1-2024

SEGAC: Sample Efficient Generalized Actor Critic for the Stochastic On-Time Arrival Problem

Honglian GUO

Zhi HE

Wenda SHENG

Zhiguang CAO
Singapore Management University, zgcao@smu.edu.sg

Yingjie ZHOU

See next page for additional authors

SEGAC: Sample Efficient Generalized Actor Critic for the Stochastic On-Time Arrival Problem

Hongliang Guo[1], Zhi He[2], Wenda Sheng[2], Zhiguang Cao[3], Yingjie Zhou[1] and Weinan Gao[4]

***Abstract*—This paper studies the stochastic on-time arrival (SOTA) problem in transportation networks and introduces a novel reinforcement learning-based algorithm, namely sample efficient generalized actor critic (SEGAC). Different from almost all canonical SOTA solutions, which are usually computationally expensive and lack generalizability to unforeseen destination nodes, SEGAC offers the following appealing characteristics. SEGAC updates the ego vehicle's navigation policy in a sample efficient manner, reduces the variance of both value network and policy network during training, and is automatically adaptive to new destinations. Furthermore, the pre-trained SEGAC policy network enables its real-time decision-making ability within seconds, outperforming state-of-the-art SOTA algorithms in simulations across various transportation networks. We also successfully deploy SEGAC to two real metropolitan transportation networks, namely Chengdu and Beijing, using real traffic data, with satisfying results.**

***Index Terms*—Generalized Actor Critic, Stochastic On-Time Arrival (SOTA), Sample Efficiency, Variance Reduction.**

I. Introduction

NAVIGATING reliably in stochastic transportation networks is of crucial importance for ensuring a satisfactory travelling experience and has thus become a prominent research topic in the field of intelligent transportation system (ITS) over the past two decades [1]. While the term 'reliability' has different definitions in the reliable navigation domain, and a brief literature review is presented in Section II, this paper focuses on the stochastic on-time arrival (SOTA) problem, whose objective is to find an apriori path or a dynamic routing policy with the maximal on-time arrival probability.

The SOTA problem finds extensive applications in various time-critical navigation scenarios, such as emergency medical services, timely catch up of air flights and/or scheduled appointments [2]. Mainstream SOTA solutions either use dynamic programming (DP)-based methods to recursively estimate the optimal policy's value function at a given node or employ mathematical programming (MP)-based methods which formulate SOTA as a mathematical optimization problem, and resort to off-the-shelf optimization solvers, *e.g.*, CPLEX, Gurobi, for the optimal solution. However, these conventional methods have their limitations. For instance, DP-based methods are computationally expensive and thus are infeasible for real-time decision-making in large-scale transportation networks. On the other hand, MP-based methods, although providing globally optimal solutions, may struggle to handle the dynamic and stochastic nature of transportation networks efficiently.

Moreover, to the best of our knowledge, almost all state-of-the-art solutions to the SOTA problem are 'all-to-one', *i.e.*, offer solutions from 'all' origin nodes to one fixed destination node, or 'one-to-one', *i.e.*, offer solutions from one origin node to one specific destination node. This limitation restricts their applicability to scenarios where dynamic routing to various destination nodes is required, such as ride sharing platforms or autonomous vehicles navigating through an urban environment with different drop-off locations.

In this paper, we study the SOTA problem from reinforcement learning (RL) perspective, and propose a sample efficient generalized actor critic (SEGAC) method as an 'all-to-all' solution to the SOTA problem, *i.e.*, SEGAC's resulting policy network applies to 'all' origin nodes and 'all' destination nodes, without the need of re-training. SEGAC comprises (1) a *generalized* actor-critic network module, which is able to update the policy network for the *non-additive* SOTA objective; (2) an off-policy policy gradient correction module, which uses importance sampling to calculate the target policy's gradient with the trajectory data collected by a behavior policy; (3) a node embedding module which maps nodes to condensed real-valued vectors; and (4) a variance reduction module, which reduces the generalized actor-critic network's variance.

With the four functional modules, SEGAC makes the following contributions to the research field of SOTA: (1) SEGAC generalizes the canonical actor critic method to fit with the *non-additive* SOTA objective; (2) SEGAC is sample efficient, in that it is able to reuse the trajectory data generated by past behavior policies for current policy network's parameter update; (3) SEGAC offers an 'all-to-all' solution to the SOTA problem, *i.e.*, we do not need to re-train SEGAC for a new destination and/or origin node; (4) SEGAC offers a low-variance solution during training stages due to the variance reduction module.

The remainder of the paper is organized as follows. We present a brief literature review of reliable navigation with a focus on the SOTA problem along the aspects of navigation objectives, prevailing methodologies and travel-time assumptions in Section II, and then introduce the basic background knowledge of RL, and SOTA's problem setup from RL's perspective, in Section III. SEGAC is introduced in Section IV

[1]College of Computer Science, Sichuan University (SCU), Chengdu, China, 610064. yjzhou09@gmail.com. Corresponding author: Yingjie Zhou.

[2]School of Automation Engineering, University of Electronic Science and Technology of China (UESTC), Chengdu, China, 611731.

[3]School of Computing and Information Systems, Singapore Management University, 178902, Singapore. zhiguangcao@outlook.com

[4]State Key Laboratory of Synthetical Automation for Process Industries, Northeastern University, Shenyang, China.

This work was supported in part by the 111 Project under Grant No. B21044 and Sichuan Science and Technology Program under Grant No. 2023NSFSC1965.

and compared with state-of-the-art DP/MP-based methods, the vanilla actor critic method and two recently proposed RL baselines for the SOTA problem in a range of transportation networks in Section V. We deploy SEGAC to two metropolitan transportation networks, *i.e*., Chengdu and Beijing, with real traffic data in Section VI, and end the paper with conclusion and future work in Section VII.

II. Literature Review

In this section, we conduct a brief literature review of reliable navigation in stochastic transportation networks. The term 'reliability' in the reliable navigation domain can be defined as the navigation policy's (1) maximal stochastic on-time arrival (SOTA) probability [3], [4], (2) minimal linear combination of mean and standard deviation travel time [5], *i.e*., mean-std shortest path, or (3) minimal travel time satisfying a given confidence level(α) [6], [7], *i.e*., α-reliable shortest path. Since this paper targets the SOTA problem, we review SOTA-related methodologies and travel-time assumptions in the remainder of this section, and interested audiences are referred to [8], [9] for comprehensive reviews of other reliable navigation methodologies. Table I delivers a bird's-eye-view of the SOTA literature, and details will be presented in the following two subsections.

TABLE I
Bird's-Eye View of the SOTA Literature

Travel Time Assumptions		Methodologies	
With/Without Model	Model-based [1], [10]–[16]	DP-based [4], [12], [13], [17]–[20]	
	Model-free [21]–[25]	MP-based [21]–[23], [25], [26]	
Spatial Correlation	Spatially Independent [2], [4], [9], [11]–[13], [16], [19]		
	Spatially Correlated [1], [15], [17], [27]	RL-based [9], [24]	
Temporal Correlation	Stationary [1], [3], [4], [15], [21], [22], [26]	Distribution Induced	Gaussian [1], [15], [16]
	Time-Varying [17], [18], [27]–[30]		SLN [27]
			Lévy [10]

A. Methodologies for the SOTA Problem

The stochastic on-time arrival problem in ITS is to find an adaptive routing policy (policy-based SOTA) or an apriori path (path-based SOTA), which navigates the ego vehicle to the destination with maximal on-time arrival probability. In this paper, we categorize the SOTA methodologies into (1) DP-based methods, (2) MP-based methods, (3) RL-based methods, and (4) specific distribution induced methods.

