Solving Stochastic On-Time Arrival Problem via Universal Value Function

Abstract

We consider Stochastic On-Time Arrival (SOTA) problem in traffic routing that we intend to acquire a routing policy to maximize the probability of arriving destination from origin in a limited time budget. Past methods based on Value Iteration have to set a budget prior to training process, that could lead to redundant calculations with different budgets on same origin-destination (o-d) pair. What's more, when new destinations are added into routing, they have to calculate the routing policies from scratch, which is a time-consuming process and could affect the algorithms' practicality. In this paper, we present Universal Value Function (UVF) to tackle these problems. Our method doesn't need to set a specific budget in advance, instead, we learn a UVF, which is a time-cost distribution, for traffic node with regards to destination in road map. Besides, in order to adapt to the new destinations quickly, we adopts a parametric model, which we call Universal Approximate Function (UAF), to inference the routing policies for new destinations with a learning process by approximating existed UVFs. Experiments' results of our method on GridWorld and a real city road map show our method can attain optimal routing polices and achieve the generalization ability for new destinations.

Introduction

When we drive outside, we usually hope to find a route with minimal cost to arrive destination. While, due to the uncertainties of traffic conditions on road, the minimal time cost route may suffer a high variance of cost that could impede our travel experience. Especially when we wish to know if we can arrive the destination in our time budget, and what's the probability of arriving on time. Imaging we hope to attend an conference in 30 mins, and we don't want to be late. There is one route with minimal cost 24 mins but also with a high risk to arrive late due to road uncertainty. Another route costs 27 mins but ensure we arrive 100% on time. Obviously the second route is we prefer to. We call this kind question as Stochastic On-Time Arrival (SOTA) problem (Fan, Kalaba, and Moore 2005). In this problem setting, we wish to find an optimal routing policy that not only help us find a path connects the origin and destination but also maximizes the probability of arriving in our time budget.

(Fan, Kalaba, and Moore 2005) studies this problem firstly in a framework of stochastic Markov Decision Process. They present successive approximation (SA) algorithm to solve SOTA problem. SA is a value iteration algorithm to find the unique optimal policy by maintaining a value function in iterations, under the assumption of a perfect model of the environment. (Fan, Kalaba, and Moore 2005)'s algorithm has inspired or become a key part in many routing methods, like (Yang et al. 2018; Samaranayake, Blandin, and Bayen 2012; Andonov and Yang 2018), but it meets some drawbacks. One is it has to set a budget prior to training. For one o-d pair, different budgets could lead to different routing polices, and the algorithm needs to be run once for every budget. Another drawback is it lacks generalization ability. For every new destination, the algorithm has to be rerun from scratch to calculate the arrival probability and get routing policy. So it has to store all routing policies for all potential o-d pairs to perform a successful routing inquiry. Besides, the slow convergence's speed, which is the original problem of this algorithm, also hinders the algorithm's application in practice. (Samaranayake, Blandin, and Bayen 2012)'s method can accelerate the calculation speed but still meets the two drawbacks.

In this paper, we propose a new method named Universal Value Function (UVF) algorithm for SOTA problem. Our method doesn't need to set a prior budget, instead, we learn a distribution on budgets to indicate the probability of arriving destination in different time budget. This step will obviously reduce the redundant calculations and improve the efficiency to route. What's more, our method can generalize to new destinations by using a Universal Approximate Function (UAF) to approximate the distributions. So we can perform a quick response to new destinations and don't have to run our algorithm. With this step, calculating or storing the routing policies of the whole o-d pairs in a road map becomes unnecessary.

The rest of the paper is organized as follows. Section 2 introduces the necessary background to present our approach and state formally our problem. We present the details of our algorithm for SOTA problem in Section 3. In Section 4, we evaluate our algorithm on two experiments. Section 5 discusses the related work and Section 6 concludes.

Preliminaries

Markov Decision Process (MDP)

A Markov Decision Process (MDP) can be represented by the tuple $(S, A, \mathbb{T}, r, \gamma)$ with state space S and action space A, which we assume are both finite, transition probability $\mathbb{T}(s', r|s, a)$, reward r and discounted factor $\gamma \in (0, 1)$. The goal of reinforcement learning is to find a policy π to maximize the accumulated reward $R = \sum_k \gamma^k r_k$. The value function of a state s under policy π, denoted $V_\pi(s)$, is the expected return when starting in s and following π thereafter. It's defined as $V_\pi(s) = \mathbb{E}_\pi[\sum_{k=0}^{\infty} \gamma^k r_k | s_k = s], \forall s \in S$. For a road network $M = (V, E)$, V can be seen as state space and E action space.

