A Spatio-Temporal Representation for the Orienteering Problem with Time-Varying Profits

Zhibei Ma and Kai Yin and Lantao Liu and Gaurav S. Sukhatme *

Abstract

We consider an orienteering problem (OP) where an agent needs to visit a series (possibly a subset) of depots, from which the maximal accumulated profits are desired within given limited time budget. Different from most existing works where the profits are assumed to be static, in this work we investigate a variant that has time-dependent profits. Specifically, the profits to be collected change over time and they follow different (e.g., independent) time-varying functions. The problem is essentially NP-hard. To tackle the challenge, we present a simple and effective framework that incorporates time-variations into the fundamental planning process. Specifically, we propose a deterministic spatio-temporal representation where both spatial description and temporal logic are unified into one routing topology. By employing existing basic sorting and searching algorithms, the routing solutions can be computed in an extremely efficient way. The proposed method is easy to implement and extensive numerical results show that our approach is time efficient and generates near-optimal solutions.

Introduction

The rapid progress of smart vehicle technologies allow us to envision that, in the future autonomous vehicles are able to carry out various tasks with little or even no human effort. We are interested in designing an efficient routing method to navigate an vehicle (agent) among a number of known and fixed depots, where each depot has some profit (e.g., score, benefit, utility, load) to be collected. If the travel time is limited, it is likely that the agent is not able to traverse all depots due to the limited time budget. This variant of routing problems is called the orienteering problem (OP) [10]. In a nutshell, an OP aims to find a tour traversing a subset of depots so that the accumulated profit collected from those traversed depots are maximized.

Different from many existing vehicle routing problems which focus on analyzing path properties in the static context (i.e., unchanging environment/topology with stationary cost metrics) [16, 11], in this work we are intrigued to investigate a time-varying variant of OPs, i.e., each depot has a time-varying profit.

Here are a few motivational examples:

- As illustrated in Fig. 1, an autonomous truck needs to pick up goods from a number of fixed depots where manufacturing factories are located. The goods are produced consistently and accumulated as time goes by — so they are time-varying. The growing speeds/patterns of goods at different factories may be nonidentical: larger factories grow faster and smaller ones grow slower. Assume the truck has sufficiently large capacity and it empties all goods from a depot when it arrives there. The objective is to find a route so that the truck will load the most goods in a given time window. Similar application scenarios also include, e.g., mail/garbage trucks and smart taxis since the volumes of mails/garbage and the number of passengers are also time-varying.

- In environmental monitoring, autonomous robots are deployed to collect environment data in order to estimate an underlying environment state. However, the environment attributes at different locations can be time-varying (e.g., dissolved chemical compounds and algae blooms in the water vary both spatially and temporally.) An important objective of environment monitoring is to plan so-called “informative paths” [4] that navigate the robots to acquire data from those most information-rich spots which best help estimate the environment. The learned environment model allows us to intervening with the environment. For
instance, given a learned ocean model with known algae bloom growing patterns, marine robots can be routed to clear up the algae blooms as efficient as possible.

In this paper, we present our initial study for the time-varying OP. We start our analysis from the single agent planning case, and attempt to establish a new framework that is different from traditional modeling and solving routines. We are showing that, a big challenge of the time-varying OP (even for single agent case) lies in the requirement of a special treatment in an extra dimension — the time dimension, which inevitably introduces extra complexity as we need to model, predict, and integrate future dynamics.

Related Works

The routing problems have been well studied topics in many research domains including the operations research, theoretical computer science, and transportation systems [2, 3, 6]. Typical routing problems involve incorporation of constraints expressed from the nature of the target problems, which also narrow the space of searching for solutions [18]. We are interested in a variant of routing problems called the orienteering problems (OPs) [10, 21]. An OP considers both travel cost (e.g., travel time) and scores/credits collected along the travel. The goal of the OP is to determine which subset of vertices to visit and in which order so that the collected score within a given period is maximized. The OPs integrate characteristics of both knapsack problems (KPs) [14] and travelling salesman problems (TSPs) [18], and OPs are NP-hard as well. In contrast to the TSP, not all vertices of an OP need to be visited due to the limited time budget. During the past a few decades, several variants of OPs have been studied, such as time-dependent OPs, Team OPs, (Team) OPs with Time Windows and OPs with stochastic profits. Recent survey papers [21, 11] have profoundly discussed state-of-the-art techniques of these variants as well as their applications.