DP-based methods for the SOTA problem typically apply the Bellman principle of optimality to formulate the mathematical model of the problem and recursively update each node's estimated maximal on-time arrival probability, see [4], [12], [13], [17]–[20] as examples. A typical example within the category is the successive approximation (SA) algorithm [4], which has been deemed as the pioneering DP-based SOTA solutions. SA describes the relationships between adjacent nodes' maximal on-time arrival probabilities through convolution integrals, and the adjacent node with the maximal estimated SOTA probability is chosen as the next executing node. While DP-based methods are straightforward to understand and capable of delivering a stable SOTA solution, they require the availability of full travel-time distribution and incur the non-polynomial convolutional computation, which prevents the real-time decision-making ability.

On the other hand, MP-based methods are usually deemed as data-driven solutions, which formulate SOTA as a mathematical optimization problem with pre-collected travel-time samples, and employ off-the-shelf optimization solvers, *e.g*., CPLEX and Gurobi, for the ultimate solution, see [21]–[23], [25], [26] as examples. Cao *et al*. formulate the path-based SOTA problem as a cardinality minimization problem, and then use L_1-norm approximation to transform the originally formulated problem into a mixed integer linear programming (MILP) problem, which is solved by the canonical optimization solver [21]. Similarly, Yang and Zhou consider the spatial-temporal travel-time dependencies, and also formulate the SOTA problem into an MILP framework. Then, they propose a one-step mixed integer programming (OSMIP) method to calculate the final solution [23]. MP-based methods for the SOTA problem are model-free/data-driven solutions, which do not need the well-calibrated travel-time distribution models. However, the formulated mathematical problem is usually an MILP, which cannot be solved within polynomial time. Hence, MP-based methods are typically computationally expensive and cannot be applied to real-time application scenarios.

Recently, there has been an emerging trend of applying reinforcement learning (RL) methods to solve the SOTA problem. Cao *et al*. formulate the SOTA problem within the Markov decision process (MDP) framework, and propose a practical Q-learning (PQL) method, which uses the multi-layer perceptron to approximate RL agent's Q values, *i.e*., the corresponding nodes' on-time arrival probabilities [24]. Guo *et al*. propose a cascaded temporal difference (CTD) method, which simultaneously estimate the mean and variance of a given routing policy [9]. With the normal distribution assumption of the travel time, CTD can be applied to solve the SOTA problem. Most recently, Guo *et al*. invent a DRL-Router, which employs *distributional* reinforcement learning to estimate the full travel-time distribution of a given routing policy, and it is potentially applicable for a range of reliable navigation objectives including SOTA [31]. Since SEGAC also belongs to the RL-based methods, we articulate the differences/advantages of SEGAC when compared to the existing RL-based solutions, *i.e*., PQL, CTD and DRL-Router: (1) SEGAC is essentially an off-policy RL method, and hence is much more sample efficient than PQL, CTD and DRL-Router, which are essentially on-policy methods; (2) SEGAC offers an 'all-to-all' solution, which is more general than other RL-based router, which can only offer 'all-to-one' solutions, and (3) both PQL and CTD impose basic travel-time model assumptions, *e.g*., PQL needs the mean and variance of the K-shortest paths to be known, and CTD assumes a Gaussian distribution of the travel time, while SEGAC is completely model-free, without any travel-time distribution assumption.

Apart from reinforcement learning, machine learning-based methods, such as neural-heuristic analysis, have also been applied to the vehicle routing domain. Neural-heuristic methods combine the power of neural networks with traditional heuris-

tics to address vehicle routing problems. These approaches leverage neural networks for learning complex spatial patterns and making predictions, while heuristics provide structured rules for guiding the decision-making process, see [32]–[36] as examples. However, to the best of our knowledge, neural-heuristic methods are mostly applied to the vehicle routing problem (VRP), which aims at determining the most efficient way to deliver goods or services to a set of customers or locations with a fleet of vehicles while minimizing costs or maximizing efficiency. However, in this paper, we target navigating a single vehicle to one fixed destination with the maximal SOTA probability in stochastic transportation networks.

Besides the three aforementioned types of SOTA solutions, there exist a few research works, which target efficient SOTA solutions for specific travel-time distributions, *e.g.*, normal distribution [16], multi-variate Gaussian distribution [1], [15], shifted log-normal (SLN) distribution [27], Lévy distribution [10]. For example, when assuming that the edges' travel time follows independent normal distributions, Lim *et al*. propose an efficient λ-optimal path finding method, which confines the optimal SOTA solution within a set of λ-optimal paths, with known upper and lower bound of λ [16]. Later, Guo *et al*. extend the results to the multi-variate Gaussian distribution use case, and prove that the SOTA problem is equivalent to a mean-std shortest path problem, which can be solved by a polynomial-complexity algorithm [1]. With the same travel-time distribution assumption, Gao *et al*. make use of the conjugate relationship between the posterior and prior multi-variate Gaussian distribution, and propose a Gaussian process *proactive* path planning (GP4) method for the policy-based SOTA problem [15].

B. Travel-Time Assumptions

The SOTA solution strategies vary with different travel-time distribution assumptions of the underlying transportation network. In this paper, we categorize the transportation network's travel-time distribution modeling along the taxonomies of (1) whether a travel-time distribution modeling is needed or not; and (2) travel-time distribution's spatial and temporal correlation characteristics.

Most of the research works in reliable navigation are offering model-based solutions, where the underlying transportation network's travel-time distribution is needed as algorithm inputs, see [1], [10]–[16]. For example, Prakash proposes a decreasing order of time (DOT) algorithm [18], which discretizes the travel-time distribution within a given time interval as histograms, and employ DP to iteratively calculate the optimal routing policy given the remaining time budget. The author also delivers a 'Pruning' algorithm in the same paper, which outputs an apriori path with the maximal SOTA probability.

On the other hand, a few research works are providing model-free/data-driven solutions to the SOTA problem, see [21]–[25] as examples. The related methodologies typically make use of the pre-collected travel-time samples instead of travel-time distribution models for the SOTA solution. For example, Cao *et al*. propose a practical Q-learning (PQL) method, which uses a neural-network represented function approximator to estimate the on-time arrival probability of a given node [24]. The pre-collected travel-time samples are used to calculate the training targets of the neural network, and in this way, PQL is deemed as a model-free/data-driven solution to the SOTA problem.

Another perspective of dichotomizing the research works in reliable navigation along the taxonomies of travel-time distribution assumptions is its spatial and temporal correlation characteristic. The simplest and most commonly used assumption is that the travel-time distribution is independent and stationary, *i.e.*, the edge's travel-time distribution is independent from each other, and does not change over time, see [2], [4], [9], [11]–[13], [16], [19] as examples.

Meanwhile, there are research works targeting spatially correlated travel-time distributions and/or time-dependent transportation networks [1], [17], [18], [27]–[29], [37]. For example, Nie and Wu study the SOTA problem where the travel-time distribution has limited spatial and temporal dependencies, and use the first-order stochastic dominance theory to gradually prune the candidate path set to the optimal one [17]. Guo *et al*. study the reliable navigation problem with multi-variate Gaussian distribution assumption, and propose a Gaussian process path planning (GP3) method, which solves the SOTA problem by transforming it to an equivalent mean-std shortest path problem [1]. Yang and Zhou investigate the time-dependent travel-time distribution use case; formulate the SOTA problem into the mixed integer linear programming (MILP) framework, and employ off-the-shelf MILP solvers, *e.g.*, CPLEX and Gurobi, for the ultimate solution [23].

Here, we wish to articulate that this paper targets the *independent* and *stationary* travel-time distribution use case, and treats the spatially correlated and time-varying travel-time distribution as one of our future work directions.

III. RL Backgrounds and Problem Formulation

Since SEGAC is essentially an off-policy reinforcement learning method for the SOTA problem, we introduce RL-related backgrounds including Markov decision process (MDP), temporal difference (TD) learning, policy gradient (PG) and actor critic (AC), and also lay down SOTA's problem formulation towards the end of the section. A list of major notations used in the paper is given in Table VII of Appendix A, where the first set of notations is transportation network related; the second set is SEGAC and/or RL related.

