2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
Congress Center Hamburg
Sept 28 - Oct 2, 2015. Hamburg, Germany

Anytime Planning of Optimal Schedules for a Mobile Sensing Robot

Jingjin Yu Javed Aslam Sertac Karaman Daniela Rus

***Abstract*—We study the problem in which a mobile sensing robot is tasked to travel among and gather intelligence at a set of spatially distributed points-of-interest (POIs). The quality of the information collected at a POI is characterized by some sensory (reward) function of time. With limited fuel, the robot must balance between spending time traveling to more POIs and performing time-consuming sensing activities at POIs to maximize the overall reward. In a dual formulation, the robot is required to acquire a minimum amount of reward with the least amount of time. We propose an anytime planning algorithm for solving these two NP-hard problems to arbitrary precision for arbitrary reward functions. The algorithm is effective on large instances with tens to hundreds of POIs, as demonstrated with an extensive set of computational experiments. Besides mobile sensor scheduling, our algorithm also applies to automation scenarios such as intelligent and optimal itinerary planning.**

I. Introduction

Consider the scenario in which an environmental scientist wants to plan an automated, GPS-guided trip for an aerial robot to collect valuable data (*e.g.*, audio, video, temperature, and so on) at a set of spatially distributed, remote geolocations (equivalently, *points-of-interest* or POIs). To execute its task, the robot must travel to these POIs and then perform data collection at each visited POI, both of which are time consuming. Frequently, the quality of the information gathered at a POI directly depends (in a diminishing manner) on the amount of time spent at the POI. When the robot is only available to the scientist for a limited time, which is the case for many shared resources, our environmental scientist is faced with the challenge of designing an optimal plan for the robot to get the most valuable combination of data according to some metric. Is there a principled approach that she can use for planning such a trip with optimality guarantees?

In this paper, we propose the Optimal Tourist Problem[1] (OTP) and an associated anytime algorithm to address the above mentioned autonomous planning and scheduling scenario. In the basic setup, a robot (equivalently, a tourist) is tasked to visit up to n POIs that are spatially distributed. Each POI is associated with a *sensory reward function* (or simply *reward function*), a non-negative bounded function of the time spent at the POI. Because traveling between POIs and doing sensing at a POI to gain information are both time consuming, optimization problems naturally arise. We introduce two related formulations that are duals of each other. In the first problem, a *reward-maximizing tourist* (RMT) seeks to maximize the total collected information given limited time budget. From a dual perspective, in the second problem, a *budget-minimizing tourist* (BMT) seeks to minimize the time spent to collect a predetermined amount of reward. We provide a mixed integer programming (MIP) based *anytime* algorithm for solving both RMT and BMT.

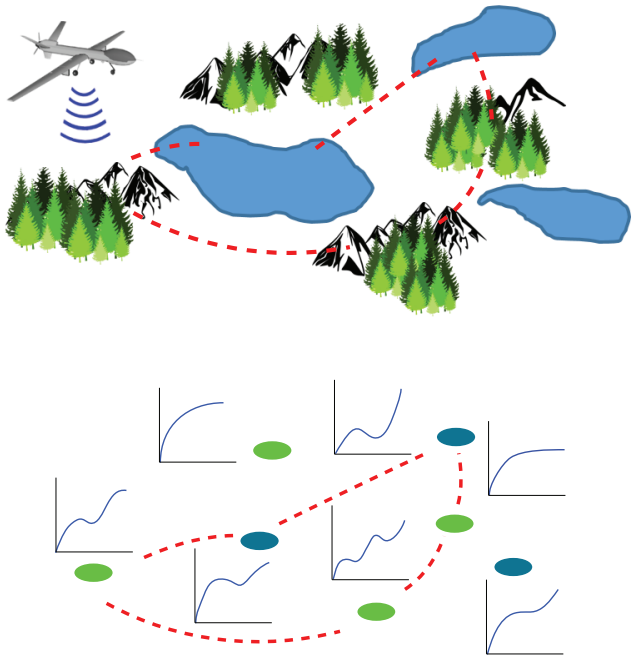

Fig. 1. [top] A scientific monitoring scenario in which a UAV collects data (*e.g.*, image, video, air quality, temperature, and so on) from multiple geolocations (POIs). In this case, there are four mountainous regions and three lakes. [bottom] Each POI is associated with a known sensory function that characterizes the information gain over the time spent at a POI. With limited fuel, the UAV must maximize the total information gain by carefully selecting the subset of POIs to visit and deciding how long to carry out its sensing activity at each visited POI. As an illustration, the best schedule may only visit two mountainous regions and two lakes.

Motivation and related work. The primary motivation behind our study of OTP is its application to robotic surveillance and monitoring problems such as automated reconnaissance and scientific survey, which we refer to under the umbrella term of *informative path and policy planning*. In such a

J. Yu and D. Rus are with the Computer Science and Artificial Intelligence Lab at the Massachusetts Institute of Technology. E-mail: {jingjin, rus}@csail.mit.edu. J. Aslam is with the College of Computer and Information Science at Northeastern University. Email: jaa@ccs.neu.edu. S. Karaman is with the Department of Aerospace and Astronautics Engineering at the Massachusetts Institute of Technology. E-mail: karaman@mit.edu.

This work was supported in part by ONR projects N00014-12-1-1000 and N00014-09-1-1051, and the Singapore-MIT Alliance on Research and Technology (SMART) Future of Urban Mobility project.

[1]The name of the problem is inspired by the Traveling Salesperson Problem (TSP), a planning problem contained in our problem formulation.

problem, a path or a policy is planned to satisfy some information collection objective, sometimes under additional constraints such as path length or total time limit. In [1], an $O(\log n)$ approximation algorithm yields iterative TSP paths that minimize the maximum latency (the inverse of the frequency with which a node is visited) across all n nodes in a connected network. In [2], the authors proposed a method for generating speed profiles along predetermined cyclic paths to keep bounded the uncertainty of a varying field using single or multiple robots. For the problem of observing stochastically arriving events at multiple locations with a single mobile robot, a $(1+\varepsilon)$-optimal algorithm was proposed in [3] to solve the multi-objective optimization problem of maximizing event observation in a balanced manner and minimizing delay between event observations across the locations. Recently, a method called *Recursive Adaptive Identification* is proposed as a polynomial time polylogarithmic-approximation algorithm for attacking adaptive informative path planning (IPP) [4]. Under suitable conditions, submodularity can be explored to provide approximation guarantees for IPP variants [5]. Sampling based methods [6]–[8] have also been applied to IPP problems with success. For example, in [9], Rapidly-Exploring Random Graphs (RRG) are combined with branch-and-bound methods for planning most informative paths.

