Team Orienteering Coverage Planning with Uncertain Reward

Bo Liu[1], Xuesu Xiao[1] and Peter Stone[1,2]

Abstract— Many municipalities and large organizations have fleets of vehicles that need to be coordinated for tasks such as garbage collection or infrastructure inspection. Motivated by this need, this paper focuses on the common subproblem in which a team of vehicles needs to plan coordinated routes to patrol an area over iterations while minimizing temporally and spatially dependent costs. In particular, at a specific location (e.g., a vertex on a graph), we assume the cost accumulates over time and its growth rate is a random variable with a fixed but unknown mean, and the cost is reset to zero whenever any vehicle visits the vertex (representing the robot "servicing" the vertex). We formulate this problem in graph terminology and call it Team Orienteering Coverage Planning with Uncertain Reward (TOCPUR). We propose to solve TOCPUR by simultaneously estimating the accumulated cost at every vertex on the graph and solving a novel variant of the Team Orienteering Problem (TOP) iteratively, which we call the Team Orienteering Coverage Problem (TOCP). We provide the first mixed integer programming formulation for the TOCP, as a significant adaptation of the original TOP. We introduce a new benchmark consisting of hundreds of randomly generated graphs for comparing different methods. We show the proposed solution outperforms both the exact TOP solution and a greedy algorithm. In addition, we provide a demo of our method on a team of three physical robots in a real-world environment. The code is publicly available at `https://github.com/Cranial-XIX/TOCPUR.git`.

I. INTRODUCTION

Mobile agent fleets are now being used for many purposes in our daily life, such as a team of mobile robots delivering food [1], a school bus fleet picking up students, or a garbage truck fleet collecting garbage.

In many such situations, visiting a particular location results in some benefit (e.g. collecting piled up garbage), which we model as a reward. The overall objective is therefore to collect as much reward as possible, while ensuring that each vehicle's travel time stays within some budget. This problem can be formulated as the Team Orienteering Problem (TOP) [2]. However, TOP assumes the reward at each location is a known constant before being collected and set to zero. This formulation does not suit problems in which the reward can accumulate over time. For example, consider a garbage truck fleet collecting garbage in a city. The amount of garbage in general grows over time and it becomes much more beneficial to visit a location that has not been visited for a long time. The expected garbage growth rate at different locations might be different and unknown to the agents beforehand. But whenever an agent visits a location and collects the garbage, it can update its

[1]Xuesu Xiao, Bo Liu, and Peter Stone are with Department of Computer Science, University of Texas at Austin, Austin, TX 78712 {bliu, xiao, pstone}@cs.utexas.edu. [2]Peter Stone is also affiliated with Sony AI .
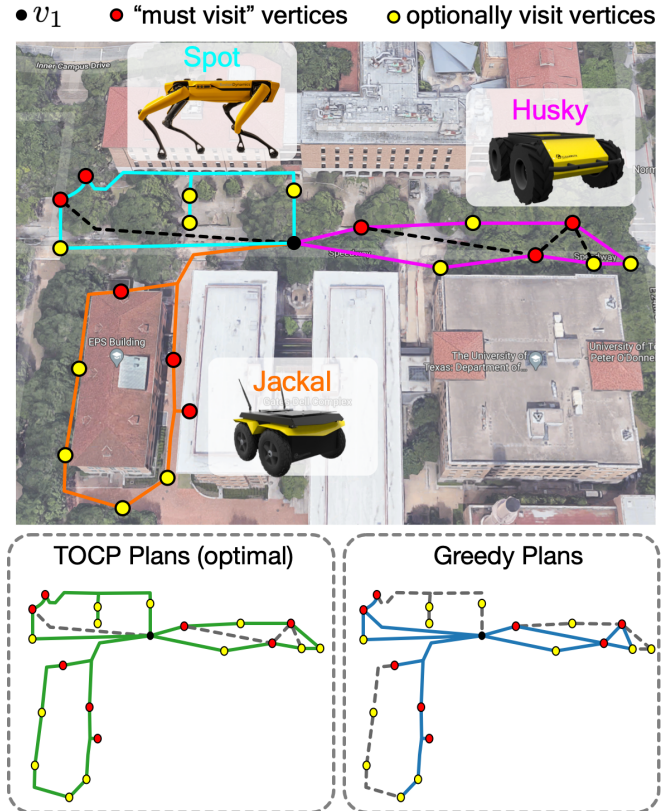
Fig. 1: Real-World Demonstration: From the same black vertex (v_1), a fleet of three robots is tasked with visiting all red vertices and as many yellow vertices as possible. Within the same budget, the optimal plan of the TOCP problem covers all red and yellow vertices (**lower left**), while the greedy plan misses many optional yellow vertices and does a lot of backtracking (**lower right**).

estimation of the growth rate at that location. In addition to TOP assuming a known constant reward, in its typical formulation each location can only be visited once, which again significantly limits its application.