A. Reinforcement Learning Backgrounds

An RL agent aims at maximizing its expected discounted cumulative reward through interacting with the environment and optimizing its decision-making policy [38]. The environment's underlying dynamics is modelled as a Markov decision process (MDP), defined by the tuple ($\mathcal{S}$, $\mathcal{A}$, $\mathbb{P}[s'|s,a]$, $\mathbb{P}[r|s,a]$, γ), where $\mathcal{S}$ and $\mathcal{A}$ represent the RL agent's state space and action space, respectively. $\mathbb{P}[s'|s,a]$ and $\mathbb{P}[r|s,a]$ refer to the (unknown) state-transition probability and reward function, respectively, and γ is the discount factor. The

RL agent's objective is to find/learn an optimal policy π^*, which maximizes the expected discounted cumulative reward, *i.e.*, $\mathbb{E}[\sum_{k=0}^{\infty}\gamma^k r_k]$.

One of the canonical methods for solving the RL problem is temporal difference (TD) learning, which enables the RL agent to learn from incomplete sequences of experiences. TD error (δ) measures the discrepancy between the expected return and the estimated value, providing a learning signal for policy improvement. In TD learning, the state-value function $V(s)$ is updated iteratively with the TD error and a learning rate α:

$$V(s_t) \leftarrow V(s_t) + \alpha\delta_t, \tag{1}$$

where $\delta_t = R_{t+1}+\gamma V(s_{t+1})-V(s_t)$. TD methods, such as Q-learning [39] and SARSA [38], have been applied to solving RL problems, offering a balance between sample efficiency and bias-variance trade-off.

On the other hand, policy gradient (PG) methods focus on directly parameterizing the policy $\pi_{\boldsymbol{\theta}}(a|s)$ and optimizing it to maximize the expected return. The objective is to find the policy parameter vector $\boldsymbol{\theta}$ which maximizes the expected return, *i.e.*, $J(\boldsymbol{\theta})$. Policy gradients are estimated using the gradient of the expected return, with respect to the policy parameters:

$$\boldsymbol{\nabla} J(\boldsymbol{\theta}) = \mathbb{E}\left[\sum_{t=0}^{T} \boldsymbol{\nabla}_{\boldsymbol{\theta}} \pi_{\boldsymbol{\theta}}(a_t|s_t) Q(s_t, a_t)\right], \tag{2}$$

where $Q(s_t, a_t)$ is the state-action value which can be estimated with temporal difference (TD) learning, or Monte Carlo (MC) sampling.

Considering the high variance of PG methods during training stages, researchers propose the actor critic (AC) method [40], which serves as a synergy between value-based RL methods, *e.g.*, SARSA, and policy-based RL methods, *e.g.*, PG. AC methods combine the advantages of policy gradient and value function estimation. An actor network parameterizes the policy, while a critic network approximates the state-value function. The critic evaluates the value of state-action pairs, guiding the actor's policy updates through TD error. The actor's policy parameters are updated based on the policy gradient, while the critic's parameters are updated to minimize the TD error. This combination enhances learning stability and convergence in RL tasks.

B. SOTA Problem Formulation

The SOTA reliable navigation problem is to find an apriori path or an adaptive routing policy, depending on user's preference, which navigates the ego vehicle from an origin node o to a destination node d, with the maximal on-time arrival probability before a given time budget T. The underlying stochastic transportation network is represented by a directed and connected graph $\mathcal{G}(\mathscr{N}, \mathscr{E})$, where $\mathscr{N}$ ($|\mathscr{N}| = n$) refers to the set of nodes and $\mathscr{E}$ ($|\mathscr{E}| = m$) refers to the set of edges. Denote c_{ij} as edge ij's travel time, which is a random variable (RV), and $\mathscr{P}_{od}$ as the complete set of feasible paths connecting o to d.

The path-based SOTA problem can be formulated as:

$$\begin{aligned} \underset{\boldsymbol{x}}{\text{maximize}} \quad & \mathbb{P}(c_{\boldsymbol{x}} \leq T) \\ \text{subject to} \quad & \boldsymbol{x} \in \mathscr{P}_{od}, \end{aligned} \tag{3}$$

where T is the user-specified path planning budget, $c_{\boldsymbol{x}}$ is Path x's total travel time. While for the policy-based SOTA problem, the output is an adaptive routing policy π, which takes the ego vehicle's current residing node i, destination node d and remaining budget T_i as inputs, and outputs the next executing edge ij, *i.e.*, $ij = \pi(i, d, T_i)$. Denote $t^{\pi}(o, d)$ as the total travel time (an RV) when the ego vehicle starts at o; follows policy π for en-route decision making and arrives at d, and the problem is to:

$$\begin{aligned} \underset{\pi}{\text{maximize}} \quad & \mathbb{P}(t^{\pi}(o, d) \leq T) \\ \text{subject to} \quad & ij = \pi(i, d, T_i), \\ & T_j = T_i - t_{ij}, \end{aligned} \tag{4}$$

where t_{ij} is the instantiated travel time of edge ij, T_i and T_j are the remaining budgets for node i and node j, respectively. The term 'adaptive' refers to the fact that the ego vehicle is able to adapt its decision-making policy en-route with the ever-changing remaining budget T_i.

Examining Eq. (4), we can see that if the routing policy π does not take the real-time remaining budget T_i as one of the inputs, and outputs an edge ij based on the pre-departure budget T, it becomes a *non-adaptive* routing policy, which is essentially the solution to the path-based SOTA problem defined in Eq. (3). Therefore, it is easy to switch between policy-based solution and path-based solution with a policy network representation of π. With real-time remaining budget T_i as the input, π offers the *policy*-based solution, on the other hand, with only the pre-departure budget T as input, π offers the *path*-based solution to the SOTA problem.

IV. Sample Efficient Generalized Actor Critic

This section presents the core RL methodology for SOTA, namely sample efficient generalized actor critic (SEGAC). As the SOTA objective is essentially *non-additive*, it makes the canonical policy gradient theorem invalid; furthermore, the canonical TD method, *e.g.*, SARSA, is also inapplicable in that the Bellman optimality condition is violated by the non-additive nature of the SOTA objective. In the following subsections, we will present (1) the generalized policy gradient theorem, which applies to the non-additive SOTA problem; (2) an on-policy extended value function approximation algorithm, which extends the canonical TD learning method to the SOTA use case; (3) the integrated generalized actor critic method with corresponding off-policy corrections for sample efficiency; (4) the variance reduction module for SEGAC and (5) the state space encoding mechanism and feature training process, which enables SEGAC with the 'all-to-all' solution characteristic. Note that SEGAC is able to output both path-based and policy-based SOTA solutions, we focus on the policy-based SOTA solution in the main presentation flow, and explain, in Section IV-F, how we feed the policy/actor network with a different feature set, which fosters the path-based SOTA solution. We end the section with the algorithm's

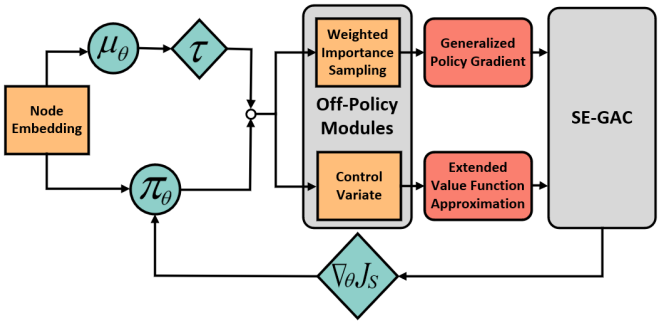


Fig. 1. The SEGAC framework: (1) the node embedding module maps nodes into condensed d-dimensional real-valued vectors, which facilitate SEGAC's generalization ability across different destination nodes; (2) $\mu_{\boldsymbol{\theta}}$ is the behavior policy generating the trajectory sequences (τ), and $\pi_{\boldsymbol{\theta}}$ is the evaluation policy whose policy parameters ($\boldsymbol{\theta}$) are to be updated; (3) the Generalized Policy Gradient (GPG) module estimates the gradients of the behavior policy $\mu_{\boldsymbol{\theta}}$, and when combined with the weighted importance sampling module, it is transformed to OP-GPG, which estimates the policy gradient of $\pi_{\boldsymbol{\theta}}$; (4) the E-VFA module estimates the TD-error of $\pi_{\boldsymbol{\theta}}$, and the control variate module decreases the variance of E-VFA; (5) GPG and E-VFA are fed into SEGAC, which outputs the final policy gradient update direction of $\pi_{\boldsymbol{\theta}}$.

computational complexity analysis. Fig. 1 displays SEGAC's overall framework, whose composing modules are presented in the following subsections.