While OPs can be formulated as mixed integer programs (MIPs), the problem size typically is too large to directly use commercial solvers. A wide range of decomposition methods such as branch-and-price algorithms have been developed so that a large-scale MIP can be decomposed into smaller problems (e.g., a master problem and a series of sub-problems) which can then be iteratively solved by commercial solvers [12, 15]. In order to reduce the heavy computational burden in decomposition methods, heuristics and metaheuristics have been extensively studied, typically including tabu-based or neighborhood search based procedures [5, 20, 23].

Although in general OPs have been well researched, the time-dependent OPs (TOPs) have received relatively less attention comparing with other variants [9, 11]. Even so, most of existing TOPs discuss the time-varying properties that are associated with the real travel time between pairwise nodes, and assume that travel time between two nodes depends on the departure time at the first (or an earlier) node [8, 7, 22, 17, 24]. Very rarely we could find the works that discuss about time-varying scores/credits of OPs. One work that share certain similarity with this proposed problem is [1], where multiple vehicles need to serve a number of clients and the profit of each client follows a decreasing function of time. The work analyzed a lower bound and upper bound based on a classic MIP formulation.

Instead of employing conventional techniques such as the column generation approach used in [1], in this work we present our first study that models and tackles the problem from a different perspective: we start from establishing a representation built from the spatial and temporal constraints, so that the time dependence attribute is transferred from the bulk MIP to a separate and intuitive representation. With that, fast approximate OP solutions can be found by employing and extending order-sensitive topological algorithms.

Problem Description and Formulation

A routing problem can be represented with a graph $G = (V, E)$, where $V$ is the set of vertices and $E$ denotes the set of edges. Let us denote the number of vertices as $|V| = n + 1$, and suppose every edge takes time to traverse. Although the travel time, in many situations, depends on the properties of traffic network such as congestion and capacity, we assume that the travel time $t_{ij}$ between two vertices $v_i$ and $v_j$ is time-invariant for simplicity*. We associate each vertex $v_i$ with a time-varying value, called profit, which is denoted by $f_i(t) \geq 0$ at time $t \geq 0$. Here $f_i(\cdot)$ is of arbitrary nonlinear function form. We assume that function $f_i(\cdot)$ is known or can be predicted or approximated.

Suppose that the agent (vehicle) starting from a dummy node $v_0$ at time 0 travels across a subset of vertices on the graph $G$. When the agent visits vertex $v_i$ at time $t_i$, it will collect the profit $f_i(t_i)$. The remaining profit at the vertex $v_i$ right after the agent leaves becomes 0 and accumulates again. Additionally, the profit at $v_0$ is assumed to be 0, i.e., $f_0(t) = 0$. Assuming the agent visits each vertex once, the objective is to determine the order of a subset of vertices to visit so that total profits collected by the agent is maximized within a given planning period $T$. Note that we do not assign a specific destination to the agent, as the problem with a fixed end vertex is a special case for our problem.

Let $x_{ij} = 1$ if the agent travels from $v_i$ to $v_j$, and 0 otherwise. Let $t_0 = 0$. We also introduce extra variables $u_i$ with $u_0 = 0$ to eliminate the subtours. Then the problem in this study can be formulated as the following mixed-integer

\*It is relatively straightforward to incorporate time-dependent travel times in our proposed model described in the following section.
A Spatio-Temporal Representation

It appears a daunting task to solve the problem (1)-(8) due to the complexity of objective function. Most of existing literature in OPs either lump all constraints together and solves it by conventional solvers of MIPs [19, 21], or decouples the constraints into master-subproblem modules such as column generation [1], or use certain heuristics such as center-of-gravity heuristic [10]. In contrast, we desire to develop a framework starting from a constraint-included representation that is intuitive to understand, easy to implement from scratch, and flexible to modify and extend. In this section, we present a means for embedding constraints into a spatio-temporal representation built on which the original routing problem can be tackled by efficient methods (though it is still a NP-hard problem).