As a result, in this paper, we introduce a novel coverage planning problem, called Team Orienteering Coverage Planning with Uncertain Reward (TOCPUR). In TOCPUR, a team of mobile agents is tasked to patrol a set of predefined locations over multiple iterations. The goal is to reduce the time- and place-dependent costs (i.e., negative rewards). In particular, we assume the cost grows between iterations and stays constant within an iteration, and each vertex can be visited multiple times in an iteration but the cost is only reduced once (e.g., the garbage is collected during the first visit on that day where each day is an iteration). Optionally,
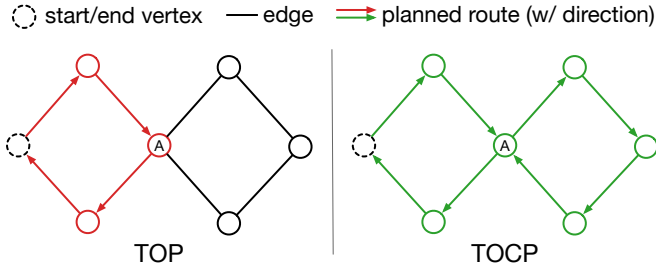
Fig. 2: The TOP **(left)** and the TOCP **(right)** plans on the same graph instance. In the standard TOP formulation, an agent cannot traverse a vertex twice. Hence in TOP, it is impossible to visit the nodes on the right of A in a closed route that starts/ends at the leftmost vertex.

we allow users to specify a subset of locations that the fleet has to visit. This option is useful when the fleet of agents has a primary task (e.g. routine check at certain locations) and some secondary tasks (e.g. collecting as much garbage as possible).

We solve TOCPUR by simultaneously estimating the unknown and growing costs over the area and solving a novel variant of the TOP, called the Team Orienteering Coverage Problem (TOCP), which allows multiple visits to the same nodes. In this paper, we refer to TOCP and TOP plans as the solutions to their corresponding problems. We introduce a benchmark of hundreds of randomly generated graphs and show improved performance using the proposed method compared to both the TOP solution and a greedy algorithm. We also demonstrate the proposed planner working on a team of three physical robots in a real-world environment (Fig. 1).

II. RELATED WORK

In this section, we briefly review prior literature in orienteering problems and coverage planning.

a) Orienteering Problem: The proposed TOCPUR problem is closely related to the team orienteering problem (TOP) [2], which is a multi-tour extension of the orienteering problem (OP) [3]. The OP originates from the Travelling Salesman Problem and is heavily studied in the optimization community. In the standard OP, an agent needs to plan a path under a fixed length constraint that collects as much reward as possible as it traverses the vertices along that path. The TOP is then a multi-tour extension of the OP where a team of agents plan consecutively, which is equivalent to a single agent planning multiple times. The major difference between TOCPUR and TOP is that in TOCPUR, we allow each vertex to be visited multiple times, which better resembles many real-world scenarios but makes the problem harder. As a result, the planned route becomes a walk instead of a path (e.g. a walk without loops) for each agent (See Fig. 2). While this modification seems to be a small change, the core constraints in the formulation of the standard TOP problem can no longer be used. This modification is also the key contribution of our formulation of the TOCPUR problem. In addition, TOCPUR models cumulative reward over time where at each time step, the reward is sampled identically

and independently from a fixed unknown distribution. This setup is similar to the OP with stochastic profits (OPSP) [4]. However, in OPSP, the objective is to maximize the probability that the total collected profit will be greater than a predefined target value. In addition, OPSP does not allow for reward/profit to grow over time.

Besides TOP and OPSP, there exists a big family of variants of the OP. For instance, prior research has considered the OP with time window (OPTW) [5] where the agent can only visit a vertex within a pre-specified time window. Generalized OP (GOP) extends OP such that the reward at each vertex is a non-linear function with respect to a set of attributes. OP with variable profit (OPVP) considers multiple passes for each node where each pass collects a fixed percentage of the remaining resource [6]. Multi-agent OP (MAOP) [7] considers a competitive game among multiple agents trying to solve the OP individually. More recently, multi-visit TOP (MVTOP) [8] has been proposed where each vertex needs to be served by multiple agents with different skills in a certain order. In general, different combinations of having multiple tours, multiple agents, different time windows, and stochastic rewards have been studied. However, in the formulation of all these variants of the OP, each vertex is only allowed to be visited once by a single vehicle in a single tour. TOCPUR, by allowing multiple visit to a single vertex for a single agent, has much broader applications. Gunawan et al. [9] provide a comprehensive overview of existing variants of the OP.

b) Coverage Planning: Coverage planning (CP) is the task in robotics of determining a path to cover all points in an area while avoiding obstacles. Standard CP problems involve both high-level path planning and low-level motion planning. Typical methods will break the free space, i.e., the space free of obstacles, down into simple, non-overlapping regions called cells [10]. Then an exhaustive walk through the graph defined by the decomposed cells is found. TOPCUR is similar to CP problems in that it aims to fully cover a set of must-visit vertices and maximizes the coverage over other vertices through planned routes. In our current formulation, TOPCUR primarily focuses on the high-level path planning.