A problem intimately connected to OTP is the Orienteering Problem (OP) [10]–[12], which is obtained when rewards at the POIs are fixed in RMT. The fixed reward is collected in full once a POI is visited. OP, which is easy to see as an NP-hard problem, is observed to be difficult to solve exactly for instances with over a hundred POIs. On the side of approximation algorithms, constant approximation ratios down to $(2+\varepsilon)$ are only known under metric settings for OP with uniform reward across the POIs [13]. On the other hand, many MIP-based algorithms exist for OP and related problems [11], [12]. These algorithms often allow the precise encoding of the problem in the MIP model. A work in this domain that is closest to ours studies an OP problem in which the reward may depend on the time spent at the POIs [14]. It proposes a solution method that iteratively adds constraints that are violated by the incomplete model. The method applies to only limited reward function classes and does not always bound the achievable approximation ratio as we do. On the side of trip planning problems, many interesting work [15]–[17] compute "optimal" itineraries according to some reward metric. For example, the authors of [15] apply a recursive greedy approximation algorithm for OP [18] to plan suggested itineraries. However, the provided guarantee is over 2-optimal, which is unsatisfactory in practice. In the end, most of these work focus on the data mining aspect of trip planning problems, *e.g.*, how POI related data, such as the average visiting times for POIs and tourist preference through POI correlations, may be derived and used.

Contributions. Our study brings two main contributions. First, we introduce two novel, complementary planning problems that allow the traveling and sensing costs of a mobile robot to be jointly considered through a single and natural optimality criterion. Both formulations are applicable to practical mobile sensor scheduling scenarios. In addition, the formulations are highly flexible, supporting arbitrary sensory reward functions. Second, we derive a mixed integer programming (MIP) model capable of approximating arbitrary reward functions to arbitrary precision. Furthermore, the MIP model directly leads to an anytime algorithm that can quickly compute optimal or near optimal solutions to instances with tens to hundreds of POIs.

Organization. The rest of the paper is organized as follows. In Section II, we formulate the two variants of the optimal tourist problem. In Section III, we provide a step-by-step introduction of our MIP model for solving the proposed problem variants. In Section IV, we discuss the overall algorithm and some of its important properties in more detail. We present computational simulations as well as data-driven experiments in Section V, and conclude in Section VI. Due to limited space, we omit proofs and some experimental results.

II. Problem Formulation

Let the set $V = \{v_1, \ldots, v_n\}$ represents n *points-of-interest* (POIs) in $\mathbb{R}^2$. There is a *directed edge* $e_{i,j}$ between two POI vertices $v_i, v_j \in V$ if there is a path from v_i to v_j that does not pass through any intermediate POIs. When an edge $e_{i,j}$ exists, let $d_{i,j}$ denote its length. There is a mobile robot (equivalently, a tourist) that travels between the POIs following single integrator dynamics. Denoting the robot's location as $\mathbf{p}$, when the robot is traveling from one POI to another, $\dot{\mathbf{p}} = u$ with $\| u \| = 1$. Otherwise, $\dot{\mathbf{p}} = 0$.

The robot is tasked to visit the POIs and perform information collection. To do so, it starts from some *base* vertex $v_B \in B \subset V$ with $|B| = n_B \leq n$, travels between the POIs, and eventually returns to v_B. Each $v_i \in V$ is associated with a maximum sensing *reward* r_i. We assume that the obtained reward depends on the time t_i the robot spends at v_i. More precisely, the obtained reward is defined as $r_i f_i(t_i)$, in which $f_i \in [0,1]$ is some function of t_i that is non-decreasing. We further require that f_i is C^1 continuous and $f_i'(0)$ is bounded away from zero. That is, for all $1 \leq i \leq n$, f_i' is continuous and $f_i'(0) \geq \lambda$ for some fixed $\lambda > 0$. We also assume that $f(0) = 0$ for convenience (note that the assumption does not reduce generality).

Whereas our result supports arbitrary reward functions, in this paper, two specific types of one-parameter family of reward functions are studied further: *linear* and *exponential*. Let $\lambda_i > 0$ denote the (sensory) *learning rate*. In the case of a linear reward,

$$f_i(t_i) = \lambda_i t_i, \quad 0 \leq t_i \leq \frac{1}{\lambda_i}. \tag{1}$$

The exponential reward function is specified as

$$f_i(t_i) = 1 - e^{-\lambda_i t_i}, \quad 0 \leq t_i \leq +\infty, \tag{2}$$

which captures the notion of "diminishing return" that are often present in such sensing tasks.

After a trip is completed, the robot would have traveled through a subset of the edges $E_{tr} \subset E$ and have spent time

$t_1,\ldots,t_n,t_i \geq 0$ at the n POIs. It would have spent a total time of

$$J_T := \sum_{e_{i,j}\in E_{tr}} d_{i,j} + \sum_{i=1}^{n} t_i \tag{3}$$

and gained a total reward of

$$J_R := \sum_{i=1}^{n} r_i f_i(t_i). \tag{4}$$

Note that the robot may pass throught some edges $e_{i,j}$ more than once, in which case $d_{i,j}$ is included once each time $e_{i,j}$ is enumerated in (3). That is, E_{tr} is a multi-set. We define $T := \{t_1,\ldots,t_i\}$, $R := \{r_1,\ldots,r_n\}$, and $F := \{f_1,\ldots,f_n\}$.

During the planning phase, one often faces the challenging task of planning ahead so as to spend the optimal amount of time to travel and to perform sensing to gain the most out of a trip. This gives rise to two OTP variants. In the first, the robot is given a time budget M_T, during which one hopes to maximize the total reward. That is,

Problem 1 (Reward-Maximizing Tourist (RMT)) *Given a 5-tuple (V,B,D,R,F) and $M_T > 0$, compute the sets E_{tr} and T such that J_R is maximized under the constraint $J_T \leq M_T$.*

We do not need to specify the edge set E because it is implicitly fixed by D. The second, equally natural problem is in a sense a dual problem of RMT, in which the goal is to minimize the time spent to achieve a predetermined reward.