As the problem essentially aims to determine the visiting time and order of vertices, it inspires us to incorporate a time dimension to extend the 2-dimensional graph on a spatial plane to a 3-dimensional graph (topology). To make the model implementable, the time range $[0, T]$ is discretized into a sequence of time intervals $\Delta_t$ of equal length, and the interval $\Delta_t$ is used as a time unit that specifies discrete time resolution. Thus, the travel time $t_{ij}$ can be expressed as multiple times of the unit time. For example, $t_{ij} = n_{ij}\Delta_t$ and $T = n_T\Delta_t$, where $n_{ij}$ and $n_T$ are integers.

Figure 2 shows two examples that describe the basic idea. Intuitively, one can imagine that each spatial graph vertex is a di-rected edge from vertex $v_i$ to $v_j$, and only if the two vertices are spatially traversable and the time difference between the two time layers is exactly equal to the anticipated real travel time between the two vertices.

Each vertex contains a profit and the profit is time-varying. In other words, the profit values at different time layers are not the same. Therefore, the problem is equivalent to finding a path from the given start vertex at time $0$ such that the path transits other vertices within time $T$ and the total collected profit is maximized. It is worth mentioning that, the spatio-temporal edges are “directed” since each edge must start from a vertex at an earlier time layer and ends at one at a later moment; Because the time is un-diirectional and cannot travel backwards, it is impossible to form a loop or cycle on the spatio-temporal graph. As a result, such a spatio-temporal representation is equivalent to a vertex-weighted Directed Acyclic Graph (DAG). This allows us to conveniently develop our own routing method built on many existing efficient DAG algorithms.

More formally, the spatio-temporal representation is a vertex-weighted DAG, denoted by $G' = (V', E', W')$, where $V' = V \times T$, $T = \{0, \Delta_t, ..., n_T\Delta_t\}$, $E'$ is a directed edge from vertex $v'_{iu} = (v_i, t)$ to $v'_{js} = (v_j, s)$, and

$$\text{max } \sum_{j=1}^{n} x_{ij}$$

subject to

$$\sum_{j=1}^{n} x_{ij} = 1, \quad \forall i$$

$$\sum_{i=0}^{n} x_{ik} = 1, \quad \forall k$$

$$t_{ij} \leq T \cdot \sum_{i=0}^{n} x_{ij}, \quad \forall j$$

$$t_i + x_{ij} (r_{ij} + T) < T + t_j, \quad \forall i, j$$

$$1 \leq u_i \leq n, \quad \forall i \neq 0$$

$$u_i - u_j + 1 \leq n(1 - x_{ij}), \quad \forall i \neq 0, j \neq 0$$

$$x_{ij} \in \{0, 1\}, \quad \forall i, j$$

where $V$ is the set of vertices, $E$ is the set of edges, $W$ is the set of weights, $V'$ is the set of vertex-weighted DAG, $E'$ is the set of directed edges, and $W'$ is the set of directed edge weights.
process of optimizing the profits. Thus, the problem (1)-(8) related constraints can be decoupled and eliminated from the that, the time related constraints have been incorporated into customer’s constraint. For this example, we have 9 directed edges in total. With graph $G = (V, E)$ and given time limit $T$, the spatio-temporal graph $G' = (V', E', W')$ can be constructed by Alg. 1. Assume there are $n$ vertices in $G$, the time complexity of Alg. 1 is $O(n^2 T/\Delta_t)$, where there are three for loops each of which is associated with either $n$ or $T$. Also there are $n(T/\Delta_t + 1)$ vertices in $G'$, so the space complexity is $O(n T/\Delta_t)$.

Algorithm 1: SpatioTemporalGraph $(G,T,v_0,\Delta_t)$

**Input:**
1. Graph $G = (V,E)$, time limit $T$, start vertex $v_0$, time interval $\Delta_t$.

**Output:**
1. for each vertex $v_i$ in graph $G$ do
2. for each $t$ in time period $[0, T]$, time step is $\Delta_t$ do
3. $v'_{jt}.id = i$
4. $v'_{jt}.weight = f_j(t)$, put $w'_{jt}$ into the weight set $W'$
5. $v'_{jt}.profit = f_j(t)$
6. $v'_{jt}.sum = -\infty$
7. $v'_{jt}.parent = -1$
8. for each $v_i$'s neighbor $v_j$ do
9. if $t + \tau_{ij}$ is equal to $t'$ AND $t' <= T$ then
10. put edge $v_{ij}t'$ into the edge set $E'$
11. push $v'_{jt'}$ into edge $v'_{jt'}$.successors
12. update indegree of $v'_{jt'}$
13. put $v'_{jt'}$ into the vertex set $V'$
14. set the $v'_{jt'}$.sum to 0
15. return graph $G' = (V', E', W')$

Note: $f_j(t)$ is the time-varying profit function of vertex $v_j$ at time $t$. $v'_{0,v}$ is the start vertex in $G'$, and .sum is used to store accumulated profit from prior traversal along a path. The label .parent points to the predecessor vertex.