Stochastic Shortest Path Problem with Risk Minimization

A Stochastic Shortest Path (SSP) problem is considered as a special case of undiscounted MDP where the objective is to find a optimal policy that reaches a goal state with minimum cost. SSP can be represented as the tuple $(S, A, G, \mathbb{T}, r, \gamma)$ where $S, A, \mathbb{T}, \gamma$ are defined same as in MDP, $G \in S$ is the set of goal states, reward r in MDP can be seen as a cost here.

Let's consider a different problem in SSP problem: what's the probability that the total cost less than a threshold t? That means we are going to calculate $P(\sum_k r_k < t)$, and this is the purpose of risk minimization. Taking risk minimization into account is a way to enable us to find a more reliable policy in SSP problem. (Wu and Lin 1999) study minimizing risk models in which they minimize a threshold probability $P_i^\pi(Z < t)$ with respect to policy π in MDP, where Z is a total reward, t is a threshold and i is an initial state. (Ohtsubo 2003) consider a minimizing risk model in a stochastic shortest path problem that its threshold probability is $P_i^\pi(Z > t)$, where Z is an undiscounted total cost. The paper proves that a unique optimal value exists and an optimal stationary policy in a budget can be acquired via value iteration.

In order to solve this kind problem, a new undiscounted SSP is introduced. Let's say, a tuple $(S_R, A, G, \mathbb{T})$, where $S_R = (S \times R)$ is the new state space by augmenting the original state space S with a budget $t \in R, R = [0, +\infty)$, A is action space, $G \in S$ is the goal set, $\mathbb{T}$ is the transition probability function. We define Π as a set of policies. The value function and optimal value function for finite horizon case is defined as:

$$V_g^\pi(i,t) = P_i^\pi(Z > t)$$
$$V_g^\star(i,t) = \inf_{\pi \in \Pi} V_g^\pi(i,t)$$

where Z is the accumulated cost, $g \in G$ is a destination. The reward function here becomes:

$$r_t(s,a,g) = \begin{cases} 1, & \mathbb{T}((g,\cdot)|(s,\cdot),a) = 1.0 \\ 0, & \text{otherwise} \end{cases}$$

A policy π is said to be optimal if $V_g^\star(i,t) = V_g^\pi(i,t)$ for every $(i,t) \in S_R$. Their corresponded Bellman equation and Bellman optimality equation are:

$$V_g^\pi(i,t) = \sum_a \pi(a|(i,t)) \sum_{j,\omega} \mathbb{T}((j,t-\omega)|(i,t),a) V_g^\pi(j,t-\omega)$$
$$V_g^\star(i,t) = \min_a \sum_{j,\omega} \mathbb{T}((j,t-\omega)|(i,t),a) V_g^\star(j,t-\omega)$$
$$\forall i \neq g, t \geq 0 \tag{1}$$

where j can be seen as a successor state of i in SSP. The optimal policy is get in this way:

$$\pi^\star(i,t) = \arg\max_a \sum_{j,\omega} \mathbb{T}((j,t-\omega)|(i,t),a) V_g^\star(j,t-\omega) \tag{2}$$

Stochastic On-Time Arrival Problem

In traffic routing, Stochastic On-Time Arrival (SOTA) aims at finding out the optimal policy by maximizing the probability of arriving destination from origin in a time budget. That means, we need to find a policy π to maximize the arrival probability. It can be formulated as:

$$\max_{\pi \in \Pi} P_i^\pi(Z < t) \tag{3}$$

where i is initial state, or origin node, t is the budget and Z is undiscounted total cost accumulated from origin to destination. Obviously, this kind problem can be put into the framework of SSP with risk minimization as shown above.

For a road network $M = (V, E)$ where V is node set and E is edge set, we define a new SSP with a tuple $(S_R, A, G, \mathbb{T})$, where $S_R = (V \times R)$ is the augmented state space, $A = E$ is action space, G is destination set, the transition function $\mathbb{T}(s'|s,a), \forall s, s' \in S_R, a \in A$. We set value function and optimal value function as:

$$V_g^\pi(i,t) = P_i^\pi(Z < t)$$
$$V_g^\star(i,t) = \sup_{\pi \in \Pi} V_g^\pi(i,t)$$

The value function has these properties: $V_g^\pi(g,t) = 1, \forall g \in G, V_g^\pi(i, t \to +\infty) = 1$, and $V_g^\pi(i, t < 0) = 0$.