Problem 2 (Budget-Minimizing Tourist (BMT)) *Given a 5-tuple (V,B,D,R,F) and $M_R > 0$, compute the sets E_{tr} and T such that J_T is minimized under the constraint $J_R \geq M_R$.*

III. MIP Models for BMT and RMT

In this section, we propose mixed integer programming (MIP) models for solving RMT and BMT. First, we describe an MIP model derived from an existing one for the orienteering problem (OP) that applies to RMT and BMT problems with $|B| = 1$ (*i.e.*, a single base) and linear reward functions. Then, the MIP model is generalized to allow multiple bases and arbitrary reward functions. We point out that the proposed problems are computationally intractable, which can be easily shown, given their similarity to TSP and OP.

A. MIP Model for a Single Base and Linear Rewards

In this subsection, we introduce an MIP model for RMT and BMT with a single base and with the set F being linear functions. The model is partially based on models from [11], [14]. Without loss of generality, let the robot start from v_1. Because the reward at a given POI only depends on the total time spent at the POI, we also assume that the time the robot spent at a POI is spent during a single visit to the POI. When the robot spends time at a POI, we say it *stays* at the POI. With these assumptions, the robot will eventually have stayed at some ℓ POIs with the order $v_{s_1},\ldots,v_{s_\ell}$, and have spent time $t_{s_1},\ldots,t_{s_\ell}$ at these POIs. For $i \notin \{s_1,\ldots,s_\ell\}$, $t_i = 0$.

Although the robot only needs to stay at a POI at most once, it may need to pass through a POI multiple times (*e.g.*, if the POI is a travel hub). To distinguish these two types of visits to a POI, we perform a transitive closure on the set D. That is, we compute all-pairs shortest paths for $v_i, v_j \in V, 1 \leq i,j \leq n$. This gives us a set of shortest directed paths $P = \{p_{i,j}\}$ with corresponding lengths $D' = \{d'_{i,j}\}$. We say that the robot *takes* a path $p_{i,j}$ if it stays at v_j immediately after staying at v_i, except when the robot starts and ends its trip at v_1. With this update, the robot's final tour is simply $p_{s_1,s_2},\ldots,p_{s_\ell,s_1}$. Let x_{ij} be a binary variable with $x_{ij} = 1$ if and only if $p_{i,j}$ is taken by the robot.

The number of times that the robot stays at (resp. leaves after staying) a POI vertex v_i is $\sum_{j=1,j\neq i}^{n} x_{ji}$ (resp. $\sum_{j=1,j\neq i}^{n} x_{ij}$). Both summations can be at most one since by assumption, the robot never stays at a POI twice. The tour constraint then says they must be equal, *i.e.*, $\sum_{j=1,j\neq i}^{n} x_{ji} = \sum_{j=1,j\neq i}^{n} x_{ij}$. Let x_i be the binary variable indicating whether the robot stayed at v_i. We have the following edge-use constraints

$$\sum_{j=1,j\neq i}^{n} x_{ij} = \sum_{j=1,j\neq i}^{n} x_{ji} = x_i \leq 1, \quad \forall 2 \leq i \leq n. \tag{5}$$

The case of $i = 1$ is special since v_1 is always visited, even if the robot does not actually *stay* at v_1. For this purpose, we add a self-loop variable x_{11} at v_1 and require

$$\sum_{j=1}^{n} x_{1j} = \sum_{j=1}^{n} x_{j1} = x_1 = 1. \tag{6}$$

The constraints (5) and (6) guarantee that the robot takes a tour starting from v_1. However, they do not prevent multiple disjoint tours from being created. To prevent this from happening, a *sub-tour restriction* constraint is introduced. Let $2 \leq u_i \leq n$ be integer variables for $2 \leq i \leq n$. If there is a single tour starting from v_1, then u_i can be chosen to satisfy the constraints (see [11], [14] for the technical details)

$$u_i - u_j + 1 \leq (n-1)(1-x_{ij}), \quad 2 \leq i,j \leq n, i \neq j. \tag{7}$$

With the introduction of the variables $\{x_{ij}\}$, the time spent by the robot is given by

$$J_T = \sum_{i=1}^{n} \sum_{j=1,j\neq i}^{n} x_{ij} d'_{ij} + \sum_{i=1}^{n} t_i. \tag{8}$$

To represent the total reward J_R, we introduce a continuous variable $w_i, 1 \leq i \leq n$, to denote the reward collected at v_i. For a linear f_i, λ_i, the learning rate, is simply the slope of f_i. The reward w_i and the visiting time t_i then satisfy

$$w_i \leq r_i x_i, \quad w_i = t_i \lambda_i, \tag{9}$$

The first constraint in (9) allows reward only if the robot stays at v_i and limits the maximum reward at r_i. The second constraint reflects the linear dependency of the reward w_i over the visiting time t_i. The total reward J_R is simply

$$J_R = \sum_{i=1}^{n} w_i. \tag{10}$$

Solving RMT with a single base and linear reward functions can then be encoded as an MIP maximizing J_R subject to

$J_T \leq M_T$, (5), (6), (7),and (9). Similarly, solving BMT with a single base and linear reward functions can be encoded as an MIP minimizing J_T subject to $J_R \geq M_R$, (5), (6), (7), and (9).

B. Incorporating Multiple Bases

For the case of $|B| > 1$, to enable the selection of any particular $v_i \in B$, a virtual *origin* vertex o is created, which is both a source and a sink. Then, each base vertex v_i is split into two copies, v_i^{in} and v_i^{out}. The edges connecting v_i to other POI vertices of V are split such that all edges going from v_i to other POI vertices are now rooted at v_i^{out} and all edges connecting other POI vertices to v_i are now ending at v_i^{in}. In addition, two crossover edges between v_i^{in} and v_i^{out} are added, one in each direction. Lastly, an outgoing edge from o to v_i^{out} and an incoming edge from v_i^{in} to o are added. An illustration of this gadget is given in Figure 2. The gadget is duplicated for every element of B using the same origin vertex o. The basic MIP model from the previous subsection is then updated to