Routing Algorithm

An important advantage of the spatio-temporal graph lies in that, the time related constraints have been incorporated into such a spatio-temporal representation, so that temporal related constraints can be decoupled and eliminated from the process of optimizing the profits. Thus, the problem (1)-(8) is equivalent to finding a path $P$ from vertex $v_{0,0}$ to $v_{ks}$ on $G'$ where $s \leq T$ such that $\sum_{(v_{t,u} \in P)} w_{t,u}$ is maximum.

Since the spatio-temporal graph is essentially a DAG, we develop our routing solution via extending classic DAG algorithms. Specifically, we found that the profit maximization can be transformed to a longest path problem by accumulating the profits collected from vertices instead of summing up edge lengths along a path. We manipulate the DAG so that vertices are sorted in a topological order along the temporal dimension, and then employ a dynamic programming paradigm to compute the maximal profit path.

It is also noteworthy that, while developing a solution to the time-varying OP, we take into account of two concerns that are related to applications.

- **Specification of a Routing Destination:** Many routing problems, including classic OPs, require to specify a routing destination. The destination can be the original depot where the agent departs (e.g., a mail truck needs to return a central processing office); the destination can also be an arbitrary depot located somewhere else (e.g., a freight truck needs to pick up goods and unload them into some specified processing location that is different from the starting depot).

- Persistent Task: Many long-term missions need repetitive and persistent routing, for which specifying a routing destination is not necessary or even inappropriate. For instance, in the persistent environmental monitoring task, we do not need the robot to stop at any specified location, as the robot will need to resume to next round of routing after the completion of current one. Thus, the routing destination should be computed on the fly based on the profit optimization constrained by time $T$, instead of being manually specified.

We will show that the proposed framework works for both specified and unspecified routing destinations.

Topological Sorting of DAG in Temporal Dimension

The main purpose of topological sorting of the DAG is that, the vertices are “placed” onto different “stages” according to their temporal constraints, so that a dynamic programming structure (discussed in the following subsection) can be applied.

Formally, a topological sort of a directed graph is a linear ordering of its vertices such that for every directed edge $e_{uv}$ from vertex $u$ to vertex $v$, $u$ comes before $v$ in ordering. A topological sort of a graph requires that the graph must be a DAG. We employ a well-known algorithm developed by Kahn [13] to sort our spatial-temporal graph $G'$, with main steps pseudo-coded in Alg. 2. Briefly, we first find a list of vertices that have no incoming edges (with $\text{deg}^-(v) = 0$), and insert them into a set $S$. Note, at least one such vertex must exist in a non-empty graph. Then we traverse the set $S$. Each time we remove a vertex $v$ from $S$, and add it to the tail of the list $L$. After removing $v$, the indegree of its successors should be decreased by 1. Then we insert those vertices with updated indegree equal to 0 in the set $S$.

To analyze the time complexity, assume there are $n$ vertices in $G$, so there will be $n(T/\Delta_t + 1)$ vertices in $G'$. For each vertex $v'_{jt}$, there will be at most $n-1$ directed edges. Therefore $|E'| = O(n^2 T/\Delta_t)$, and the time complexity of Alg. 2 is $O(|V'| + |E'|) = O(n^2 T/\Delta_t)$. Some algorithms (mentioned in the text) have been omitted for brevity.
Computing Maximal Profit Path

We transfer the time-varying OP to a longest path problem in a DAG. The classic longest path problem is the problem of finding a simple path of maximum length in a given graph. We employ a dynamic programming structure to memorize incumbent maximal accumulated profit at each vertex of topologically sorted stages. Note that, in our problem, we need to optimize profits collected from vertices, instead of adding up length of edges. Therefore, instead of using the longest path update function between two successive stages:

\[
l(v) = l(u) + \tau_{uv}, \quad \text{if} \quad l(u) + \tau_{uv} > l(v),
\]

where \(l(v)\) is the largest distance from start vertex to \(v\), we utilize an update function:

\[
w.sum = v.sum + w.profit, \quad \text{if} \quad v.sum + w.profit > w.sum.
\]