We assume each edge has a traversal time-cost distribution denoted as $p_{ij}, ij \in E$. We notice the fact that if we know current node $i \in V$ and a edge (action) $e = ij$, the next node is determined to be j. So, for $(i,t) \in S_R$ and $a = ij \in E$, the transition function $\mathbb{T}$ can be simplified into this form:

$$\sum_{j,\omega} \mathbb{T}((j,t-\omega)|(i,t),a) = \sum_{\omega} \mathbb{T}((j,t-\omega)|(i,t),a)$$
$$\mathbb{T}((j,t-\omega)|(i,t),a) = p_{ij}(\omega), \forall t-\omega \geq 0$$

With this knowledge, the Bellman equation and Bellman optimality equation of SOTA problem can be reformulated as follows:

$$V_g^\pi(i,t) = \sum_j \pi(j|(i,t)) \sum_\omega p_{ij}(\omega) V_g^\pi(j,t-\omega)$$
$$V_g^\star(i,t) = \max_j \sum_\omega p_{ij}(\omega) V_g^\star(j,t-\omega) \tag{4}$$
$$\forall i \neq g, t \geq 0$$

where we simplify action $a = ij$ as j for convenience. And, of course, the optimal policy is reformulated as:

$$\pi^\star(i,t) = \arg\max_j \sum_\omega p_{ij}(\omega) V_g^\star(j, t-\omega) \tag{5}$$

Basically, the above is what (Fan, Kalaba, and Moore 2005) has done.

The most popular solution of this equation is based on **Value Iteration** (**VI**) (or **Successive Approximation** (**SA**) in some other literature). SA algorithm in continuous form is:

$$\begin{aligned} V_g^k(i,t) &= \max_{j\in N(i)} \int_0^t p_{ij}(\omega) V_g^{k-1}(j,t-\omega)d\omega, i \neq g, t \geq 0 \\ \pi^k(i,t) &= \arg\max_{j\in N(i)} \int_0^t p_{ij}(\omega) V_g^{k-1}(j,t-\omega)d\omega, i \neq g, t \geq 0 \end{aligned} \tag{6}$$

where $N(i)$ is the successors of node i. With adequate iterations of k, the algorithm will converge to the unique optimal value, in spite of VI's slow convergence speed.

$V_g(i,t)$ represent the probability of arriving on-time from node i to destination g in budget t with two properties:

- $\lim_{t\to 0} V_g(i,t) = 0, \lim_{t\to+\infty} V_g(i,t) = 1$
- $V_g(i,t)$ is a monotonically no-decreasing function according to t. (See (Ohtsubo 2003) for more details)

Obviously, $V_g(i,\cdot)$ works as a c.d.f at here. The budget t must be fixed before training. In our method, we don't need to calculate $V_g(i,\cdot)$ for every budget. We will directly calculate this $V_g(i,\cdot)$ in once time.

Method Description

Discretization Scheme of Budgets

We need to make an assumption that, without loss of generality, G is a connected graph with no isolated nodes in it, and starting from any node, we can reach any other node in finite time T. We see T as the max budget in our problem setting. In SOTA, one basic question is how to represent the budget slot. Basically, the general solution is to discrete the range of T to a n-length vector D as:

$$D = [\delta, 2\delta, \cdots, n\delta], \quad \delta = \lfloor T/n \rfloor$$

δ is the smallest time budget. With this D, we can try to learn a discrete probability distribution on it with regards to different o-d pairs. Another solution is to continuous probability distribution such as Gaussian distribution. While, we will consider it in future work.

Reformulation of SOTA problem

According to Equations **??**, the original SOTA problem can be reformulated in the form of two vectors' dot product:

$$\begin{aligned} V_g(i,t) &= \max_{j\in N(i)} \int_0^t p_{ij}(\omega) V_g(j,t-\omega)d\omega \\ &= \max_{j\in N(i)} \mathbb{E}_{\omega\sim p_{ij}(\cdot)}[V_g(j,t-\omega)] \\ &\approx \max_{j\in N(i)} \frac{1}{N}\sum_{k=1}^N V_g(j,t-\omega_k) \\ &= \max_{j\in N(i)} F_1 \cdot V_1 \\ & \omega_k \sim p_{ij}(\cdot), \quad i \neq g \end{aligned} \tag{7}$$

where $F_1 = [\frac{1}{N}, \frac{1}{N}, \cdots, \frac{1}{N}]$ is a frequency vector and $V_1 = [V_g(j,t-\omega_1), V_g(j,t-\omega_2), \cdots, V_g(j,t-\omega_N)]^T$ is a value vector.

We don't know the real travel time distribution p_{ij} in reality, for the travel time depends on many factors such as weather condition, traffic congestion and so on. While, based on the fact that we have drivers' trajectories data in which contain the cost time of vehicles on road map, so we can get an empirical distribution $\hat{p}_{ij}$ to replace true p_{ij}. We treat one passing on edge (i,j) with a cost ω_k as once sampling from the edge's real travel time distribution p_{ij}. The probability of travel cost ω_k on edge (i,j) can be estimated by: $\hat{p}_{ij}(\omega_k) = \frac{|\omega_k|}{\sum_k |\omega_k|}$. As the amount of samples increase, by law of large number theory, we have $\hat{p}_{ij} \to p_{ij}$.