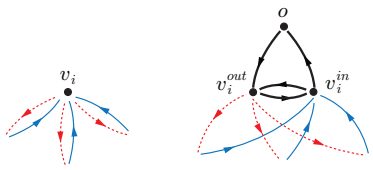

Fig. 2. [left] A base vertex v_i and its outgoing (dotted) and incoming (solid) edges. [right] The gadget that split v_i into v_i^{in} and v_i^{out}, along with the split edges and the newly added four (bold) edges.

enable the routing of the robot through at least one element of B. For each $v_i \in B$, we create four additional binary variables to represent whether the four newly added edges are used in a solution. These variables are $x_i^{o,out}$ (edge from o to v_i^{out}), $x_i^{in,o}$ (edge from v_i^{in} to o), $x_i^{out,in}$ (edge from v_i^{out} to v_i^{in}), and $x_i^{in,out}$ (edge from v_i^{in} to v_i^{out}). To ensure that at least one vertex of B is used, we add the constraint

$$\sum_{v_i \in B} x_i^{o,out} = 1. \tag{11}$$

The edge-use constraints also need to be updated accordingly. Due to the vertex split for vertices from the set B, we have two sets of such edge-use constraints. The constraint (5) now applies to all non-base vertices. The constraint (6) is updated for all base vertices $v_i \in B$ to

$$\sum_{j=1, j\neq i}^{n} x_{ij} + x_i^{out,in} - x_i^{in,out} - x_i^{o,out} = 0, \tag{12}$$

$$\sum_{j=1, j\neq i}^{n} x_{ji} + x_i^{out,in} - x_i^{in,out} - x_i^{in,o} = 0. \tag{13}$$

With constraint (11), o goes to exactly one v_i^{out} and later returns to v_i^{in}. Then, constraints (12) and (13), along with the existing edge-use constraint (5), ensure that one or more tours are created. Finally, to prevent multiple tours from being created, we update the variables u_i's to $1 \leq u_i \leq n$ for $1 \leq i \leq n$. For a base vertex $v_i \in B$, we add the constraint

$$u_i - u_j + 1 \leq (2 - x_{ij} - x_i^{in,out})n. \tag{14}$$

If v_i is not a base vertex, we require

$$u_i - u_j + 1 \leq (1 - x_{ij})n. \tag{15}$$

Constraints (14) and (15) replace the constraint (7). The constraint (15) has the same effect as the constraint (7) in preventing a separate tour from being created. For base vertices, when $x_i^{in,out} = 1$, which is the case when $x_i^{o,out} \neq 1$, the constraint (14) is the same as (15). If $x_i^{o,out} = 1$, then (14) becomes $u_i - u_j + 1 \leq n + (1 - x_{ij})n$, which always holds. That is, the constraint (14) treats the selected base vertex differently.

C. Linearization of Arbitrary Reward Functions

To accommodate arbitrary reward structure into our MIP model, a linearization scheme is used. We show that, with carefully constructed linear approximations of f_i's, arbitrarily optimal MIP models can be built. The basic idea behind our linearization scheme is rather simple. Given a C^1 continuous $f_i \in [0,1]$ with $f_i'(0) \geq \lambda > 0$, it can be approximated to arbitrary precision with a continuous, piecewise linear function $\widetilde{f}_i$ such that for arbitrary $\varepsilon > 0$ and all $t_i \geq 0$,

$$\frac{|f_i - \widetilde{f}_i|}{f_i} \leq \varepsilon, \tag{16}$$

with $\widetilde{f}_i$ having the form (see, *e.g.*, Figure 3)

$$\widetilde{f}_i = \begin{cases} a_{i,1}t_i + b_{i,1}, & 0 \leq t_i \leq t_{i,1} \\ a_{i,2}t_i + b_{i,2}, & t_{i,1} \leq t_i \leq t_{i,2} \\ \ldots, & \ldots \\ a_{i,k_i}t_i + b_{i,k_i}, & t_{i,k_i-1} \leq t_i \leq \infty \end{cases} \tag{17}$$

A procedure for computing such an $\widetilde{f}_i$ is provided by Theorem 3 in Section IV.

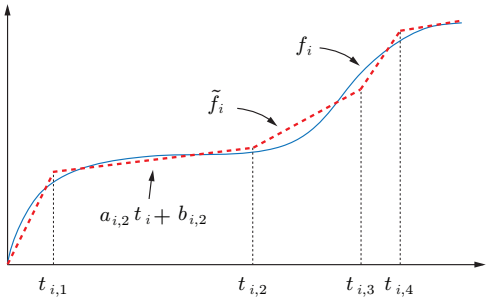

Fig. 3. Approximation of some f_i with a continuous, piecewise linear function (bold dashed line segments). The approximation is concave between $[0, t_{i,2}]$, $[t_{i,2}, t_{i,3}]$, and so on.

Once a particular $\widetilde{f}_i$ is constructed, the constraints on the reward w_i must be updated. To make the explanation clear, we use the $\widetilde{f}_i$ from Figure 3 as a concrete example. Starting from $t_i = 0$, we introduce a new continuous variable t_i^1 over the first maximally concave segment of f_i. In the case of the $\widetilde{f}_i$ in Figure 3, the first maximally concave segment contains two line segments, ending at $t_{i,2}$. In this case, we have $0 \leq t_i^1 \leq t_{i,2}$. To represent the reward obtained over the first maximally

concave segment, a continuous variable w_i^1 is introduced, which satisfies $w_i^1 \le a_{i,1}t_i^1 + b_{i,1}$ and $w_i^1 \le a_{i,2}t_i^1 + b_{i,2}$.

Then, for the next maximally concave segment, another continuous variable t_i^2 is introduced. In our example, the second maximally concave segment contains one line segment and thus

$$t_{i,2} \le t_i^2 \le t_{i,3}. \tag{18}$$

We need to ensure that t_i^2 is active only if t_i^1 is maximized. We achieve this through the introduction of an additional binary variable x_i^2, which is set to satisfy the constraint $x_i^2 \le t_i^1/t_{i,2}$. The constraint ensures that $x_i^2 = 1$ only if t_i^1 is maximized and takes the value $t_{i,2}$. To avoid potential numerical issues that may prevent $x_i^2 = 1$ from happening, in practice, we may write the constraint as $x_i^2 \le (t_i^1 + \delta)/t_{i,2}$, in which δ is a small positive real number. We can then activate t_i^2 through $t_i^2 \le x_i^2(t_{i,3} - t_{i,2}) + t_{i,2}$, which also renders the constraint (18) unnecessary. The reward for this second maximally concave segment, w_i^2, is then

$$w_i^2 \le a_{i,3}t_i^1 + b_{i,3} - (a_{i,2}t_{i,2} + b_{i,2}).$$

After all of $\widetilde{f}_i$ are encoded as such, we combine the individual time and reward variables into t_i and w_i as

$$t_i = t_i^1 + (t_i^2 - t_{i,2}) + \ldots, \tag{19}$$

$$w_i = w_i^1 + w_i^2 + \ldots. \tag{20}$$

We note that the additional constraints that are introduced is proportional to the complexity of $\widetilde{f}_i$. It can be shown that the overall MIP model constructed in this way allows arbitrary approximations of the original problem.