III. BACKGROUND

In this section, we introduce the notations and formally define the TOCPUR problem using graph terminology.

A. Notation

We represent the area to be patrolled as a symmetric directed graph $G = (V, E)$, where $V = \{v_1, v_2, \ldots, v_N\}$ is the set of N vertices and E is the edge set. e_{ij} denotes an edge from v_i to v_j, which has a length of l_{ij}.[1] In addition, we assume the agent fleet consists of M agents in total, which we call agent 1 to M when the context is clear. We use $[x]$ to denote $\{1, 2, \ldots, x\}$, and $[a, b]$ to denote $\{a, a+1, \ldots, b\}$.

---

[1]Since G is symmetric directed, if e_{ij} exists, e_{ji} also exists and $l_{ij} = l_{ji}$. The symmetric directed assumption represents bidirectional traffic.
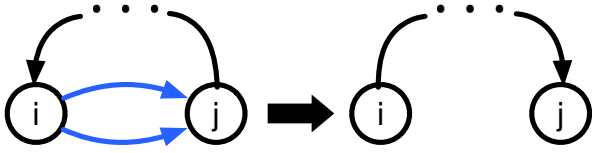
Fig. 3: If e_{ij} is visited twice (blue), we can remove them by reversing the direction of another route connecting v_j to v_i.

B. Problem Formulation

In this section, we define the TOCPUR problem. We consider H iterations of route planning for a vehicle fleet of size M over a symmetric directed graph $G = (V, E)$. Each vertex v_i accumulates a cost of $\kappa_{i,t}$ right before the t-th iteration, where $\mathbb{E}[\kappa_{i,t}] = \mu_i^*$ and μ_i^* is unknown a priori. In other words, the cost at each vertex approximately grows linearly in expectation at an unknown rate (In the garbage truck application, think of an iteration as a day of garbage collection and $\kappa_{i,t}$ as the amount of garbage that appears overnight at location i prior to day t). Next, denote $T_{i,t}$ as the most recent iteration prior to t when the vehicle fleet visited v_i. Then v_i at iteration t accumulates a total cost of $c_{i,t} = \sum_{k=T_{i,t}+1}^{t} \kappa_{i,k}$. The entire cost over G at iteration t is specified as $c(G,t) = \sum_{i \in [N]} c_{i,t}$. Intuitively, the total cost at the end of an iteration is the sum of the current accumulated costs of all the vertices *not* visited during that iteration (the current cost of those visited is reset to 0). The goal is then to plan M routes $\{\tau_{m,t}\}_{m=1}^{M}$, each no longer than a maximum length $l_{\max}$, for all vehicles at each iteration $t \in [H]$, such that the expected total cost on G over the horizon is minimized. Specifically, each route $\tau_{m,t} = (v_1, \ldots, v_1)$ is a sequence of vertices that starts and ends in v_1. Optionally, we can include a set of "must visit" vertices I such that the fleet has to visit all vertices in I during each iteration. The whole problem can be described as the following optimization:

$$\underset{\{\{\tau_{m,t}\}_{m=1}^{M}\}_{t=1}^{H}}{\text{minimize}} \quad \mathcal{L} = \mathbb{E}\Big[\sum_{t=1}^{H} c(G,t)\Big], \quad \text{where}$$

$$c(G,t) = \sum_{i=1}^{N} \sum_{k=T_{i,t}+1}^{t} \kappa_{i,k}.$$

$$\text{subject to} \quad \forall m,t \quad \sum_{e_{ij} \in \tau_{m,t}} l_{ij} \le l_{\max},$$

$$\forall i,t, \quad T_{i,t+1} = \begin{cases} t & \text{if } v_i \in \bigcup_m \tau_{m,t}, \\ T_{i,t} & \text{otherwise} \end{cases}$$

$$\forall i, \quad T_{i,1} = 0,$$

$$\text{(optionally)} \quad \forall v_i \in I, \quad v_i \in \bigcup_m \tau_{m,t}.$$