Universal Value Function

Our purpose is to learn a distribution $V(\cdot|i,g)$ on D with regards to each i, g. We call $V(\cdot|i,g)$ as Universal Value Function (UVF), for it's an extension form to $V_g(i,t)$ and generalize the state and goal. We can easily find the UVF has the following property:

$$V(\cdot|g,g) = \mathbb{I}$$

where $\mathbb{I} = [1,1,\cdots,1], |\mathbb{I}| = n$. This means the arrival probability under any budget starting from any destination to itself is 1. With this UVF, we can continue to reformulate equation 7.

Frequency Vector We notice that if $t - \omega_k \geq 0$, it will fall into one position of D. the frequencies of $t - \omega_k$ on edge (i,j) are recorded by a frequency vector $F_{ij,t}$, t is the remaining budget. $F_{ij,t}$ is initialized as a n-length vector filled with 0. For one $\omega_k \sim \hat{p}_{ij}(\cdot)$, if the value of $t - \omega_k \geq 0$, then set the position index as $l = \lfloor (t-\omega_k)/\delta \rfloor$, so the value of $F_{ij,t}[l]$ will increase by 1. After we sample N times and the process is over, we get the auxiliary vector $F_{ij,t} = \frac{1}{N} \cdot F_{ij,t}$. For the samples $t - w_k < 0$, we still count them in N. See Algorithm 1 in details.

We use an example to illustrate this process: assuming D = [5, 10, 15, 20], with $\delta = 5, n = 4, T = 20$ and remaining time budget $t = 15$. So the initialization of the auxiliary vector on edge (i,j) will be $F_{ij,15} = [0,0,0,0]$. We make 5 times samples $[w_1 = 11, w_2 = 18, w_3 = 2, w_4 = 12, w_5 =$

Algorithm 1: Get Frequency Vector $F_{ij,t}$

```
$N = 0$
for $\omega_k \sim \hat{p}_{ij}(\cdot)$ do
    if $t - \omega_k \geq 0$ then
        $l = \lfloor (t - \omega_k)/\delta \rfloor$ ;
        $F_{ij,t}[l] += 1$ ;
    end
    $N += 1$
end
$F_{ij,t} = \frac{1}{N} F_{ij,t}$
return $F_{ij,t}$
```

7] from environment, then $t - w_k = [4, -3, 13, 11, 8]$. Removing -3, calculating $\lfloor (t - \omega_k)/\delta \rfloor$, we get a vector [0, 2, 2, 1]. So the vector would be $F_{ij,15} = [1, 1, 2, 0]$. After normalization, $F_{ij,15} = [\frac{1}{5}, \frac{1}{5}, \frac{2}{5}, 0]$. ($-3$ is removed but we still count it)

With the support of frequency vector $F_{ij,t}$, the reformulation of SOTA becomes into this form:

$$V(t|i,g) = \begin{cases} 1, & i = g, t \in D \\ \max_{j \in N(i)} \{F_{ij,t} \cdot V(\cdot|j,g)\}, & i \neq g, t \in D \end{cases} \tag{8}$$

Frequency Matrix $V(t|i,g)$ outputs the maximize probability of arriving in budget t from i to g. But at here, one trouble thing is we need to repeat this process n times to update V for every $t \in D$. While, one solution is to make V calculate all $t \in D$ in once time. What we do is to adopt a frequency matrix B_{ij} composed of $F_{ij,t}, t \in D$. That means, $B_{ij} = [F_{ij,\delta}, F_{ij,2\delta}, \cdots, F_{ij,n\delta}]$. One benefit of doing in this way is we can adopt full-batch method to accelerate the training speed. See Algorithm 2 for how to get B_{ij}.

Algorithm 2: Get Frequency Matrix B_{ij}

```
$B_{ij} = []$
for $t \in D$ do
    get $F_{ij,t}$ from Algorithm 1
    $B_{ij}$.append($F_{ij,t}$)
end
return $B_{ij}$
```

With frequency matrix B_{ij}, our final formulation of UVF will be like this:

$$V(\cdot|i,g) = \begin{cases} \mathbb{I}, & i = g \\ \max^n_{j \in N(i)} \{B_{ij} V(\cdot|j,g)\}, & i \neq g \end{cases} \tag{9}$$

The operator $\max^n_{j \in N(i)}$ allows us to select the maximized value among $\{j | j \in N(i)\}$ for every $t \in D$. In practice, B_{ij} can be calculated in advance to accelerate calculating speed. The corresponding routing policy is derived by:

$$\pi(\cdot|i,g) = \begin{cases} g & i = g \\ \arg\max^n_{j \in N(i)} \{B_{ij} V(\cdot|j,g)\}, & i \neq g \end{cases} \tag{10}$$

