geq 0$ if

$$m+1 \leq (1-p_{1,K})\left((K-2)\frac{\mu}{1-\mu}\right) + p_{1,K}$$

□

Combining Lemmas 10 and 11 completes the proof for Lemma 1.