$$(1)$$

Here, $\bigcup_m \tau_{m,t} = \bigcup_m \{v \mid v \in \tau_{m,t}\}$ and we abuse the notation a bit to denote $e_{ij} \in \tau$ if v_i, v_j appear consecutively in τ. This optimization problem (Opt. 1) is NP-hard even when we know $\{\mu_i^*\}$ exactly since it can easily reduce to the Travelling Salesman Problem. We emphasize two points: 1) As $\kappa_{i,t}$ is drawn randomly from a distribution with an unknown mean μ_i^*, it is in general impossible to have an optimal open-loop plans for solving Opt. 1. 2) Unlike in TOP, we *do not* assume that each v_i can only appear once in any $\tau_{m,t}$. However, as we will see in the following, we can safely assume each e_{ij} appears at most once in any $\tau_{m,t}$.

IV. METHOD

In this section, we propose to solve the TOCPUR problem by optimizing the per-iteration objective while updating the reward estimation simultaneously. The per-iteration objective is then specified by a novel mixed integer programming formulation. In addition, we provide a simple greedy method as a baseline for solving the problem approximately.

A. Reward Estimation and Per-Iteration Planning

Solving Opt. 1 exactly for large H is computationally intractable. In fact, even when $H = 1$ and $\{\mu_i^*\}_{i=1}^{N}$ are known, the problem remains NP-hard. In addition, because $\{\mu_i^*\}_{i=1}^{N}$ are hidden, an optimal solution must consists of closed-loop plans that take past observations (e.g. κ) into consideration. Therefore, we propose to optimize the expected cost $\mathbb{E}[c(G,t)]$ iteratively while updating the estimates of $\{\mu_i^*\}_{i=1}^{M}$ simultaneously. Specifically, we keep track of $T_{i,t}$ for each node v_i, as well as the total observed cumulative cost at v_i, i.e. $C_{i,t} = \sum_{k=1}^{T_{i,t}} \kappa_{i,k}$. The maximum likelihood estimation of μ_i^* at time t is therefore

$$\hat{\mu}_{i,t} = \begin{cases} \frac{C_{i,t}}{T_{i,t}} = \frac{\sum_{k=1}^{T_{i,t}} \kappa_{i,k}}{T_{i,t}} & T_{i,t} > 0 \\ \mu_{\text{default}} & T_{i,t} = 0, \end{cases} \quad (2)$$

where μ_{default} is a default value or a pre-specified value based on prior knowledge when there has been no visits to a node. Therefore, at each iteration t, we can use $\{\hat{\mu}_{i,t}\}_{i=1}^{N}$ to estimate $\mathbb{E}[c(G,t)]$, i.e. $\hat{c}(G,t) = \sum_{i=1}^{N} \hat{\mu}_{i,t}(t - T_{i,t}) \approx \sum_{i=1}^{N} \mu_i^*(t - T_{i,t}) = \mathbb{E}[c(G,t)]$. Following the convention in orienteering problems [2], we formulate the per-iteration optimization as a mixed integer program (MIP). In the following, we will temporarily ignore the subscript t. Let $\{x_{ijm}\}$, $\{y_{im}\}$ and $\{z_i\}$ be the binary decision variables and $i, j \in [N]$ and $m \in [M]$. $x_{ijm} = 1$ if agent m traverses e_{ij} and 0 otherwise. Similarly, $y_{im} = 1$ if agent m visits v_i and $z_i = 1$ if any agent visits v_i. It is sufficient to assume x_{ijm} is binary, as justified by the following proposition.

Proposition 1: Any optimal solution of Opt. 1 has an equivalent solution where each e_{ij} is traversed at most once for a single agent at a single iteration.

Proof: Assume otherwise. Let e_{ij} be the edge that is visited at least twice. If v_i is only visited once, then certainly we can remove one traversal of e_{ij} with no problems. If v_i is visited more than once, then there has to exist a route p, i.e. a sequence of vertices, that goes from v_j back to v_i. But then, since G is symmetric directed, we can remove both traversals of e_{ij} by reversing the traversal direction of the edges along the route p. This change results in visiting the same set of vertices with a shorter length (See Fig. 3). ∎