Universal Value Iteration

The solution to UVF is not tricky, similar to traditional value iteration, we can use a universal value iteration to acquire the optimal value of our UVF.

$$V^{k+1}(\cdot|i,g) = \max^n_{j \in N(i)} \left\{B_{ij} V^k(\cdot|j,g)\right\}, i \neq g \tag{11}$$

Calculating UVF for all $(i,g) \in S \times G$ would be a time-consuming and space-consuming process and unnecessary. In fact, we can just use part of (i,g) for UVF and then adopt an parametric model f_θ, which we call Universal Approximate Function (UAF), to approximate the UVF. We can add a softmax function at the last layer of f_θ to ensure the output is a probability distribution. Then we can use f_θ to do inference on new destinations. The loss function is defined as KL divergence between two distributions. As we know $V(\cdot)$ is a c.d.f, we must use its p.d.f $v(\cdot)$ in loss function. It is easy to get from $V(\cdot)$ by subtracting a shift result of itself: $v(\cdot) = V(\cdot) - V(\cdot)$.shift(1).

$$L(\theta) = \text{KL}(f_\theta(i,g) || v(\cdot|i,g)) \tag{12}$$

Algorithm 3: UVF and UAF learning with Universal Approximate Value Iteration

```
Input A set $G$ of destinations, road network
  $M = (V, E)$
a discrete $D$ with $\delta$ and $n$,
an initialized parametric model $f_\theta$,
iterations $K_1, K_2$,
step-size $\alpha$,
empirical distribution $\hat{p}_{ij}, \forall ij \in E$
Construct frequency matrices $\{B_{ij}\}, \forall (i,j) \in E$ via
  Algorithm 2
for $g \in G$ do
    for $k \in \{0, 1, \cdots, K_1 - 1\}$ do
        for $i \in V$ and $i \neq g$ do
            Collect $N(i)$, node $i$'s successors
            # update UVF
            V$^{k+1}(\cdot|i,g) =$
              $\max^n_{j \in N(i)} \left\{B_{ij} V^k(\cdot|j,g)\right\}$
            # update policy $\pi$
            $\pi^{k+1}(\cdot|i,g) =$
              $\arg\max^n_{j \in N(i)} \left\{B_{ij} V^k(\cdot|j,g)\right\}, i \neq g$
        end
    end
end
for $k \in \{0, 1, \cdots, K_2 - 1\}$ do
    for $g \in G$ do
        for $i \in V$ and $i \neq g$ do
            L($\theta$) = KL($f_\theta(i,g) || v(\cdot|i,g)$)
            # Update parameter $\theta$
            $\theta \leftarrow \theta + \alpha \nabla_\theta L(\theta)$
        end
    end
end
Output $V(\cdot), \pi(\cdot), f_\theta$
```

Routing Process

The routing will go on after we get the policy π. The routing process for our method is similar to SA's, except when we meet new destinations. The routing algorithm is as follows. This kind routing is called policy-based routing (Niknami and Samaranayake 2016) or adaptive routing (Flajolet, Blandin, and Jaillet 2018). It doesn't need to find a prior path before departure. It will make a decision at every intersection until arriving destination. In algorithm, we introduce a lower triangle matrix A whose all non-zero elements are 1 to multiply f_θ to make sure the result is a c.d.f.

Algorithm 4: Routing Policy

```
**Input** origin $i$, destination $g$ and time budget $T'$
empirical distribution $\hat{p}_{ij}, \forall ij \in E$
Construct frequency matrices $\{B_{ij}\}, \forall (i,j) \in E$ via
  Algorithm 2
UVFs $V(\cdot)$, routing policy $\pi$ and $f_\theta$
**for** $i \neq g$ **do**
    Collect $N(i)$, node $i$'s successors
    **if** $(i,g) \notin \pi(\cdot)$ **then**
        $V(\cdot|i,g) = Af_\theta(i,g)$ ;
        $\pi(\cdot|i,g) = \arg\max^{n}_{j \in N(i)} \{B_{ij} V(\cdot|j,g)\}$ ;
    # Choose Next Move
    $j = \pi(T'|i,g)$
    Perform routing policy, arrive next node $j$,
      observe travel cost $t$ on edge $(i,j)$
    Set $T' = T' - t$
    **if** $j == g$ *and* $T' >= 0$ **then**
        return True ;
    **else if** $T' < 0$ **then**
        return False ;
    **else**
        $i = j$ ;
    **end**
**end**
```

Experiments

We evaluate our method in this section on two experiments. We will firstly perform experiment on Grid-World and then on a real city map. Our goal in these experiments is to study the following questions:

1. Can our UVF converge to the optimal value same as SA's in SOTA problem?
2. Can our approximate function achieve generalization ability for new destinations in the environment?