Main procedures of computing the maximal profit path is described in Alg. 3. Briefly, the topologically sorted vertices \(V'\) of the spatio-temporal graph \(G'\) is used as an input. Then vertices from different stages form dynamic programming subproblems and they are updated with accumulated profits recursively, starting from \(v'_{0,0}\). Here we use \(v'_{it}.path\) to store vertices along the path from start \(v'_{0,0}\) to \(v'_{it}\) (Note, only spatial information is recorded). To prevent from forming routing subtours, we check and discard those already visited vertices before each value update.

After the completion of dynamic programming, each vertex contains information of the maximal profit path that routes from \(v'_{0,0}\) to it. Since the graph \(G'\) has incorporated the time limit \(T\), every vertex is feasible to the time constraint. To find a maximal profit path to a specified vertex (destination) \(v'_{it} \in G'\), one simply needs to enumerate all states \(v'_{it,j} \in G', \forall j\) of vertex \(v'_{it}\) and retrieve the path with the largest value. If a destination is not specified, one needs to enumerate all states of all vertices and find out the maximal one among them. The time complexity of Alg. 3 is \(O(n^2T/\Delta_t)\) due to its two for loops.

Main Routing Algorithm

With the components described above, the main algorithm is shown in Alg. 4. According to the analysis of each part, the overall time complexity of our algorithm is \(n^2T/\Delta_t\), and the space complexity is \(n^2T/\Delta_t\) as well.

---

**Algorithm 2: TopologicalSort \((G')\)**

1: \(L\): an empty list that will contain the sorted elements
2: \(S\): a set of all nodes with no incoming edges
3: while \(S\) is not empty do
4: \(\text{remove a vertex } v \text{ from } S\)
5: \(\text{add } v \text{ to the end of } L\)
6: \(\text{for each successor } u \text{ of } v \text{ do}\)
7: \(\text{\hspace{1cm}deg}^{-1}(u) = \deg^{-1}(u) - 1\)
8: \(\text{\hspace{1cm}if } \deg^{-1}(u) \text{ equals to 0 then}\)
9: \(\text{\hspace{1.5cm}insert } u \text{ into } S\)
10: \(\text{return } L \text{ (a topologically sorted order)}\)

*Note: \(G'\) is a DAG, the indegree of \(v\) is denoted as \(\deg^{-1}(v)\).*

**Algorithm 3: MaximalProfitPath \((L)\)**

1: \(v_{it}.sum = 0, i = 0, t = 0\)
2: \(\text{for each vertex } v'_{it} \text{ in topologically sorted order } L \text{ do}\)
3: \(\text{\hspace{1cm}for each vertex } v'_{it,j} \text{ in } v'_{it}.successors \text{ do}\)
4: \(\text{\hspace{2cm}if vertex } v_j \text{ is not in the } v'_{it}.path \text{ then}\)
5: \(\text{\hspace{3cm}if } v'_{it}.sum + v'_{it,j}.profit > v'_{it,j}.sum \text{ then}\)
6: \(\text{\hspace{4cm}v'_{it,j}.sum = \text{\hspace{1cm}max} \hspace{1cm}v'_{it,j}.sum, v'_{it}.sum + v'_{it,j}.profit\}\)
7: \(\text{\hspace{4cm}update the } v'_{it,j}.parent \text{ to } v'_{it}\)
8: \(\text{\hspace{4cm}v'_{it,j}.path = v'_{it}.path + v_j}\)
9: \(\text{\hspace{1cm}for each vertex } v'_{it} \text{ in } G' \text{ do}\)
10: \(\text{\hspace{2cm}find the maximal } \text{sum}\)
11: \(\text{\hspace{2.5cm}retrieve a path } P \text{ by backtracking from end to start}\)
12: \(\text{return } P\)

*Note: The label .path is a vector that stores all visited vertices from start to \(v_0\).*