The per-iteration optimization is then formulated as the following MIP, which we call the team orienteering coverage

problem (TOCP).[2]

$$\text{maximize} \quad \sum_{m=1}^{M} \sum_{i=2}^{N} \hat{c}_{i,t} z_i \qquad (3)$$

$$\forall m, \qquad \sum_{j=2}^{N} x_{1jm} = \sum_{i=2}^{N} x_{i1m} = 1 \qquad (4)$$

$$\forall m, i, \qquad x_{iim} = 0 \qquad (5)$$

$$\forall m, i, \qquad y_{im} \leq \sum_{j=1}^{N} x_{ijm} \leq \frac{y_{im} \cdot l_{\max}}{\min_{e_{ij} \in E} l_{ij}} \qquad (6)$$

$$\forall i, \qquad z_i \leq \sum_{m} y_{im} \leq z_i \cdot M \qquad (7)$$

$$\forall m, i, \qquad \sum_{j=1}^{N} x_{ijm} = \sum_{j=1}^{N} x_{jim} \qquad (8)$$

$$\forall m, \qquad \sum_{i=1}^{N} \sum_{j=1}^{N} l_{ij} x_{ijm} \leq l_{\max} \qquad (9)$$

$$\forall m, \qquad \sum_{j=1}^{N} u_{1jm} - \sum_{i=1}^{N} u_{i1m} = \sum_{j=2}^{N} y_{jm} \qquad (10)$$

$$\forall m, i > 1, \qquad \sum_{j=1}^{N} u_{ijm} - \sum_{j=1}^{N} u_{jim} = y_{im} \qquad (11)$$

$$\forall m, i, j, \qquad 0 \leq u_{ijm} \leq N \cdot x_{ijm} \qquad (12)$$

$$\text{(optionally)} \quad \forall v_i \in I, \quad z_i = 1 \qquad (13)$$

Eq. 3 is the objective. Eq. 4 ensures all agents start and end in v_1. Eq. 5 eliminates self-loops. Eq. 6-7 enforce the definitions of y_{im} and z_i. For instance, $y_{im} = 0$ if and only if $\forall j, \ x_{ijm} = 0$.[3] Eq. 8 ensures the conservation of flow. Eq. 9 ensures each agent travels within the length budget $l_{\max}$. Eq. 10-12 ensures the found route τ_m for each agent m is strongly connected. The variables $\{u_{ijm}\}$ define an amount of flow that begins at v_1 and is reduced by 1 at every node m visits in sequence. For instance, the net outflow at v_1 for m should be $\sum_{j \in [2,N]} y_{jm}$ (Eq. 10), the number of vertices m visits in τ_m. If m's trajectory were to include two disconnected components, then there would be no way to consume all of the flow (e.g., Eq. 10-12 would be violated). Note that u_{ijm} should only be positive if m traverses e_{ij}, which is ensured by Eq. 12. Finally, Eq. 13 optionally ensures that all "must visit" vertices in I are visited. In the above MIP, Eq. 6-7 and Eq. 10-12 are *novel* constraints designed specifically for TOCP. The whole algorithm for solving TOCPUR is summarized in Alg. 1.

Remark: The sub-optimality of Alg. 1 originates from two sources: 1) the decomposition sub-optimality caused by decomposing the long horizon planning into per-iteration planning; and 2) the uncertainty sub-optimality caused by inaccurate estimation of $\{\mu_i^*\}$, which decreases as we visit each vertex more often. It is not immediately clear

---

Algorithm 1 Reward Estimation and Per-Iteration Planning

1: **Maintain:** for each v_i, we maintain $T_{i,t}$, the most recent iteration when v_i was visited ($T_{i,1} = 0$), and $C_{i,t}$, the observed cumulative cost at v_i up to time $T_{i,t}$.
2: **Input:** the graph $G = (V, E)$, the "must visit" vertices I, the maximum traversal budget $l_{\max}$.
3: **for** $t = 1$ to H **do**
4: $\quad \forall i$, update $\hat{\mu}_{i,t}$ according to Eq. 2.
5: $\quad \forall i$, $\hat{c}_{i,t} = \hat{\mu}_{i,t} \cdot (t - T_{i,t})$.
6: $\quad$ Plan $\{\tau_{m,t}\}_{m=1}^{M}$ by solving Opt. 3-13.
7: $\quad \forall i \in [N], \quad T_{i,t+1} = \begin{cases} t & \text{if } v_i \in \bigcup_m \tau_{m,t}, \\ T_{i,t} & \text{otherwise.} \end{cases}$
8: $\quad C_{i,t} = \sum_{k=1}^{T_{i,t}} \kappa_{i,k}$.
9: **end for**