We will answer two questions in following experiments.

Grid-World

We validate that our proposed method can attain the optimal value and generalize to unseen destinations. We present a 5×5 grid world consisting of a state space S with 25 states, and an action space with four action $\{left, right, up, down\}$. We suppose the traveling time on edge sampling from a Gaussian distribution: the mean is randomly chosen from $[10, 11, 12, 13, 14]$ and variance from $[1, 2, 3]$. The time budget is initialized as $T = 100$ with $n = 20$ and $\delta = 5$. The state is represented as a 25-d one-hot vector. The input of UAF f_θ is the concatenation of two one-hot vectors of current node i and goal g. Our UAF f_θ is a three layer neural networks equipped with a softmax before output.

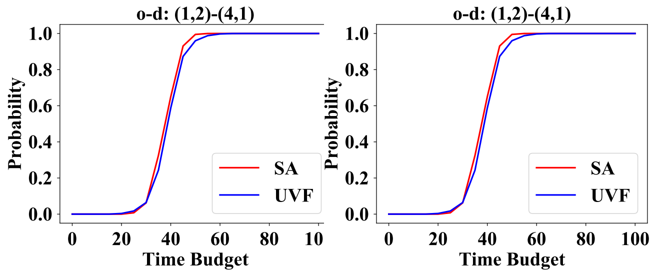

Figure 1: UVF's Result for Grid-World

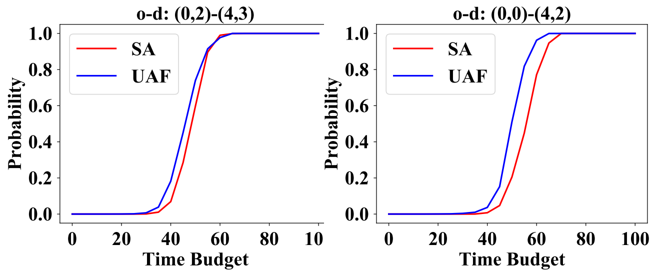

Figure 2: UAF's Result for New Destination

$$\begin{aligned} h_1 &= \tanh(\theta_1 * x + \theta_{b_1}) \\ h_2 &= \tanh(\theta_2 * h_1 + \theta_{b_2}) \\ h_3 &= \theta_3 * h_2 + \theta_{b_3} \\ output &= \text{softmax}(h_3) \end{aligned} \tag{13}$$

where $x = \text{concat}[Em(i), Em(g)]$ and $Em(\cdot)$ is one-hot vector.

For question 1, we firstly select a destination set G, then we run both SA and our method on o-d pairs $\{(i,g)|(i,g) \in S \times G\}$. Figure 1 exhibits the results of two algorithms on two o-d pairs. One origin point is $(1,2)$ and destination is $(4,1)$, the other origin point is $(0,0)$ and destination point is $(2,2)$. The results UVF's results are close to SA's (not exactly the same for randomly choosing cost distributions on edges on two algorithms). For question 2, we select two nodes $(4,3)$ and $(4,2)$, which are not int G, as new destinations. We inference their arriving probability distribution of two o-d pairs, $(0,2)-(4,3), (0,0)-(4,2)$, with approximate function f_θ trained by algorithm 3. From figure 2 we can find that the arriving probabilities of two nodes from approximate function are very close to SA's. This means approximate function can generalize to new destination.

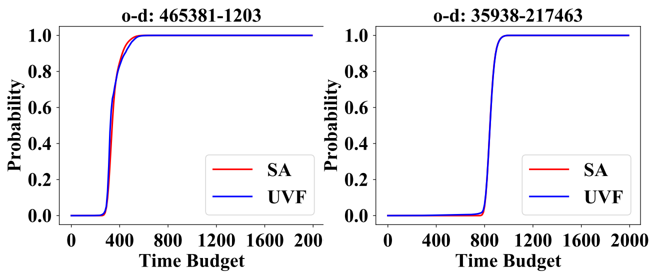

Figure 3: UVF's Result for City Map

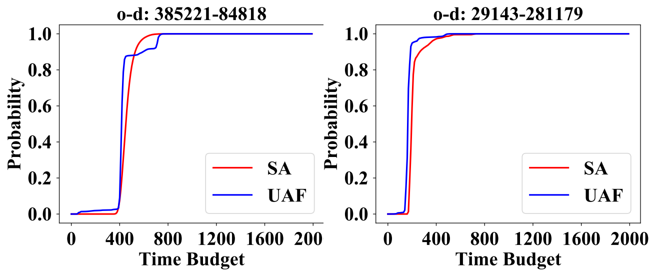

Figure 4: UAF's Result for New Destination

City Map

We evaluate our method on a real city map. The map consists of 3797 nodes and 11375 edges. We set $n = 200, \delta = 10$ for D. The structure of UAF f_θ is just more than one hidden layer compared to (13).