A. Generalized Policy Gradient for SOTA

In this subsection, we present the generalized policy gradient theorem which updates the parameters of the policy network[1]. We use the one-hot encoder to represent the ego vehicle's current residing node (i) as well as the destination node (d), and then concatenate them with the remaining budget T_i to form the state representation, *e.g.*, $s_k = (\psi(i); \psi(d); T_i)$, where $\psi(j)$ refers to the mapping of j to the one-hot encoder. The action (a) corresponds to which edge to execute. The policy network $\pi_{\boldsymbol{\theta}}$ is a multi-layer perceptron (MLP), which maps the state (s_k) to probabilistic action selections, *i.e.*, $\mathbb{P}[a_k = a] = \pi(a|s_k; \boldsymbol{\theta})$.

Defining $J_s(\pi_{\boldsymbol{\theta}}) = \mathbb{P}[t^\pi(o,d) \leq T]$ and $\tau^{(j)} = \left(s_0^{(j)}, a_0^{(j)}, r_1^{(j)}, s_1^{(j)}, a_1^{(j)}, \ldots, s_{H_j-1}^{(j)}, a_{H_j-1}^{(j)}, r_{H_j}^{(j)}, s_{H_j}^{(j)}\right)$ as the j^{th} trajectory sample generated by π, where $R(\tau^{(j)}) = \sum_{k=0}^{H_j-1} r_{k+1}^{(j)}$ is the instantiated total travel time for $\tau^{(j)}$, we lay down the generalized policy gradient theorem whose proof process is presented in Appendix B.

Theorem 1 (The Generalized Policy Gradient Theorem)**.** *The derivative of $J_s(\pi_{\boldsymbol{\theta}})$ with respect to $\boldsymbol{\theta}$ is the expectation of the product of the $\pi_{\boldsymbol{\theta}}$-induced trajectory's SOTA probability and the gradient of the* log *of policy $\pi_{\boldsymbol{\theta}}$, i.e., $\nabla_{\boldsymbol{\theta}} J_s(\pi_{\boldsymbol{\theta}}) = \mathbb{E}_{\tau\sim\pi}[\mathbb{P}[R(\tau) \leq T] \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(\tau)]$, where τ is $\pi_{\boldsymbol{\theta}}$-induced trajectory, $R(\tau)$ is τ's total travel time (an RV), and $\pi_{\boldsymbol{\theta}}(\tau)$ refers to the probability of generating τ by $\pi_{\boldsymbol{\theta}}$.*

Theorem 1 lays a solid theoretical foundation of applying PG-related techniques to solve the SOTA problem. In practice, suppose that we have M trajectory samples generated by $\pi_{\boldsymbol{\theta}}$, *i.e.*, $\forall j \in \{1, 2, \ldots, M\}$, we have $\tau^{(j)} = \left(s_0^{(j)}, a_0^{(j)}, r_1^{(j)}, s_1^{(j)}, a_1^{(j)}, \ldots, s_{H_j-1}^{(j)}, a_{H_j-1}^{(j)}, r_{H_j}^{(j)}, s_{H_j}^{(j)}\right)$, we can approximate the expectation term with Monte Carlo sampling, and get:

$$
\begin{aligned}
&\nabla_{\boldsymbol{\theta}} J_s(\pi_{\boldsymbol{\theta}}) \\
=&\mathbb{E}_{\tau\sim\pi}\left[\mathbb{P}[R(\tau) \leq T] \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(\tau)\right] \\
\approx&\frac{1}{M}\sum_{j=1}^{M} \mathbb{1}\{R(\tau^{(j)}) \leq T\} \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(\tau^{(j)}) \\
=&\frac{1}{M}\sum_{j=1}^{M} \Big(\mathbb{1}\{R(\tau^{(j)}) \leq T\} \times \nabla_{\boldsymbol{\theta}} \sum_{k=0}^{H_j-1} \big(\log \pi(a_k^{(j)}|s_k^{(j)}; \boldsymbol{\theta}) \\
&+ \log \mathbb{P}[r_{k+1}^{(j)}, s_{k+1}^{(j)}|s_k^{(j)}, a_k^{(j)}]\big)\Big) \\
=&\frac{1}{M}\sum_{j=1}^{M} \big(\mathbb{1}\{R(\tau^{(j)}) \leq T\} \times \sum_{k=0}^{H_j-1} \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(a_k^{(j)}|s_k^{(j)})\big), \quad (5)
\end{aligned}
$$

where $\mathbb{1}\{\text{statement}\}$ is the indicator function, which returns 1 if the statement is true, and returns 0 if the statement is false. Note that in Eq. (5), the derivation of the last line makes use of the fact that the environmental dynamics (transition probability and reward function) does not depend on $\boldsymbol{\theta}$, *i.e.*, $\forall\, 0 \leq k \leq H_j - 1$, and thus we have $\nabla_{\boldsymbol{\theta}} \log \mathbb{P}[r_{k+1}^{(j)}, s_{k+1}^{(j)}|s_k^{(j)}, a_k^{(j)}] = 0$.

With the generalized policy gradient theorem and Eq. (5), we present the on-policy generalized policy gradient (OP-GPG) algorithm for the SOTA problem, as depicted in Algorithm 1.

Algorithm 1: On-Policy Generalized Policy Gradient

Input: (1) origin o, destination d; (2) time budget: T; (3) learning rate: α; (4) max. iteration: $T_{\max}$; (5) num. of episodes per iteration: M;
Output: Policy network $\pi_{\boldsymbol{\theta}}$;
Init: (1) Randomly initialize the policy network $\pi_{\boldsymbol{\theta}}$; (2) counter $\leftarrow 0$;

```
while counter ≤ T_max − 1 do
    Run π_θ and collect M trajectories, i.e., τ^(j);
    Calculate ∇_θ J_s(π_θ) according to Eq. (5);
    Update the policy network's parameter as:
      θ = θ + α ∇_θ J_s(π_θ);
    counter ← counter + 1;
Final.
```

B. Extended Value Function Approximation (E-VFA)

The last subsection presents the OP-GPG algorithm for the SOTA problem. However, examining Eq. (5), we can see that the 'rewarding' to a given policy $\pi_{\boldsymbol{\theta}}$, is too sparse, *i.e.*, $\mathbb{1}\{R(\tau^{(j)}) \leq T\} \in \{0, 1\}$. For example, if $\mathbb{1}\{R(\tau^{(j)}) \leq T\} = 0$, the corresponding policy's parameter is not updated at all. While in practice, the policy fails to lead the ego vehicle to arrive on time, and it should be penalized. A naive implementation is to subtract all the policy's return by a constant factor (0.5), *i.e.*, $\mathbb{1}\{R(\tau^{(j)}) \leq T\} - 0.5$, and

[1]Note that conforming to RL conventions, in the PG context, the ego vehicle's decision-making mechanism is called the 'policy network'. While in the AC context, the same mechanism is referred to as the 'actor network'. In essence, the policy network is the same as the actor network.

use the altered return as the computation basis for policy update. However, subtracting the policy's return by a constant factor (0.5) for all the SOTA problem configurations, *i.e.*, all origin-destination (OD) pairs (o and d) and all budgets (T), is inefficient and may prolong OP-GPG's convergence time. This paper proposes an extended value function approximation (E-VFA) method to estimate the evaluation policy's on-time arrival probability when given OD and T, and use the E-VFA estimates as the subtracting baseline.