Algorithm 2 Greedy Per-Iteration Planning

1: **Input:** the graph $G = (V, E)$, the estimated cost growth for all vertices $\{\hat{\mu}_{i,t}\}_{i=1}^{N}$, the time since last visit to each node $\{T_{i,t}\}_{i=1}^{N}$, the "must visit" vertices I, the maximum traversal budget $l_{\max}$, $D : V \times V \to \mathbb{R}$, a function that outputs the shortest distance between any pair of vertices (e.g., calculated by Floyd-Warshall).
2: $\forall i \in [N]$, compute $\hat{c}_{i,t} = \hat{\mu}_{i,t} \cdot (t - T_{i,t})$.
3: $A \leftarrow I$ and $B \leftarrow V \setminus I$. (must/optionally visit vertices)
4: $\forall m \in [M]$, $\tau_{m,t} = [\ ]$, $f_m = 0$ (whether m finishes), $v_m = v_1$ (current location of m), $l_m = 0$ (travelled distance of m).
5: **while** $\sum_m f_m < M$ **do**
6: $\quad$ **for** agent $m \in \{x \mid f_x = 0, x \in [M]\}$ **do**
7: $\quad\quad Z \leftarrow \{v \mid l_m + D(v_m, v) + D(v, v_1) \leq l_{\max}\}$.
8: $\quad\quad$ **if** $A \cap Z \neq \emptyset$ **then**
9: $\quad\quad\quad v \leftarrow \arg\min_{x \in A \cap Z} D(v_m, v)$; $X \leftarrow A$.
10: $\quad\quad$ **else if** $B \cap Z \neq \emptyset$ **then**
11: $\quad\quad\quad v \leftarrow \arg\max_{v \in B \cap Z} \hat{c}_{i,t}/D(v_m, v)$; $X \leftarrow B$.
12: $\quad\quad$ **else**
13: $\quad\quad\quad v = v_1$; $f_m = 1$; $X \leftarrow \emptyset$.
14: $\quad\quad$ **end if**
15: $\quad\quad l_m \leftarrow l_m + D(v_m, v)$.
16: $\quad\quad$ Let p be the shortest path from v_m to v.
17: $\quad\quad X \leftarrow X \setminus p$; $\tau_{m,t}$ append p; $v_m \leftarrow v$.
18: $\quad$ **end for**
19: **end while**

---

whether the decomposition sub-optimality can ever arise. To illustrate that it can arise, we provide an example such that even when the true $\{\mu_i^*\}$ are provided, the per-iteration optimal plans are still not optimal over the horizon. The example is illustrated in Fig. 4 with $H = 2$. In Fig. 4, the leftmost graph shows the initial costs on all vertices. For illustration simplicity, we assume there is no growing cost at this moment (so each vertex only has accumulate an initial cost). Due to the travel length budget $l_{\max} = 5$, the per-iteration optimal plans achieve a total cost of 12, while the optimal plans achieve a cost of 9 over two steps.
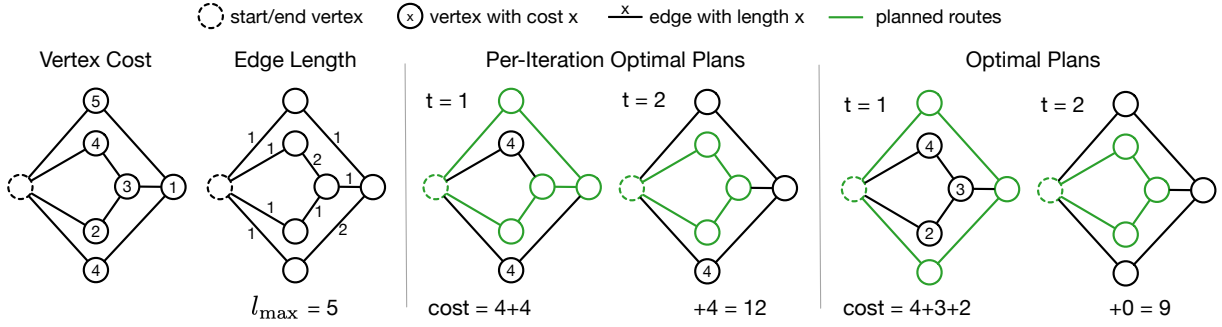

Fig. 4: An example problem with $H = 2$ and $M = 1$, where the per-iteration plans are not optimal. **left:** the graph G. The initial vertex cost is specified on the vertex and the edge length is labeled on each edge. Note that all edges are bi-directional and we assume $l_{max} = 5$. **middle:** the optimal per-iteration plans visits 4 vertices at $t = 1$, and only one of the two un-visited vertices at $t = 2$. **right:** the optimal plans visit 3 nodes at each iteration, covering all nodes in the end.

B. Greedy Per Step Planning

In addition to the exact MIP solution from Opt. 3-13, we provide a greedy algorithm that efficiently and approximately solves TOCP and therefore TOCPUR, summarized in Alg. 2. The principle idea is to prioritize must visit vertices first. When all "must visit" vertices have been visited, we prioritize vertices with a larger ratio between $\hat{c}_{i,t}$ and the current distance to v_i.

V. Experiments

In this section, we conduct simulated experiments to evaluate two hypotheses: 1) the proposed iterative method outperforms an exact TOP method, and 2) solving the MIP in Opt. 3-13 exactly outperforms the simple greedy algorithm.