$$\begin{aligned} h_1 &= \tanh(\theta_1 * x + \theta_{b_1}) \\ h_2 &= \tanh(\theta_2 * h_1 + \theta_{b_2}) \\ h_3 &= \tanh(\theta_3 * h_2 + \theta_{b_3}) \\ h_4 &= \theta_4 * h_3 + \theta_{b_4} \\ output &= \text{softmax}(h_4) \end{aligned} \tag{14}$$

f_θ's input is the concatenation of node embedding of i and g which are obtained by running node2vec algorithm (Grover and Leskovec 2016). We randomly select 1000 destinations for UVF learning and UAF training. We use Pytorch to implement our code and the optimizor is Adam(Kingma and Ba 2014). The iterations $K_1 = 50$ and $K_2 = 100$. The iteration amount in SA is also set to K_1.

The results of UVF is almost the same as SA's, because the two algorithms can both converge to the optimal value. See figure 3 for reference. The inference results for new destinations are listed in 4. Table 1 lists the average cost time of two algorithms to calculate the arriving probability for one destination with regards to all other nodes. From table 1 we can see our UVF is about 30× faster than SA for one destination's calculation.

Algorithms	SA	UVF
cost time (s)	1808	91

Table 1: Cost Time for One Destination

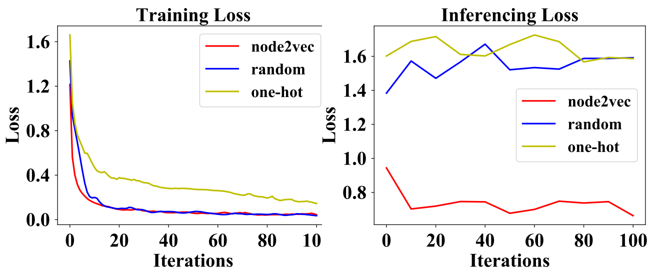

Figure 5: Train Loss and Test Loss for Three Embeddings

Representation of Node Will the representation of node affect the generalization result? The node embedding used in experiment comes from node2vec (Grover and Leskovec 2016) in which the embedding is majorly influenced by its neighbors. If the nodes are closer to each other in geometry, the cosine distances of embeddings from node2vec are smaller than others. The adjacent destinations should have similar arriving probability for other nodes. We believe this property contributes to generalization on new destinations for our UAF. We compare three kinds node representation: 1) one-hot vector; 2) node embedding with randomly initialized; 3) node embedding from node2vec. 5 shows the KL loss of training and inference changing by iteration. We can see the training loss of node2vec and random embedding's are reduced to same level but nodes2vec's is fastest than random embedding at start. The one-hot vector's convergence speed is much slower than other two embeddings. While, for the inference task, the random embedding and one-hot vector both fail here. This is because these two embeddings don't contain any information on geometry. The nodes with close geometry distance should have similar arrival probability distributions.

Related work

(Ohtsubo 2003) firstly analyzes a more general problem and proves that this kind question has an optimal solution which could be derived by value iteration algorithm. Based on this result, (Fan, Kalaba, and Moore 2005) adopts successive approximation algorithm to get access the optimal solution of SOTA problem. (Nie and Fan 2006) uses a discrete SOTA algorithm running in an optimal polynomial time and thus improves the computational efficiency significantly. (Hoy and Nikolova 2015) shows that the optimal routing policy can be approximated arbitrarily well in polynomial time under arbitrary monotone utility functions, so they present a general purpose technique based on adaptive discretization that works for any monotone utility function. (Samaranayake, Blandin, and Bayen 2012) present an efficient algorithm for finding an optimal routing policy with a well bounded computational complexity, improving on an existing solution that takes an unbounded number of iterations to converge to the optimal solution. (Yang et al. 2018) propose PACE method by fusing path's information in routing.

(Nikolova et al. 2006) propose quasi-convex maximization method to resolve this kind question by assuming each edge has independent normally distributed travel time. (Nikolova 2010) present a framework for risk-averse stochastic combinatorial optimization, which is built on non-convex optimization, to resolve fully-polynomial approximation schemes including the stochastic on-time arrival problem. (Cao et al. 2014) propose a data-driven approach by transferring the problem into a mix integer linear programming problem. (Cao et al. 2017) adopt Q-learning to estimate arriving probability and learn routing policy, by using trajectories sampled from K-shortest path tree between origin and destination.