Define $b^{\pi}_{\boldsymbol{\phi}}(s)$ as the E-VFA estimated on-time arrival probability for policy π at state s, *e.g.*, $b^{\pi}_{\boldsymbol{\phi}}(s) = \mathbb{P}[t^{\pi}(o,d) \leq T]$. We adopt the same state representation scheme for E-VFA as that of the policy network $\pi_{\boldsymbol{\theta}}$, *e.g.*, $s = (\psi(o), \psi(d), T)^{\top}$ if the ego vehicle is at node o, with time budget T and destination node d. Note that $b^{\pi}_{\boldsymbol{\phi}}(s)$ is also an MLP parameterized by $\boldsymbol{\phi}$. Next, we present a corollary displaying the extended Bellman equation of $b^{\pi}_{\boldsymbol{\phi}}(s)$, and use it as the basis for the E-VFA algorithm.

Corollary 1. *The extended Bellman equation for the SOTA objective is expressed as:*

$$b^{\pi}_{\boldsymbol{\phi}}(s) = \sum_{a} \big(\pi_{\boldsymbol{\theta}}(a|s) \sum_{s'} \mathbb{P}[s'|s,a] b^{\pi}_{\boldsymbol{\phi}}(s')\big). \tag{6}$$

In Corollary 1, the probability transition function, $\mathbb{P}[s'|s,a]$, includes both the ego vehicle's residing node transition and the remaining budget transition, *e.g.*, from $T(s)$ to $T(s')$, after executing action a. The proof process of Corollary 1 is straightforward and omitted in the main manuscript. Interested audiences are referred to the publicly available code repository (see Section V) for details. Note that the extended Bellman equation as expressed in Eq. (6) does not contain the reward function, *i.e.*, $\mathbb{P}[r|s,a]$, explicitly, in that it is already implicitly expressed in the remaining budget ($T(s')$) as a part of the next time-step state (s').

Corollary 1 provides the basis for estimating a given policy's on-time arrival probability in a bootstrapped way with temporal difference (TD) learning. Next, we express the E-VFA algorithm with one-step TD method in Algorithm 2. Note that one may also choose to use multi-step TD, TD(λ) or Monte Carlo (MC) to instantiate the on-policy E-VFA algorithm. We omit the corresponding details in the main manuscript for presentation sanity, and put the respective pseudo codes in the code repository. In Algorithm 2, for Line 8, when calculating a given state (s_k)'s on-time arrival probability under π, we first extract the vehicle's current residing node i, destination node d, and remaining budget T_i. If $T_i < 0$, we set $b^{\pi}_{\boldsymbol{\phi}^-}(s_k) = 0$; else if $i = d$, we set $b^{\pi}_{\boldsymbol{\phi}^-}(s_k) = 1$, otherwise, we feed the corresponding feature vector into the MLP, which calculates the value for $b^{\pi}_{\boldsymbol{\phi}^-}(s_k)$.

Algorithm 2: Extended Value Function Approximation

Input: (1) Policy network $\pi_{\boldsymbol{\theta}}$; (2) mini-batch size: K; (3) max. episodes: $T_{\max}$; (4) max. training iteration: T_{iter}

Output: E-VFA estimated baseline $b^{\pi}_{\boldsymbol{\phi}}$;

Init: (1) Initialize replay memory D; (2) Randomly initialize $\boldsymbol{\phi}$ for $b^{\pi}_{\boldsymbol{\phi}}$; (3) initialize the target network's parameter $\boldsymbol{\phi}^- = \boldsymbol{\phi}$; (4) $j \leftarrow 1$;

1 **while** $j \leq T_{\max}$ **do**
2 Randomly initialize origin o, destination d and budget T;
3 Run $\pi_{\boldsymbol{\theta}}$ and collect the trajectory, *i.e.*, $\tau^{(j)} = \big(s_0^{(j)}, a_0^{(j)}, r_1^{(j)}, s_1^{(j)}, \ldots, s_{H_j-1}^{(j)}, a_{H_j-1}^{(j)}, r_{H_j}^{(j)}, s_{H_j}^{(j)}\big)$;
4 $\forall\, 0 \leq k \leq H_j - 1$, store transition $(s_k^{(j)}, a_k^{(j)}, r_{k+1}^{(j)}, s_{k+1}^{(j)})$ in D;
5 $j \leftarrow j + 1$;
6 **foreach** $j \in \{1, 2, \ldots, T_{iter}\}$ **do**
7 Sample K mini-batch transitions $(s_k, a_k, r_{k+1}, s_{k+1})$ from D;
8 Set $y_k = b^{\pi}_{\boldsymbol{\phi}^-}(s_{k+1})$;
9 Perform a stochastic gradient descent step on $\frac{1}{K}\sum_{k=1}^{K}(y_k - b^{\pi}_{\boldsymbol{\phi}}(s_k))^2$ with respect to $\boldsymbol{\phi}$;
10 Reset $\boldsymbol{\phi}^- = \boldsymbol{\phi}$;
11 Final.

C. Sample Efficient Generalized Actor Critic (SEGAC)

The last subsection introduces the E-VFA algorithm, which estimates the on-time arrival probability of the evaluation policy (π). In this subsection, we will first integrate OP-GPG and E-VFA to reach the on-policy generalized actor critic (OP-GAC) algorithm for the SOTA problem. After that, we introduce the off-policy versions of OP-GPG and E-VFA, which ultimately upgrade OP-GAC to sample efficient generalized actor critic (SEGAC).

1) On-Policy Generalized Actor Critic: With E-VFA estimated baseline, *i.e.*, $b^{\pi}_{\boldsymbol{\phi}}(s)$, we can improve the OP-GPG algorithm to the *generalized* actor critic version. In generalized actor critic, we replace the calculation of $J_s(\pi_{\boldsymbol{\theta}})$'s gradient as:

$$\nabla_{\boldsymbol{\theta}} J_s(\pi_{\boldsymbol{\theta}}) = \frac{1}{M}\sum_{j=1}^{M}\Big(\big(\mathbb{1}\{R(\tau^{(j)}) \leq T\} - b^{\pi}_{\boldsymbol{\phi}}(s)\big) \times \sum_{k=0}^{H_j-1} \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(a_k^{(j)}|s_k^{(j)})\Big), \tag{7}$$

where the corresponding notation symbols have been introduced in the last two subsections, respectively, and one may also refer to Table VII of Appendix A for brief descriptions. With Eq. (7), one can reach the on-policy generalized actor critic (OP-GAC) algorithm, as shown in Algorithm 3. The algorithm flow process is similar to that of Algorithm 1. However, within the 'while' loop, after collecting M trajectories, OP-GAC will first update the E-VFA estimated baseline function $b^{\pi}_{\boldsymbol{\phi}}$, and then use it as the subtracting baseline to calculate the gradient of $J_s(\pi_{\boldsymbol{\theta}})$.