A. Simulated Experiments

To compare the proposed method with the exact TOP method and the greedy method, we introduce a new benchmark that consists of 600 randomly generated graphs. For each graph, the number of vertices N is drawn uniformly at random from $\{10, 12, 14, 16, 18, 20\}$. The vertex v_1 is positioned at $(0, 0)$ in the standard euclidean plain. For vertices $v_2, \ldots, v_N$, each vertex's position is drawn uniformly at random from a 10×10 square centered at the origin, with its sides parallel to the x- and y-axes. Then each vertex v_i is connected to its closest n_i vertices, where n_i is drawn uniformly at random from $\{3, 4, 5\}$. l_{max} is drawn from $U(20, 20+2N)$, where $U(a, b)$ denotes a uniform distribution from a to b. The number of agents M is drawn uniformly at random from $\{2, 3, 4, 5\}$. The number of "must visit" vertices, N_I, is set to $\min(X, M)$, where X is drawn uniformly at random from $\{1, 2, 3\}$. Denote $V_{reachable}$ as all vertices reachable from v_1 within the travel budget l_{max}; we sample N_I vertices from $V_{reachable}$ without replacement to form I. Finally, for each v_i, the expected growth of cost μ_i^* is drawn from $U(0.1, 0.9)$. For each iteration t, the actual growth $\kappa_{i,t} = \min(\max(x_{i,t}, 0), 1)$, where $x_{i,t} \sim \mathcal{N}(\mu_i^*, 0.1)$. Finally, we generate 120 random graphs with random seeds ranging from $[1, 120]$, following the above procedure for each horizon H in $\{2, 4, 6, 8, 10\}$, which results in a total of 600 random graphs. We summarize the experiment results
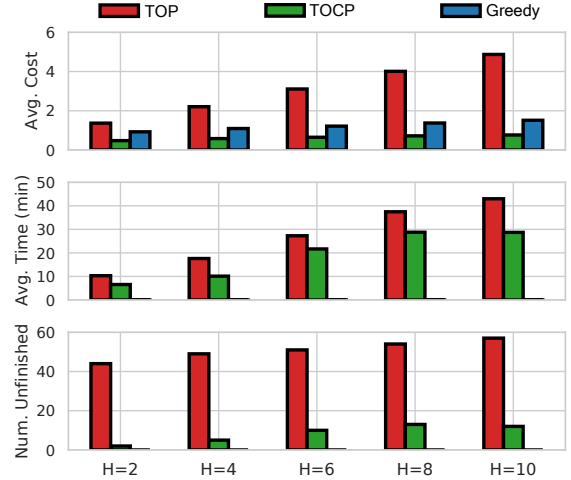

Fig. 5: Comparison among TOCP, TOP, and the greedy algorithm on 600 randomly generated graphs.

in Fig. 5. In all cases, we follow Alg. 1, and varying the per-iteration method (line 6 of Alg. 1). We compare TOCP solutions with exact TOP solutions and the solutions found by Alg. 2. We limit the per-iteration computation time of TOP and TOCP to 1000 seconds. Therefore, occasionally TOP and TOCP might not find a feasible solution. The first row of Fig. 5 shows the total cumulative cost for each horizon H, averaged over the subset of 120 random graphs where all methods find a solution. The total cumulative cost is essentially the objective in Eq. 1. TOCP solutions outperform the ones found by the greedy method, and both outperform TOP solutions by a large margin. We provide additional pairwise independent-samples T-tests in Table I and highlight the conclusions that are statistically significant ($p \leq 0.05$).

| | TOP-TOCP | | Greedy-TOCP | |
Horizon	t-score	p-value	t-score	p-value
$H = 2$	3.63	**.0004**	2.38	**.018**
$H = 4$	4.00	**.0001**	2.04	**.043**
$H = 6$	4.36	**.00004**	1.97	**.050**
$H = 8$	4.46	**.00003**	1.95	.054
$H = 10$	4.48	**.00003**	1.96	.052