Theorem 1 *Given an RMT instance specified by a 5-tuple (V,B,D,R,F), $M_T > 0$, and a positive real number ε, a $(1+\varepsilon)$-optimal solution of this RMT instance can be computed by solving a mixed integer programming problem, obtained over a $(1+\varepsilon/2)$ piece-wise linear approximation of F.*

For BMT, since time is split between traveling and actually staying at POIs, a direct $(1+\varepsilon)$-optimality assurance cannot be established. Nevertheless, for a BMT instance requiring $M_R > 0$, assuming that the optimal solution requires J_T^* time, we can guarantee that a reward of at least $(1-\varepsilon)M_R$ is achieved using time no more than J_T^*.

Theorem 2 *Given a BMT instance specified by a 5-tuple (V,B,D,R,F), $M_R > 0$, and a positive real number ε, let its solution have a required total time of J_T^*. Then, an MIP model can be constructed that computes a solution with $J_R \ge (1-\varepsilon)M_R$ and $J_T \le J_T^*$.*

IV. The Algorithm and its Analysis

The overall algorithm construction is outlined in Algorithm 1. In Line 1 of the algorithm, it computes all-pairs shortest paths and their respective lengths using a transitive closure based algorithm, for example, the Floyd-Warshall algorithm [19], [20]. Then, in Lines 2-4, the algorithm computes a piece-wise linear $(1+\varepsilon/2)$-approximation of each $f_i \in F$, if necessary. Finally, once D' is computed and all of $\widetilde{F}$ is built, Lines 5-11 of the algorithm can be carried out according to the steps outlined in Section III. In the rest of this section, we cover two important properties of our algorithm.

Algorithm 1: OTP Planner

Input : V,B,D,R,FM_T (or M_R), and ε
Output: J_R^* (or J_T^*) and E_{tr}

```
%Compute all pairs of shortest paths
```
1 $(P',D') \leftarrow$ FloydWarshall(V,D)
```
%Compute for each reward function a
 piece-wise linear approximation
```
2 **for** $f_i \in F, 1 \le i \le n$ **do**
3 | $\widetilde{f}_i \leftarrow$ ComputeEpsilonApproximation$(f_i, \varepsilon/2)$
4 **end**
```
%Set up and solve the MIP
```
5 $(V',D') \leftarrow$ VertexSplit(V,B,D') ; `%Split bases`
6 BuildModel$(V',D,R,\widetilde{F})$; `%Build the MIP model`
7 **if** M_T *is given* **then**
8 | Set $J_T \le M_T$ and maximize J_R ; `%Maximize reward`
9 **else**
10 | Set $J_R \ge M_R$ and minimize J_T ; `%Minimize time`
11 **end**
12 **return** J_R^*, *the maximum reward (or* J_T^*, *the minimum time), and the associated* E_{tr}

A. Finite Complexity of Piece-Wise Linear Approximation

In Section III, we mentioned that a reasonably nice reward function can be approximated to arbitrary precision using a piece-wise linear function, which is not difficult to imagine. However, to encode the approximated piece-wise linear function into the MIP model, the function must have finitely many line segments, which can always be achieved.

Theorem 3 *Let $f \in [0,1]$ be a C^1 continuous, non-decreasing function with $f(0) = 0$ and $f'(0) \ge \lambda$ for some fixed $\lambda > 0$. For any given $\varepsilon > 0$, there exists a piece-wise linear approximation of f containing only a finite number of line segments, denoted $\widetilde{f}$, such that*

$$\frac{|f(t) - \widetilde{f}(t)|}{f(t)} \le \varepsilon. \tag{21}$$

B. The Anytime Property

A very useful property of Algorithm 1 that we obtain for free is that it yields an *anytime* algorithm.[2] The anytime property is a direct consequence of solving the MIP models for RMT and BMT using an MIP solver, which generally use some variations of the branch-and-bound algorithm [21]. Roughly speaking, a branch-and-bound algorithm works with

[2]We note that both RMT and BMT admit trivial non-optimal solutions that can be easily computed. For example, for RMT a zero-reward trip requires no computational effort.

a (high-dimensional) polytope that contains all the feasible solutions to an optimization problem. The algorithm then iteratively partitions the polytope into smaller ones and truncates more and more of the polytope that are known not to contain the optimal solution. After some initial steps, a tree structure is built and the leaves of the tree contain portions of the original feasibility polytope that are still active. For each of these polytopes, suppose we are working on a maximization problem, it is relatively easy to locate a feasible solution with the correct integrality condition (*i.e.*, a feasible solution in which binary/integer variables get assigned binary/integer values). The maximum of all these feasible solution is then a lower bound of the optimal value. On the other hand, it is also possible to compute for each leaf the maximum achievable objective without respecting the integrality constraints, which yields an upper bound on the optimal value. The difference between the two bounds is often referred to as the *gap*. When the gap is zero, the optimal solution is found. Over the running course of a branch-and-bound algorithm, if the gap gradually decreases, an anytime algorithm is obtained.

For our particular problems, the anytime property is quite useful since computing the true optimal solution to the (potentially approximate) MIP model for RMT and BMT can be very time consuming. We will see in Section V that for medium sized problems, a 1.2-optimal solution, which is fairly good for practical purposes, can often be computed quickly.

V. Computational Experiments

In this section, we evaluate our proposed algorithm in several computational experiments. In these experiments, we look at the solution structure, computational performance, and an application to planning a day tour of Istanbul. The simulation is implemented using Java. Gurobi [22] is used as the MIP solver. Our computational experiments were carried out on an Intel Core-i7 3930K PC with 64GB of memory.