2) Off-Policy Generalized Policy Gradient: The on-policy generalized actor critic algorithm (as depicted in Algorithm 3) works well for the SOTA problem in the reliable navigation domain. However, it is sample inefficient, in that both OP-GPG and E-VFA need the 'on-policy' trajectory data collected from the evaluation policy π, to update the respective parameters. In

Algorithm 3: On-Policy Generalized Actor Critic

Input: (1) origin o, destination d; (2) time budget: T; (3) learning rate: α; (4) max. iteration: $T_{\max}$; (5) num. of episodes per iteration: M;
Output: Policy network $\pi_{\boldsymbol{\theta}}$;
Init: (1) Randomly initialize the policy network $\pi_{\boldsymbol{\theta}}$; (2) counter $\leftarrow 0$;

while *counter* $\leq T_{\max} - 1$ **do**
  Run $\pi_{\boldsymbol{\theta}}$ and collect M trajectories, *i.e.*, $\tau^{(j)}$;
  Calculate $b^{\pi}_{\boldsymbol{\phi}}$ with E-VFA;
  Calculate $\boldsymbol{\nabla}_{\boldsymbol{\theta}} J_s(\pi_{\boldsymbol{\theta}})$ according to Eq. (7);
  Update the policy network's parameter as: $\boldsymbol{\theta} = \boldsymbol{\theta} + \alpha \boldsymbol{\nabla}_{\boldsymbol{\theta}} J_s(\pi_{\boldsymbol{\theta}})$;
  counter $\leftarrow$ counter $+ 1$;
Final.

practice, one needs a sample efficient algorithm, which is able to make use of the 'past' policy generated trajectory data to update the 'current' actor/policy network's and critic/baseline network's parameters. Conforming to the RL conventions, we name the 'past' policy as the behavior policy, denoted as $\mu_{\boldsymbol{\theta}}$[2]. In the following, we will introduce (1) the off-policy generalized policy gradient theorem, which serves as the theoretical foundation of the off-policy GPG algorithm and (2) the importance sampling correction mechanism for the off-policy E-VFA algorithm.

Theorem 2 (Off-Policy Generalized Policy Gradient Theorem)**.** *The gradient of $J_s(\pi_{\boldsymbol{\theta}})$ for behavior policy ($\mu_{\boldsymbol{\theta}}$) generated trajectory τ can be expressed as:*

$$\boldsymbol{\nabla} J_s(\pi_{\boldsymbol{\theta}}) = \mathbb{E}_{\tau\sim\mu_{\boldsymbol{\theta}}}[\mathbb{P}[R(\tau) \leq T](\boldsymbol{\nabla}_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(\tau))\rho(\tau)], \quad (8)$$

where $\rho(\tau) = \pi_{\boldsymbol{\theta}}(\tau)/\mu_{\boldsymbol{\theta}}(\tau)$ is the importance sampling ratio of π to μ with respect to trajectory τ.

The proof process of Theorem 2 is presented in Appendix C. With Theorem 2, we deliver the off-policy generalized policy gradient approximation procedure for $\pi_{\boldsymbol{\theta}}$ with M trajectory samples generated by $\mu_{\boldsymbol{\theta}}$ as follows:

$$\boldsymbol{\nabla}_{\boldsymbol{\theta}} J_s(\pi_{\boldsymbol{\theta}}) = \frac{1}{M}\sum_{j=1}^{M} \big(\mathbb{1}\{R(\tau_{\mu}^{(j)}) \leq T\} \times \sum_{k=0}^{H_j-1} \boldsymbol{\nabla}_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(a_k^{(j)}|s_k^{(j)})\rho(\tau_{\mu}^{(j)})\big), \quad (9)$$

where $\tau_{\mu}^{(j)}$ is the j^{th} trajectory data generated by $\mu_{\boldsymbol{\theta}}$, and the importance sampling ratio is expressed as: $\rho(\tau_{\mu}^{(j)}) = \prod_{k=0}^{H_j-1} \frac{\pi(a_k|s_k)}{\mu(a_k|s_k)}$. We skip the pseudo code presentation of off-policy generalized policy gradient as it is quite similar to Algorithm 1 by merely replacing the calculation procedure of $\boldsymbol{\nabla}_{\boldsymbol{\theta}} J_s(\pi_{\boldsymbol{\theta}})$ with Eq. (9).

[2]Note that both the behavior policy $\mu_{\boldsymbol{\theta}}$ and the evaluation policy $\pi_{\boldsymbol{\theta}}$, are essentially MLPs. Here, $\boldsymbol{\theta}$ are generally referring to the MLP parameters, and we do not mean that $\mu_{\boldsymbol{\theta}}$ and $\pi_{\boldsymbol{\theta}}$ share the same parameter values.

3) Off-Policy Extended Value Function Approximation: The next module in SEGAC is to instantiate an off-policy E-VFA algorithm, which is able to make use of the behavior policy ($\mu_{\boldsymbol{\theta}}$) generated trajectory data to approximate the evaluation policy (π)'s on-time arrival probability at any given state s. Recall Corollary 1, we can see that, essentially, for any state s, we have $b^{\pi}_{\boldsymbol{\phi}}(s) = \mathbb{E}_{a\sim\pi(a|s),s'\sim\mathbb{P}[s'|s,a]}\, b^{\pi}_{\boldsymbol{\phi}}(s')$. When a is generated by behavior policy μ, we can use importance sampling ratio correction method to reach the estimate of $b^{\pi}_{\boldsymbol{\phi}}(s)$ as follows:

$$b^{\pi}_{\boldsymbol{\phi}}(s) = \mathbb{E}_{a\sim\mu(a|s),s'\sim\mathbb{P}[s'|s,a]} \frac{\pi(a|s)}{\mu(a|s)} b^{\pi}_{\boldsymbol{\phi}}(s'), \quad (10)$$

where $\frac{\pi(a|s)}{\mu(a|s)}$ is the importance sampling ratio. With Eq. (10), we can reach the off-policy E-VFA algorithm whose pseudo code is depicted in Algorithm 4.

Algorithm 4: Off-Policy E-VFA

Input: (1) Evaluation policy network $\pi_{\boldsymbol{\theta}}$; (2) behavior policy network $\mu_{\boldsymbol{\theta}}$; (3) mini-batch size: K; (4) max. episodes: $T_{\max}$; (5) max. training iteration: T_{iter}
Output: Off-Policy E-VFA estimated baseline $b^{\pi}_{\boldsymbol{\phi}}$;
Init: (1) Initialize replay memory D; (2) Randomly initialize $\boldsymbol{\phi}$ for $b^{\pi}_{\boldsymbol{\phi}}$; (3) initialize the target network's parameter $\boldsymbol{\phi}^{-} = \boldsymbol{\phi}$; (4) $j \leftarrow 1$;

while $j \leq T_{\max}$ **do**
  Randomly initialize origin o, destination d and budget T;
  Run $\mu_{\boldsymbol{\theta}}$ and collect the trajectory, *i.e.*, $\tau_{\mu}^{(j)} = \big(s_0^{(j)}, a_0^{(j)}, r_1^{(j)}, s_1^{(j)}, \ldots, s_{H_j-1}^{(j)}, a_{H_j-1}^{(j)}, r_{H_j}^{(j)}, s_{H_j}^{(j)}\big)$;
  $\forall\, 0 \leq k \leq H_j - 1$, store transition $(s_k^{(j)}, a_k^{(j)}, r_{k+1}^{(j)}, s_{k+1}^{(j)})$ in D;
  $j \leftarrow j + 1$;
foreach $j \in \{1, 2, \ldots, T_{iter}\}$ **do**
  Sample K mini-batch transitions, *e.g.*, $(s_k, a_k, r_{k+1}, s_{k+1})$, from D;
  Set $y_k = \frac{\pi(a_k|s_k)}{\mu(a_k|s_k)} b^{\pi}_{\boldsymbol{\phi}^{-}}(s_{k+1})$;
  Perform a stochastic gradient descent step on $\frac{1}{K}\sum_{k=1}^{K}(y_k - b^{\pi}_{\boldsymbol{\phi}}(s_k))^2$ with respect to $\boldsymbol{\phi}$;
  Reset $\boldsymbol{\phi}^{-} = \boldsymbol{\phi}$;
Final.