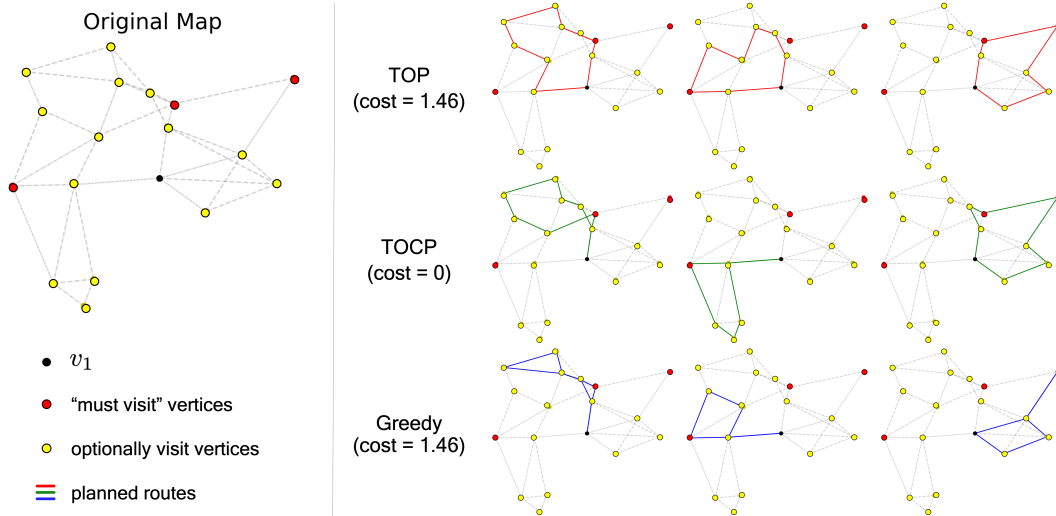TABLE I: Independent-samples T-tests for TOP v.s. TOCP and the greedy method v.s. TOCP.

Fig. 6: Solutions from TOP, TOCP and the greedy method on an example graph with $H = 1$ and $M = 3$.

The second row in Fig. 5 reports the average computation time, again over the subset of graphs where all methods find solutions. As expected, TOP and TOCP take much longer than the greedy method does. The last row of Fig. 5 reports the number of graphs for which each method fails to find a solution. Since TOP does not allow a second visit to any vertex, TOP fails more often than TOCP.

In addition to the quantitative evaluations in Fig. 5. We also provide a qualitative visualization with $H = 1$ and $M = 3$ in Fig. 6 to further showcase the difference in the three methods used. Among all three methods, TOCP is the only one that cover all vertices, thus clearing all the costs.

B. Physical Demo

We additionally record a demo of applying Alg. 1 on three physical robots. For simplicity, we assume $H = 1$ and the ground truth reward (or cost) of each node is given a priori.[4]

VI. CONCLUSION

In this work, we formulate a novel variant of the team orienteering problem (TOP) that allows multiple visits to the same vertex and uncertain cumulative costs on each vertex over a horizon. We propose a method to iteratively find the per-iteration optimal plans using a novel mixed integer programming formulation based on the maximum likelihood estimates of each vertex's costs. The simulated experiments show that the proposed method greatly outperforms the exact TOP solution. We also provide a real-world demo of the proposed method on three physical robots. In this paper, we focus on high-level route planning. An interesitng direction for future work is to incorporate obstacle avoidance into TOCPUR.

ACKNOWLEDGMENT

REFERENCES

[1] Y. Sun, L. Guan, Z. Chang, C. Li, and Y. Gao, "Design of a low-cost indoor navigation system for food delivery robot based on multi-sensor information fusion," *Sensors*, vol. 19, no. 22, p. 4980, 2019.

[2] I.-M. Chao, B. L. Golden, and E. A. Wasil, "The team orienteering problem," *European journal of operational research*, vol. 88, no. 3, pp. 464–474, 1996.

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

[4] T. Ilhan, S. M. Iravani, and M. S. Daskin, "The orienteering problem with stochastic profits," *Iie Transactions*, vol. 40, no. 4, pp. 406–421, 2008.

[5] N. Labadie, R. Mansini, J. Melechovský, and R. W. Calvo, "The team orienteering problem with time windows: An lp-based granular variable neighborhood search," *European Journal of Operational Research*, vol. 220, no. 1, pp. 15–27, 2012.

[6] G. Erdogan and G. Laporte, "The orienteering problem with variable profits," *Networks*, vol. 61, no. 2, pp. 104–116, 2013.

[7] C. Chen, S.-F. Cheng, and H. C. Lau, "Multi-agent orienteering problem with time-dependent capacity constraints," *Web Intelligence and Agent Systems: An International Journal*, vol. 12, no. 4, pp. 347–358, 2014.

[8] S. Hanafi, R. Mansini, and R. Zanotti, "The multi-visit team orienteering problem with precedence constraints," *European journal of operational research*, vol. 282, no. 2, pp. 515–529, 2020.

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

[10] T. Oksanen and A. Visala, "Coverage path planning algorithms for agricultural field machines," *Journal of field robotics*, vol. 26, no. 8, pp. 651–668, 2009.

---

[4]The video link is at `https://drive.google.com/file/d/1pwE-zLbpcYK2DGeWZ2L5ePZsuO78KCpc/view?usp=sharing`.