A. Anytime Solution Structure

Our first set of experiments was performed over a randomly generated example, created in the following way. The example contains 30 uniformly randomly distributed POIs in a 10×15 rectangle (see Figure 4). Each POI v_i is associated with a $\lambda_i \in [1,2)$ and an $r_i \in [1,2)$ that were both uniformly randomly selected. The λ_i's and r_i's are selected not to vary by much because we expect that in practice, this will present a more difficult choice for a mobile robot. For f_i, both linear (*e.g.*, with the form (1)) and exponential (*e.g.*, with the form (2)) types were used, with the learning rates specified by the λ_i's. We set $\varepsilon = 0.05$ when we approximate the non-linear f_i's with piece-wise linear functions. Note that $\varepsilon = 0.05$ yields a 1.1-optimal MIP model for exponential f_i's. These steps determine the sets V, R, F, $\widetilde{F}$. We let B to be the set $\{v_1, v_9, v_{17}, v_{25}\}$. For deciding E and D, we let there be an edge between two POI vertices v_i, v_j if the Euclidean distance between them is no more than 10. Finally, the constraints were set as follows. For RMT, $M_T = 50$ for both linear and exponential f_i's. For BMT, $M_R = 30.55$ for linear f_i's and $M_R = 25.78$ for exponential f_i's. These M_R's were selected because they are the optimal J_R value for the respective RMT problems with $M_T = 50$.

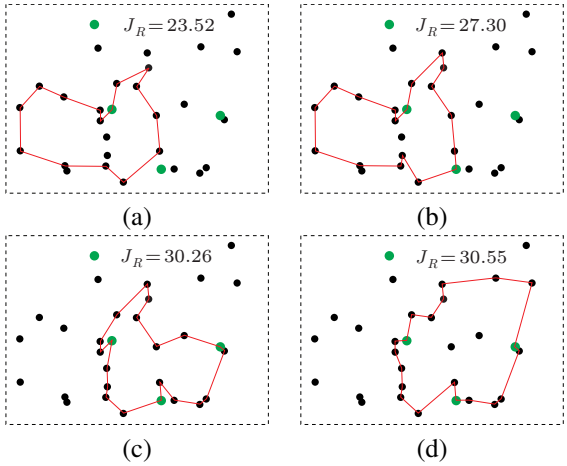

Fig. 4. Figures (a) - (d): POIs visited by the best solution to the RMT problem after the gap dips just below 50%, 20%, 10%, and 5%, respectively. The solution obtained after the gap dips below 5% is in fact the optimal solution for this particular example. The black and the green dots are the POIs and the green dots are the base vertices. The solutions are returned after 1.05, 2.16, 3.70, and 10.2 seconds of computation, respectively.

For each problem instance, we extract the solution after the gap becomes no more than $100\%, 50\%, 20\%, 10\%, 5\%, 1\%$, and 0%. Some of these solutions for the RMT instance with linear reward functions are illustrated in Figure 4. Because the large number of POIs involved, we do not list the computed t_i's but point out that, in the linear case, when the set of POIs for staying is selected, it is always beneficial to exhaust the reward at POIs with the largest learning rate since time is best used this way. The computation of these four solutions took $1.05, 2.16, 3.70$, and 10.2 seconds, respectively. Confirming that the last solution (Figure 4(d)) is indeed the optimal solution took 76 seconds. For the BMT instance with $M_R = 30.55$, we obtained a similar set of plots, which is omitted here due to space limit.

For exponential rewards, similar results were obtained. The optimal tours for RMT and BMT are illustrated in Figure 5, which, as expected, are the same. Computing the optimal solution to these more complex 1.1-optimal MIP models took 27.6 and 30.1 seconds, respectively.

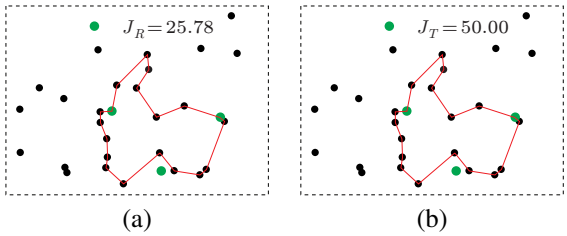

Fig. 5. (a) Optimal solution to RMT with exponential reward functions and $M_T = 50.00$. (b) Optimal solution to BMT with exponential reward functions with $M_R = 25.78$.

B. Computational Performance

Since the models for RMT and BMT attempt to solve an NP-hard problem precisely (note that the problem after

TABLE I
COMPUTATION TIME FOR SOLVING RMT AND BMT OVER POIS LOCATED AT THE LATTICE POINTS ON VARIOUS SIZED INTEGER GRIDS.

grid size	problem	reward	MIP gap (100% = 2-optimal, 0% = optimal)						
			100%	50%	20%	10%	5%	1%	0%
4×5	RMT	linear	0.085s (10)	0.135s (10)	0.203s (10)	0.261s (10)	0.675s (10)	2.285s (10)	2.357s (10)
	BMT	linear	0.070s (10)	0.108s (10)	0.271s (10)	0.571s (10)	0.974s (10)	1.101s (10)	1.102s (10)
	RMT	exponential	0.149s (10)	0.171s (10)	0.240s (10)	0.388s (10)	0.471s (10)	1.293s (10)	1.343s (10)
	BMT	exponential	0.061s (10)	0.090s (10)	0.174s (10)	0.364s (10)	0.505s (10)	0.605s (10)	0.608s (10)
6×7	RMT	linear	0.683s (10)	0.687s (10)	0.816s (10)	1.009s (10)	5.790s (10)	161.3s (7)	209.8s (7)
	BMT	linear	0.501s (10)	0.514s (10)	6.308s (10)	31.76s (10)	79.22s (10)	127.9s (10)	129.0s (10)
	RMT	exponential	0.870s (10)	0.914s (10)	1.784s (10)	5.268s (10)	17.91s (10)	182.6s (8)	234.6s (8)
	BMT	exponential	0.701s (10)	0.715s (10)	3.718s (10)	11.37s (10)	78.40s (10)	79.43s (8)	80.87s (8)
8×10	RMT	linear	2.272s (10)	2.443s (10)	2.953s (10)	21.58s (10)	87.13s (10)	454.6s (3)	809.0s (1)
	BMT	linear	2.188s (10)	2.382s (10)	3.111s (10)	20.75s (10)	134.1s (9)	284.2s (6)	295.2s (6)
	RMT	exponential	2.134s (10)	2.345s (10)	5.664s (10)	22.75s (10)	67.28s (10)	342.5s (4)	498.5s (3)
	BMT	exponential	2.530s (10)	2.849s (10)	20.64s (10)	79.32s (10)	274.6s (9)	492.9s (6)	524.7s (6)
10×20	RMT	linear	17.31s (10)	17.31s (10)	18.96s (10)	98.66s (10)	433.9s (7)	N/A	N/A
	BMT	linear	43.28s (10)	48.84s (10)	93.40s (10)	241.9s (9)	346.8s (4)	N/A	N/A
	RMT	exponential	17.33s (10)	26.87s (10)	48.06s (9)	59.64s (5)	317.3s (1)	N/A	N/A
	BMT	exponential	37.17s (10)	44.29s (10)	241.3s (10)	424.2s (6)	435.4s (1)	N/A	N/A