---

**Algorithm 4: Time-Varying OP**

**Input:**
1: A 2D graph \(G = (V, E)\), start vertex \(v_0\), time limit \(T\), time interval \(\Delta_t\)

**Output:**
2: Construct a spatio-temporal graph \(G' = \text{SpatioTemporalGraph}(G, T, v_0, \Delta_t)\)
3: \(L = \text{TopologicalSort}(G')\)
4: \(\text{path} = \text{MaximalProfitPath}(L)\)
5: return \(\text{path}\)

---

Our method has a polynomial time complexity given a fixed \(\Delta_t\), but cannot guarantee to find the optimal solution to this NP-hard problem. The main reason for reaching at possible sub-optimality lies in that, we added a non-subtour constraint in order to (1) prevent the path from traversing back and forth among a small subset of adjacent vertices and (2) comply with the structure of dynamic programming. Such a constraint eliminates certain feasible searching space which possibly contains the optimal solution. One may regard this as a trade-off between solution quality and practical runtime. Nevertheless, our evaluation results with extensive trials show that on average our method produces near-optimal solutions.

**Discussion: Effects of Discretization**

The time interval \(\Delta_t\) plays a critical role in our framework as the total profits are actually evaluated at time steps equal to multiple \(\Delta_t\). Yet it may be impractical to select such time interval because it would result in an extremely large size of DAG. One possible implementation is to round the travel time to a value closest to some integer times of \(\Delta_t\). Hence, it is important to analyze the impact of errors due to such rounding procedure.

Assume that the profit function \(f_i(t)\) satisfies the following Lipschitz condition:

\[
|f_i(t) - f_i(s)| \leq K|t - s|, \quad \forall t, s \in [0, T], \forall i \in \{0, \ldots, n\},
\]

where \(K\) is a constant independent of \(i\). Suppose that the optimal objective value for the problem (1)-(8) is \(z\), and the optimal objective value by the proposed framework is \(z'\). Then we have the result as follows.
Figure 3: Route feature of orienteering. Δt is 1, vertex number is 200, T is 200. The start vertex is at (-49,0).

Proposition 1 Under the condition (11), we have the upper bound for the difference in two objective values:

\[ |z - z'| \leq n \cdot K \cdot \Delta t. \quad (12) \]

proof: Let \( P \) be the optimal path in the graph \( G \) for the problem (1)-(8) and let \( P' \) be the optimal path in the proposed spatio-temporal DAG \( G' \). By the construction of \( G' \), \( P \) must correspond to a path \( \tilde{P} \) in the graph \( G' \). Let \( \tilde{z} \) be the total profits (i.e. the value of objective (1) on this path) for the path \( \tilde{P} \). Since the difference between the discrete time in our framework and the original real-valued time is within \( \Delta t \), the fact that any path contains no more than \( n \) vertices and the condition (11) lead to \( |z - \tilde{z}| \leq nK\Delta t \). As \( P' \) rather than \( P \) is the optimal path in \( G' \), we have \( \tilde{z} \leq z' \). By combining these results, we should have \( z - z' \leq nK\Delta t \).

Routing Properties under Temporal Variations

To examine the effects caused by temporal variations, we manipulate the profit functions. Specifically, we divide the space into four regions (quadrants) I, II, III and IV, as shown in Fig. 4. The profit functions are the same if the vertices are in the same region, but different if not. For example, in region I, the profit function for each \( v_i \) is

\[ f_i(t) = \begin{cases} 5w_i & \text{if } t \leq T/2 \\ 0 & \text{if } t > T/2 \end{cases} \]

where \( w_i \) is the weight of vertex \( v_i \) in graph \( G \). (Keep in mind that, \( w_i \) is different from \( w_i' \) which is time-varying. Here the weight \( w_i \) refers to a fixed parameter for profit function.) Similarly, the profit functions in regions II, III and IV are

\[ f_i(t)_{II} = \begin{cases} 10w_i & \text{if } t \leq T/2 \\ 0 & \text{if } t > T/2 \end{cases} \]

\[ f_i(t)_{III} = \begin{cases} 0 & \text{if } t \leq T/2 \\ 5w_i & \text{if } t > T/2 \end{cases} \]

\[ f_i(t)_{IV} = \begin{cases} 0 & \text{if } t \leq T/2 \\ 10w_i & \text{if } t > T/2 \end{cases} \]

Figure 4 reveals that, the path first transits the vertices in region II, because during the first half \( T \), vertices in region II have larger profits than those in region III. After the path enters region I and after the time passes \( T/2 \), vertices...
in region IV contain larger profits. Such variations attract
the path to go through vertices in region IV. This example
indicates that our algorithm is sensitive to the time-varying
profit functions.