4) Sample Efficient Generalized Actor Critic: With the off-policy GPG theorem, and the off-policy E-VFA algorithm for baseline calculation, we can reach the SEGAC algorithm, which makes use of the behavior policy generated trajectory for the evaluation policy network's parameter update. The core of SEGAC is to update the actor network's parameter with the

Algorithm 5: Sample Efficient Generalized Actor Critic (SEGAC)

Input: (1) origin o, destination d; (2) time budget: T; (3) learning rate: α; (4) max. iteration: $T_{\max}$; (5) pre-collected num. of trajectories: M; (6) behavior policy $\mu_{\boldsymbol{\theta}}$

Output: Policy network $\pi_{\boldsymbol{\theta}}$;

Init: (1) Randomly initialize the policy network $\pi_{\boldsymbol{\theta}}$; (2) counter $\leftarrow 0$;

Run $\mu_{\boldsymbol{\theta}}$ and collect M trajectories, *i.e.*, $\tau^{(j)}$;

while *counter* $\leq T_{\max} - 1$ **do**

  Calculate $b^{\pi}_{\boldsymbol{\phi}}$ with the off-policy E-VFA algorithm;

  Calculate $\boldsymbol{\nabla}_{\boldsymbol{\theta}} J_s(\pi_{\boldsymbol{\theta}})$ according to the off-policy generalized policy gradient approximation equation in Eq. (11);

  Update the policy network's parameter as: $\boldsymbol{\theta} = \boldsymbol{\theta} + \alpha \boldsymbol{\nabla}_{\boldsymbol{\theta}} J_s(\pi_{\boldsymbol{\theta}})$;

  counter $\leftarrow$ counter $+ 1$;

Final.

baseline subtracted generalized policy gradient as follows:

$$\boldsymbol{\nabla}_{\boldsymbol{\theta}} J_s(\pi_{\boldsymbol{\theta}}) = \frac{1}{M} \sum_{j=1}^{M} \big((\mathbb{1}\{R(\tau_{\mu}^{(j)}) \leq T\} - b^{\pi}_{\boldsymbol{\phi}}(s_0^{(j)})) \times \sum_{k=0}^{H_j - 1} \boldsymbol{\nabla}_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(a_k^{(j)} | s_k^{(j)}) \rho(\tau_{\mu}^{(j)}) \big). \quad (11)$$

The pseudo code is depicted in Algorithm 5. Note that, in Algorithm 5, we set an independent behavior policy, *i.e.*, $\mu_{\boldsymbol{\theta}}$, as the trajectory collection policy, and reuse the data for parameter update of the evaluation policy's actor-critic network. In this way, we depict the off-policy GPG and off-policy E-VFA update process explicitly. While in practice, we can just use the past evaluation policies as the behavior policy, and use the collected trajectory data from *past* evaluation policies to update the *current* evaluation policy's parameters.

D. Variance Reduction for SEGAC

The last subsection presents SEGAC as a sample-efficient solution to the SOTA problem. However, in practice, one of the main issues of SEGAC is its high variance due to the importance sampling ratio. In this subsection, we propose (1) the weighted importance sampling (WIS) method to reduce the variance of off-policy GPG, and (2) the control variate (CV) method to reduce the variance of off-policy E-VFA.

1) Weighted Importance Sampling for Off-Policy GPG: Although it can be proved that the expectation of the importance sampling ratio is equal to one, *i.e.*, $\mathbb{E}_{\tau \sim \mu_{\boldsymbol{\theta}}}[\rho(\tau)] = 1$, it has high variance especially when π and μ are quite different from each other. In order to reduce the variance during the generalized policy gradient's approximation procedure, we replace the importance sampling ratio $\big(\rho(\tau_{\mu}^{(j)})\big)$ in Eq. (9) with the weighted importance sampling, denoted as $\tilde{\rho}$. The weighted importance sampling for $\tau^{(j)}$ given M pre-collected trajectories is calculated as $\tilde{\rho}(\tau_{\mu}^{(j)}) = \rho(\tau_{\mu}^{(j)}) / \sum_{j=1}^{M} \rho(\tau_{\mu}^{(j)})$. With $\tilde{\rho}(\tau_{\mu}^{(j)})$ for $\tau_{\mu}^{(j)}$, the off-policy generalized policy gradient approximation procedure for $\pi_{\boldsymbol{\theta}}$ can be calculated as:

$$\boldsymbol{\nabla}_{\boldsymbol{\theta}} J_s(\pi_{\boldsymbol{\theta}}) = \frac{1}{M} \sum_{j=1}^{M} \big(\mathbb{1}\{R(\tau_{\mu}^{(j)}) \leq T\} \times \sum_{k=0}^{H_j - 1} \boldsymbol{\nabla}_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(a_k^{(j)} | s_k^{(j)}) \tilde{\rho}(\tau_{\mu}^{(j)}) \big). \quad (12)$$

It is straightforward to verify that $0 \leq \tilde{\rho}(\tau_{\mu}^{(j)}) \leq 1$, and in most cases, $\tilde{\rho}(\tau_{\mu}^{(j)})$ has much smaller variance than that of $\rho(\tau_{\mu}^{(j)})$. However, note that the off-policy generalized policy gradient approximation procedure for $\pi_{\boldsymbol{\theta}}$ as described in Eq. (12) is a *biased* estimate, but with much smaller variance than the unbiased one expressed in Eq. (9). In practice, one can choose between Eq. (9) and Eq. (12) depending on one's trade-off preferences between 'bias' and 'variance'.

2) Control Variate for Off-Policy E-VFA: For the off-policy E-VFA algorithm, one can also replace the importance sampling ratio in Eq. (10) with the weighted importance sampling to reduce variance. However, since the importance sampling ratio is only with respect to one state-action pair instead of the whole trajectory, replacement with WIS ($\tilde{\rho}$) is not reducing the variance significantly. The key factor that makes the E-VFA estimates exhibit high variance is that if $\pi(a|s)/\mu(a|s)$ is too small, *e.g.*, approaches zero, the target value also shrinks to almost zero, which makes the stochastic gradient descent update move towards a 'wrong' (high variance) direction.

In this paper, we propose a control variate (CV) method, which offsets the 'shrinked' target value with a 'remedy' value as follows:

$$y_k = \frac{\pi(a_k|s_k)}{\mu(a_k|s_k)} b_{\boldsymbol{\phi}^-}(s_{k+1}) + \big(1 - \frac{\pi(a_k|s_k)}{\mu(a_k|s_k)}\big) b_{\boldsymbol{\phi}}(s_k), \quad (13)$$

where the related variables can be referred to in Line 8 of Algorithm 4. Replacing Line 8 of Algorithm 4 with Eq. (13) yields the variance-reduced off-policy E-VFA algorithm, which we do not display the contents in the main manuscript, due to its high similarity to Algorithm 4. Note that the expected value of the added term in Eq. (13), *i.e.*, $\big(1 - \frac{\pi(a_k|s_k)}{\mu(a_k|s_k)}\big) b_{\boldsymbol{\phi}}(s_k)$, is zero, and it functions to reduce the variance of the estimation process of E-VFA. Therefore, the term is named as the 'control variate'.

E. State Space Encoding and Feature Training

So far, we have introduced the sample efficient generalized actor critic method with two variance reduction techniques. One more characteristic of SEGAC is the 'cold start' reliable navigation functionality, which means that the well-trained SEGAC is able to output an up-to-standard routing policy when we set an 'unforeseen' destination node. The functionality is implemented through condensed state space encoding and feature training.

In Section IV-A, we encode the ego vehicle's current residing node (i) as well as the destination node (d) with the one-hot embedding scheme. However, one-hot embedding essentially leads to a tabular solution, which is ungeneralizable to new destination nodes. In this paper, we adopt

the random embedding initialization and post feature training method as proposed in [41] to embed the nodes in Graph $\mathcal{G}$ and thereby endow SEGAC with the generalizable reliable navigation functionality.