The universal value function method has been seen in a series of work. (Schaul et al. 2015) propose a Universal Value Function Approximators (UVFA) method in reinforcement learning. This method factorized the state value into state features and goal features with matrix decomposition to achieve generalization ability on new tasks. (Ma, Wen, and Bengio 2018; Borsa et al. 2018) adopt universal successor features to do transfer learning. (Bellemare, Dabney, and Munos 2017) propose a distributional prospective to do reinforcement learning. The value function of it is defined as a distribution on accumulated rewards. Our method looks like a combination of two ideas of UVFA and distributional reinforcement learning, except we assume our environment is known and perform a universal value iteration to get the optimal value.

Conclusion

In this paper, we present a Universal Value Function (UVF) method and Universal to tackle the Stochastic On-Time Arrival (SOTA) problem. Our method doesn't depend on a prior budget, instead we learn a budget distribution on nodes. Besides, we adopt a parametric model, which we call Universal Approximate Function (UAF), to achieve the generalization to new destinations, that enables us to perform a quick response. In future, we will consider the more powerful neural networks, such as graph neural networks or graph convolutional networks, to provide more accurate generalization inference for new destinations.

References

Andonov, G.; and Yang, B. 2018. Stochastic shortest path finding in path-centric uncertain road networks. In *2018 19th IEEE International Conference on Mobile Data Management (MDM)*, 40–45. IEEE.

Bellemare, M. G.; Dabney, W.; and Munos, R. 2017. A distributional perspective on reinforcement learning. *arXiv preprint arXiv:1707.06887* .

Borsa, D.; Barreto, A.; Quan, J.; Mankowitz, D.; Munos, R.; van Hasselt, H.; Silver, D.; and Schaul, T. 2018. Universal successor features approximators. *arXiv preprint arXiv:1812.07626* .

Cao, Z.; Guo, H.; Zhang, J.; Niyato, D.; and Fastenrath, U. 2014. A data-driven method for stochastic shortest path problem. In *17th International IEEE Conference on Intelligent Transportation Systems (ITSC)*, 1045–1052. IEEE.

Cao, Z.; Guo, H.; Zhang, J.; Oliehoek, F.; Fastenrath, U.; et al. 2017. Maximizing the probability of arriving on time: A practical q-learning method. In *Thirty-First AAAI Conference on Artificial Intelligence*, 4481–4487.

Fan, Y.; Kalaba, R.; and Moore, J. 2005. Arriving on time. *Journal of Optimization Theory and Applications* 127(3): 497–513.

Flajolet, A.; Blandin, S.; and Jaillet, P. 2018. Robust adaptive routing under uncertainty. *Operations Research* 66(1): 210–229.

Grover, A.; and Leskovec, J. 2016. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, 855–864.

Hoy, D.; and Nikolova, E. 2015. Approximately optimal risk-averse routing policies via adaptive discretization. In *Twenty-Ninth AAAI Conference on Artificial Intelligence*.

Kingma, D. P.; and Ba, J. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* .

Ma, C.; Wen, J.; and Bengio, Y. 2018. Universal successor representations for transfer reinforcement learning. *arXiv preprint arXiv:1804.03758* .

Nie, Y.; and Fan, Y. 2006. Arriving-on-time problem: discrete algorithm that ensures convergence. *Transportation Research Record* 1964(1): 193–200.

Niknami, M.; and Samaranayake, S. 2016. Tractable pathfinding for the stochastic on-time arrival problem. In *International Symposium on Experimental Algorithms*, 231–245. Springer.

Nikolova, E. 2010. Approximation algorithms for reliable stochastic combinatorial optimization. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*, 338–351. Springer.

Nikolova, E.; Kelner, J. A.; Brand, M.; and Mitzenmacher, M. 2006. Stochastic shortest paths via quasi-convex maximization. In *European Symposium on Algorithms*, 552–563. Springer.

Ohtsubo, Y. 2003. Minimizing risk models in stochastic shortest path problems. *Mathematical Methods of Operations Research* 57(1): 79–88.

Samaranayake, S.; Blandin, S.; and Bayen, A. 2012. A tractable class of algorithms for reliable routing in stochastic networks. *Transportation Research Part C: Emerging Technologies* 20(1): 199–217.

Schaul, T.; Horgan, D.; Gregor, K.; and Silver, D. 2015. Universal value function approximators. In *International conference on machine learning*, 1312–1320.

Wu, C.; and Lin, Y. 1999. Minimizing risk models in Markov decision processes with policies depending on target values. *Journal of mathematical analysis and applications* 231(1): 47–67.

Yang, B.; Dai, J.; Guo, C.; Jensen, C. S.; and Hu, J. 2018. PACE: a PAth-CEntric paradigm for stochastic path finding. *The VLDB Journal* 27(2): 153–178.