linearization remains NP-hard), no polynomial time algorithm exists unless P = NP. Therefore, our evaluation of the algorithm's computational performance is limited to an empirical one. For this, two large sets of computations are performed. In the first set of computations, rectangular grids of various sizes were constructed. The POIs reside on the lattice points on these grid, with the reward and learning rate selected uniformly randomly from $[1,2)$. Vertices $n/3$ and $2n/3$ are selected as base vertices. For each choice of grid sizes, 10 example problems are created. For the RMT instances, a time budget of 1.5 times the grid perimeter is used. For the BMT instances, a reward requirement of 0.6 times the grid perimeter is used. These constraints are chosen to allow the tour to go through 10% to 25% of the total POIs. For both RMT and BMT instances, we perform computations with both linear and exponential reward functions (with 5% linearization). The average time, in seconds, required to compute a solution up to given accuracy is listed in Table I. The numbers in the parentheses denote the number of times, out of a total of ten, that the computation completed within a limit of 900 seconds.[3] Due to the combinatorial nature of the branch-and-bound algorithm, quality improvement is not always gradual. For example, for the 10×20 grid and linear RMT for all 10 instances the gap went from above 100% to below 50%, causing the time taken for these two scenarios to be identical.

Our second set of computations generates the POI locations uniformly randomly according to the same rules used in Section V-A, in a $|V|\times1.2|V|$ rectangle. Then, for RMT instances, a time budget of $4\sqrt{|V|}$ is used. For BMT instances, a reward requirement of $2\sqrt{|V|}$ is used. The rest of the setup is done similarly as in the grid case. Representative computational result over the RMT instances is listed in Table II.

From the computational experiments, we observe that in the grid case, for up to 200 POIs, the proposed method can compute a 1.2-optimal (corresponding to a 20% gap) MIP solution for almost all instances (199 out of 200 instances), under very reasonable computation time. Moreover, for up to 80 POIs, the method can compute a 1.05- optimal MIP solution for almost all instances (158 out of 160 instances). When the POIs are selected randomly, the computation seems to be more challenging. Computing 1.2-optimal MIP solution starts to become challenging when there are more than 40 POIs. The difficulty seems to come from the fact that randomly selected POIs can potentially be packed more densely in certain local regions. Nevertheless, we were still able to compute 1.5-optimal MIP solutions in most of the cases when there are 100 POIs. Overall, the two large sets of computations suggest that our algorithm can be used to do itinerary planning for practical-sized instances in large cities.

C. Application to Intelligent Itinerary Planning

As a last computational example, we illustrate how our work may be applied to other planning problems such as autonomous trip planning. For this purpose, we show how to use real data to compute a day tour of Istanbul over 20 attractions (POIs). These 20 POIs are selected by taking the top-ranked attractions from TripAdvisor's[4] city guide for Istanbul. We select the top 20 POIs that are not general areas and have at least 300 user reviews. The first few of these POIs are (the ordering is by the POI's rank): Suleymaniye Mosque, Rahmi M. Koc Museum, Rustem Pasha Mosque, Hagia Sophia Museum, Kariye Museum, and Basilica Cistern.

After the POIs are selected, we compute the maximum reward of these POIs using the formula $\sqrt[3]{n_{review}}+10-rank/5$, in which n_{review} is the total number of reviews received for the POI on TripAdvisor and *rank* is the POI's rank on TripAdvisor. The attractions are mostly museums and architectural sites, to which we assign the learning rates of $1-0.01r_i$, *i.e.*, we expect a tourist to spend more time at more renowned POIs. Using Google Map[5], we extracted the pair-wise distances between any two of these POIs and build the sets E and D. The base vertex set is selected to contain the 1st, 6th, 11th,

[3]We opted to use a table instead of a graph or a plot as a table is more compact and allow us to show the number of successful runs easily.

[4]http://www.tripadvisor.com

[5]http://maps.google.com.

TABLE II
COMPUTATION TIME FOR SOLVING RMT AND BMT OVER POIS THAT ARE UNIFORMLY RANDOMLY SELECTED.

# of samples	problem	reward	MIP gap (100% = 2-optimal, 0% = optimal)						
			100%	50%	20%	10%	5%	1%	0%
20	RMT	linear	0.118s (10)	0.236s (10)	0.730s (10)	2.645s (10)	4.832s (10)	5.887s (10)	5.92 s (10)
	RMT	exponential	0.274s (10)	0.379s (10)	3.780s (10)	6.499s (10)	13.38s (10)	17.49s (10)	17.57s (10)
42	RMT	linear	6.058s (10)	13.73s (10)	49.33s (8)	164.5s (6)	262.9s (3)	170.2s (1)	187.8s (1)
	RMT	exponential	14.13s (10)	24.80s (10)	93.65s (10)	132.2s (6)	288.2s (4)	375.9s (3)	381.1s (3)
100	RMT	linear	54.26s (10)	57.26s (10)	439.3s (8)	N/A	N/A	N/A	N/A
	RMT	exponential	59.45s (10)	125.3s (9)	309.0s (5)	790.8s (1)	N/A	N/A	N/A
200	RMT	linear	40.26s (10)	255.0s (8)	N/A	N/A	N/A	N/A	N/A
	RMT	exponential	34.50s (10)	229.5s (8)	N/A	N/A	N/A	N/A	N/A

and 16th ranked POIs. With these parameters, we solve the RMT problem with exponential reward functions and a time budget of 9 hours. From the solution (an exact solution to the 1.1-optimal MIP model, computed in about five seconds) we extracted the itinerary listed in Table III. The itinerary visits 14 POIs and yields a reward of 115 out of a total possible reward of 380. Visual inspections and consulting people local to Istanbul both suggest that the itinerary is a reasonable one.