Comparison with Optimal Solution
We compare the result of our method with the optimal so-
lution that is obtained by enumerating all solutions in a
brute-force way. Because of the prohibitive time complex-
ity $O(n!)$ for searching for the optimal solution, the practical
runtime for 13 vertices requires more than 10 minutes. Thus,
we tested up to 12 vertices to compare with the optimal so-
lutions. We investigated three representative profit functions
in the form of linear, quadratic and logarithmic, respectively.
The results are shown in Fig. 5. We can see that the results of
our method are very close to those of the optimal solutions,
for all the three functions.

Figure 6 shows a group of paths produced from our
method and the optimal solution. We use a linear function
$f_i(t) = w_i t / T$ for this example. In many cases, our algo-

Comparison with Classic OP Algorithm
We also compared our method with the classic OP algo-

Path Quality under Different Time Intervals

Time interval $\Delta_t$ determines time discretization resolution,
and therefore affects the optimality as well. Table 1 shows
statistics of the route’s quality under different time intervals.
In the table, numbers in the first row are interval values, and
numbers in the first column are the number of vertices. The
remaining values are the collected profits from our method.
We can see that, in general the route’s quality is better when
the time interval is smaller. However, obviously a smaller
time interval will inevitably lead to a larger graph and thus
require a larger running time.
Figure 7: (a) Our algorithm with time-invariant profits $f_i(t) = w_i$; (b) Classic OP with time-invariant profits; (c) Our algorithm with time-varying linear profit functions; (d) Classic OP with time-varying linear profit functions.

Figure 8: (a) Performance on time-invariant profits $f_i(t) = w_i$; (b) Performance on linear function $f_i(t) = w_i t/T$.

Figure 9: Statistics of practical running time.

| Intervals $\Delta t$ | 0.1 | 0.5 | 1   | 2   | 5   | 10  |
|----------------------|-----|-----|-----|-----|-----|-----|
| 50                   | 541.1 | 530.3 | 529.2 | 522.1 | 515.1 | 498.3 |
| 100                  | 788.2 | 783.4 | 767.9 | 735.6 | 727.5 | 619.2 |
| 150                  | 950.2 | 919.0 | 909.7 | 892.9 | 823.0 | 686.8 |
| 200                  | 1129.5 | 1117.6 | 1107.1 | 1094.3 | 935.5 | 696.0 |

Table 1: Statistics of solution quality with various time intervals $\Delta t$.

Note: $T = 150$, graph within $[-50, 50] \times [-50, 50]$ on $x$-$y$ plane, profit function $f_i(t) = w_i t/T$.

Conclusion and Future Work

We presented a framework for addressing the orienteering problem with time-varying profits. Instead of following traditional mixed integer program solution routines, we develop an intuitive and effective framework that incorporates time-variations into the fundamental planning process. Specifically, we first construct a deterministic spatio-temporal representation where both spatial description and temporal logic are unified into one routing topology, and then we extend existing sorting and searching algorithms, so that the routing solutions can be computed in an extremely efficient way. Finally, we validated our method with numerical evaluations and the results show that our framework produces near-optimal solutions in a very efficient way.

In this paper we provided an initial framework to tackle the difficulties and complexities caused by the temporal variations, for a single agent planning case. In the future, we are interested in extending the current framework to multi-vehicle routing scenarios. The problem is important because as we can envision that, the profits of smart vehicles (e.g., clients of driverless taxis, goods for smart fleet) are typically time-varying and thus an efficient vehicle routing strategy is extremely important. This problem is also challenging because, the planning not only needs to solve conflicts from the spatial perspective, but also needs to take into account the profits optimization in a coordinated manner in the time dimension.
References

[1] Afsar, H. M., and Labadie, N. 2013. Team orienteering problem with decreasing profits. *Electronic Notes in Discrete Mathematics* 41:285–293.

[2] Bellman, R. 1958. On a routing problem. *Quarterly of applied mathematics* 87–90.

[3] Bellmore, M., and Nemhauser, G. L. 1968. The Traveling Salesman Problem: A Survey. *Operations Research* 16(3):538–558.

[4] Binney, J.; Krause, A.; and Sukhatme, G. S. 2013. Optimizing waypoints for monitoring spatiotemporal phenomena. *The International Journal of Robotics Research* 32(8):873–888.

[5] Chao, I.-M.; Golden, B. L.; and Wasil, E. A. 1996. The team orienteering problem. *European journal of operational research* 88(3):464–474.

[6] Christofides, N. 1976. The vehicle routing problem. *Revue française d’automatique, d’informatique et de recherche opérationnelle. Recherche opérationnelle* 10(1):55–70.

[7] de Oliveira, H. C. B., and Vasconcelos, G. C. 2010. A hybrid search method for the vehicle routing problem with time windows. *Annals of Operations Research* 180(1):125–144.

[8] Fomin, F. V., and Lingas, A. 2002. Approximation algorithms for time-dependent orienteering. *Information Processing Letters* 83(2):57–62.

[9] Garcia, A.; Arbelaitz, O.; Vansteenwegen, P.; Souffriau, W.; and Linaza, M. T. 2010. Hybrid approach for the public transportation time dependent orienteering problem with time windows. In *International Conference on Hybrid Artificial Intelligence Systems*, 151–158. Springer.

[10] Golden, B. L.; Levy, L.; and Vohra, R. 1987. The orienteering problem. *Naval research logistics* 34(3):307–318.

[11] Gunawan, A.; Lau, H. C.; and Vansteenwegen, P. 2016. Orienteering problem: A survey of recent variants, solution approaches and applications. *European Journal of Operational Research*.

[12] Gutiérrez-Jarpa, G.; Desaulniers, G.; Laporte, G.; and Marianov, V. 2010. A branch-and-price algorithm for the vehicle routing problem with deliveries, selective pickups and time windows. *European Journal of Operational Research* 206(2):341–349.

[13] Kahn, A. B. 1962. Topological sorting of large networks. *Commun. ACM* 5(11):558–562.

[14] Kellerer, H.; Pferschy, U.; and Pisinger, D. 2004. *Introduction to NP-Completeness of knapsack problems*. Springer.

[15] Keshet, M.; Ziarati, K.; Bettinelli, A.; and Vigo, D. 2016. Enhanced exact solution methods for the team orienteering problem. *International Journal of Production Research* 54(2):591–601.

[16] Laporte, G. 1992. The traveling salesman problem: An overview of exact and approximate algorithms. *European Journal of Operational Research* 59(2):231–247.

[17] Mahmoudi, M., and Zhou, X. 2016. Finding optimal solutions for vehicle routing problem with pickup and delivery services with time windows: A dynamic programming approach based on state–space–time network representations. *Transportation Research Part B: Methodological* 89:19–42.

[18] Miller, C. E.; Tucker, A. W.; and Zemlin, R. A. 1960. Integer programming formulation of traveling salesman problems. *Journal of the ACM (JACM)* 7(4):326–329.

[19] Nemhauser, G. L., and Wolsey, L. A. 1988. Integer programming and combinatorial optimization. *Wiley*, Chichester. GL Nemhauser, MWP Savelsbergh, GS Sigismondi (1992). *Constraint Classification for Mixed Integer Programming Formulations*. COAL Bulletin 20:8–12.

[20] Tang, H., and Miller-Hooks, E. 2005. A tabu search heuristic for the team orienteering problem. *Computers & Operations Research* 32(6):1379–1407.

[21] Vansteenwegen, P.; Souffriau, W.; and Van Oudheusden, D. 2011. The orienteering problem: A survey. *European Journal of Operational Research* 209(1):1–10.

[22] Verbeeck, C.; Sörensen, K.; Aghzaff, E.-H.; and Vansteenwegen, P. 2014. A fast solution method for the time-dependent orienteering problem. *European Journal of Operational Research* 236(2):419–432.

[23] Vidal, T.; Maculan, N.; Ochi, L. S.; and Vaz Penna, P. H. 2015. Large neighborhoods with implicit customer selection for vehicle routing problems with profits. *Transportation Science* 50(2):720–734.

[24] Yang, L., and Zhou, X. 2014. Constraint reformulation and a lagrangian relaxation-based solution algorithm for a least expected time path problem. *Transportation Research Part B: Methodological* 59:22–44.