TABLE III
A 9-HOUR COMPUTED ITINERARY IN ISTANBUL.

1	Start from the Suleymaniye Mosque, stay for 0.84 hour
2	Take a taxi to Topkapi Palace (8 min), stay for 0.88 hour
3	Take a taxi to Kucuk Ayasofya Camii (6 min), stay for 0.14 hour
4	Walk to Blue Mosque (6 min), stay for 0.90 hour
5	Walk to Basilica Cistern (4 min), stay for 0.90 hour
6	Walk to Hagia Sophia Museum (4 min), stay for 0.93 hour
7	Walk to Gulhane Park (4 min), stay for 0.11 hour
8	Walk to Archaeological Museums (2 min), stay for 0.78 hour
9	Take a taxi to Rustem Pasha Mosque (6 min), stay for 0.78 hour
10	Take a taxi to Rahmi M. Koc Museum (9 min), stay for 0.76 hour
11	Take a taxi to Kariye Museum (8 min), stay for 0.82 hour
12	Take a taxi and return to Suleymaniye Mosque (12 min)

VI. CONCLUSION

In this paper, we have proposed the Optimal Tourist Problem that ties together the problem of maximizing information collection efforts at points-of-interest (POIs) and minimizing the required time spent on traveling between the set of discrete, spatially distributed POIs. A particular novelty is that our formulation encompasses a general class of time-based reward functions, which can be used to model robot sensory reward functions or learning curves for tourists. For solving the two variants of OTP, RMT and BMT, we construct an exact (when reward function is linear) or an arbitrarily optimal (when reward function is non-linear) MIP model that gives rise to an anytime algorithm for solving such problems. Computational results suggest that our algorithm is applicable to practical informative path planning problems.

REFERENCES

[1] S. Alamdari, E. Fata, and S. L. Smith, "Persistent monitoring in discrete environments: Minimizing the maximum weighted latency between observations," *The International Journal of Robotics Research*, vol. 33, no. 1, pp. 138–154, 2014.

[2] S. L. Smith, M. Schwager, and D. Rus, "Persistent robotic tasks: Monitoring and sweeping in changing environments," *IEEE Transactions on Robotics*, vol. 28, no. 2, pp. 410–426, April 2012.

[3] J. Yu, S. Karaman, and D. Rus, "Persistent monitoring of events with stochastic arrivals at multiple stations," *IEEE Transactions on Robotics*, vol. 31, no. 3, pp. 521–535, 2015.

[4] Z. W. Lim, D. Hsu, and W. S. Lee, "Adaptive informative path planning in metric spaces," in *Proceedings Workshop on Algorithmic Foundations of Robotics*, 2014.

[5] D. Golovin and A. Krause, "Adaptive submodularity: Theory and applications in active learning and stochastic optimization," *Journal of Artificial Intelligence Research*, vol. 42, pp. 427–486, 2011.

[6] L. E. Kavraki, P. Svestka, J.-C. Latombe, and M. H. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE Transactions on Robotics & Automation*, vol. 12, no. 4, pp. 566–580, Jun. 1996.

[7] S. M. LaValle, "Rapidly-exploring random trees: A new tool for path planning," Iowa State University, Tech. Rep., Oct 1998, computer Science Department TR 98-11.

[8] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *International Journal of Robotics Research*, vol. 30, no. 7, pp. 846–894, June 2011. [Online]. Available: http://ares.lids.mit.edu/papers/Karaman.Frazzoli.IJRR11.pdf

[9] G. A. Hollinger and G. S. Sukhatme, "Sampling-based motion planning for robotic information gathering," in *Robotics: Science and Systems*, 2013.

[10] I. Chao, B. Golden, and E. Wasil, "Theory and methodology - the team orienteering problem," *European Journal of Operational Research*, vol. 88, pp. 464–474, 1996.

[11] P. Vansteenwegen, W. Souffriau, and D. V. Oudheusden, "The orienteering problem: A survey," *European Journal of Operational Research*, vol. 209, pp. 1–10, 2011.

[12] D. Gavalas, C. Konstantopoulos, J. Mastakas, and G. Pantziou, "A survey on algorithmic approaches for solving tourist trip design problems," *Journal of Heuristics*, vol. 20, no. 3, pp. 291–328, 2014.

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

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

[15] M. De Choudhury, M. Feldman, S. Amer-Yahia, N. Golbandi, R. Lempel, and C. Yu, "Automatic construction of travel itineraries using social breadcrumbs," in *Proceedings of the 21st ACM conference on Hypertext and hypermedia*, 2010, pp. 35–44.

[16] S. Basu Roy, G. Das, S. Amer-Yahia, and C. Yu, "Interactive itinerary planning," in *Proceedings IEEE 27th International Conference on Data Engineering (ICDE)*, 2011, pp. 15–26.

[17] H. Yoon, Y. Zheng, X. Xie, and W. Woo, "Social itinerary recommendation from user-generated digital trails," *Personal and Ubiquitous Computing*, vol. 16, no. 5, pp. 469–484, 2012.

[18] C. Chekuri and M. Pál, "A recursive greedy algorithm for walks in directed graphs," in *Proceedings 46th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, 2005, pp. 245–253.

[19] R. W. Floyd, "Algorithm 97: shortest path," *Communications of the ACM*, vol. 5, no. 6, p. 345, 1962.

[20] S. Warshall, "A theorem on boolean matrices," *Journal of the ACM (JACM)*, vol. 9, no. 1, pp. 11–12, 1962.

[21] A. H. Land and A. G. Doig, "An automatic method of solving discrete programming problems," *Econometrica*, vol. 28, no. 3, pp. 497–520, 1960.

[22] Gurobi Optimization, Inc., "Gurobi optimizer reference manual," 2014. [Online]. Available: http://www.gurobi.com