Let $\varphi : \{0,1\}^n \rightarrow \mathcal{R}^{\tilde{d}}$ be a random embedding function, which maps the one-hot embedded n-dimensional node vector to the $\tilde{d}$-dimensional real-valued vector. Recall (in Section IV-A) that $\psi(i)$ maps node i to the n-dimensional one-hot embedded vector, we can initialize the feature of any node i in Graph $\mathcal{G}$ to the $\tilde{d}$-dimensional real-valued feature space, *i.e.*, $\forall\, i \in \mathcal{N}$, we have $\varphi(\psi(i))$ as node i's $\tilde{d}$-dimensional real-valued feature encoding. With φ, we can encode the ego vehicle state as follows: for example, the ego vehicle is at node i, with destination node d and remaining deadline T_i, we can embed the feature space as $s_i = [\varphi(\psi(i)); \varphi(\psi(d)); T_i]$, which is a $(2\tilde{d}+1)$-dimensional real-valued vector.

The random embedding function φ endows SEGAC with the generalization ability, however, the embedded feature is meaningless, and we need a method to update the feature vector to reach meaningful representations. Recall the off-policy generalized policy gradient approximation procedure in Eq. (9) or Eq. (12), one can update the first $2\tilde{d}$ dimensional vectors (T_i is the remaining budget which cannot be updated) by treating s_i itself also as a part of the policy network's parameters. Similarly, referring to Line 9 of Algorithm 4, one can also update s_i by treating it as a part of the critic network's parameters. In this way, the feature space of s_i is updated gradually towards meaningful representations, and SEGAC is able to output up-to-standard routing policies for unforeseen destination nodes. We will showcase the generalization functionality of SEGAC with experimental results in Section V-C.

F. Path-based SEGAC

Up to this point, the description of SEGAC primarily focuses on its capability of generating a dynamic routing policy based on the ego vehicle's current residing node (i), remaining budget T_i, and the destination node d. However, if the need arises for SEGAC to provide a pre-defined path-based solution for the SOTA problem, a straightforward adjustment can be made. Specifically, we configure the input features provided to the actor/critic network to include:

- origin node (o): signifying the starting point of the SOTA problem;
- pre-departure budget (T): representing the initial path planning budget;
- current residing node (i): indicating the ego vehicle's current residing node;
- destination node (d): specifying the SOTA problem's final destination.

One can observe that the ego vehicle's decision-making at the intermediate node i does not take the remaining budget T_i as input, but rather depends on the initial budget T at node o. In this way, SEGAC ensures that the resulting routing policy is deterministic and consistently connects the specified origin node o with the destination node d, thus providing a predefined path for addressing the SOTA *path* problem.

G. Computational Complexity Analysis

In this subsection, we use the big $\mathcal{O}$ notation [42] to analyze the computational complexity of SEGAC's training process as depicted in Algorithm 5. Examining Algorithm 5, we find that the core computational load happens between Line 3 and Line 6. Line 3 is essentially calculating the subtracting baseline obtained from the off-policy E-VFA algorithm. As b^{π}_{ϕ} is a multi-layer perceptron, with the number of input features as $2\tilde{d}+1$, number of hidden features as $2\times(2\tilde{d}+1)$, and number of output features as 1, we have the computational complexity of calculating the subtracting baseline as b^{π}_{ϕ} as $\mathcal{O}(2\times(2\tilde{d}+1)^2+2\times(2\tilde{d}+1))$, which can be represented as $\mathcal{O}(\tilde{d}^2)$ when omitting the non-leading terms. Line 4 functions to calculate the off-policy policy gradient approximation with Eq. (11), whose computational complexity can be expressed as $\mathcal{O}(M\times 1\times \bar{H}\times[(2\times(2\tilde{d}+1)^2+2\times(2\tilde{d}+1))+2\times(2\times(2\tilde{d}+1)^2+2\times(2\tilde{d}+1))])$, where $\bar{H}=\sum_{j=1}^{M}H_j/M$ refers to the average trajectory length of the pre-collected M trajectories. When omitting the non-leading terms, we can get that the computational complexity of Line 4 can be expressed as $\mathcal{O}(M\bar{H}\tilde{d}^2)$. Line 5 updates the policy parameters which simply has a computational complexity of $\mathcal{O}(\tilde{d})$. Line 3 to Line 5 are executed for $T_{\max}$ times. Therefore, the total computational complexity of SEGAC's training process can be expressed as $\mathcal{O}(\tilde{d}^2+M\bar{H}\tilde{d}^2+d)\times T_{\max}$, which can be simplified as $\mathcal{O}(M\bar{H}\tilde{d}^2T_{\max})$. From the expression, we can see that SEGAC is a polynomial computational complexity algorithm, with respect to the number of pre-collected trajectories (M), the average trajectory length ($\bar{H}$), the node embedding dimension ($\tilde{d}$), and the maximum training episodes $T_{\max}$.

V. Experimental Results and Analysis

In this section, we evaluate and compare SEGAC's performance with state-of-the-art SOTA solutions in a range of transportation networks. For state of the arts, we select (1) decreasing order of time (DOT) [18], (2) one-step mixed integer programming (OS-MIP) [23], (3) fourth moment approach (FMA) [2], (4) integer linear programming (ILP) [26], (5) Gaussian process path planning (GP3) [1], (6) practical Q-learning (PQL) [24], (7) cascaded temporal difference (CTD) [9]. Note that DOT and FMA are most recently proposed DP-based methodologies for the SOTA problem; OS-MIP and ILP are representative MP-based algorithms for SOTA; GP3 is a recently proposed special distribution (Gaussian) induced algorithm for the SOTA problem; and PQL and CTD are two recently proposed RL-based SOTA solutions, with CTD only applicable to Gaussian travel-time distribution assumptions. Additionally, since SEGAC is a *generalized* actor critic method, we also implement the vanilla actor critic (AC) method as the comparative baseline. Note that AC aims at maximizing the policy's expected cumulative return, which corresponds to the least expected time (LET) path in the path planning domain.

Table II summarizes the related parameter configurations for state-of-the-art SOTA solutions and SEGAC. All algorithms are implemented in Python 3.7, and we conduct the experiments on a 3.60 GHz, AMD Ryzen 5, 3600, 6-Core desktop

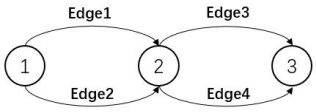
TABLE II
ALGORITHM PARAMETERS

Para.	Description	Algorithm	Value
α	learning rate	SEGAC, CTD, PQL	1e-3
M	pre-collected trajectories	SEGAC	100
$T_{\max}$	Max. iterations	SEGAC,ILP,OS-MIP	2e4
$\tilde{d}$	node embedding dimension	SEGAC	32
S	# of travel time samples	ILP, OS-MIP	50
$\text{Epi}_{\max}$	maximum episode number	SEGAC, CTD	8e4
$\bar{\alpha}$	variance learning rate	CTD	1e-3
ϵ	exploration strategy	CTD, PQL	0.05
K	# of shortest paths	PQL	5
ϑ	accuracy threshold	FMA	0.05
K_1	Max. # of policy evaluation	FMA	200
K_2	Max. # of policy iteration	FMA	200
δ	time window duration	DOT	0.01
η_r	relative tolerance	GP3	0.05

processor with the 64-bit version of Windows 10 operating system and 16GB RAM. The source code of SEGAC and all selected state-of-the-art SOTA solutions are publicly available[3]. In the following several subsections, we will (1) evaluate both the on-policy and off-policy functional modules within SEGAC, *i.e.*, OP-GPG, OP-GAC, off-policy GPG, SEGAC, and variance-reduced SEGAC, with a simple yet illustrative network; (2) benchmark SEGAC's performance and efficiency (in terms of the decision-making/planning time) with state-of-the-art SOTA solutions in four canonical transportation networks, namely Sioux Falls network (SFN), Anaheim, Winnipeg and Chicago Sketch; (3) evaluate SEGAC's unique functionality of providing the 'all-to-all' routing service in Anaheim, Winnipeg, and Chicago Sketch[4]; and (4) perform the ablation study of SEGAC in all four transportation networks.

A. SEGAC's Functional Module Evaluation

In this subsection, we perform the 'unit test' of SEGAC in a simple network as shown in Fig. 2(a). The corresponding edges' travel time distributions are displayed in Fig. 2(b). Note that SEGAC is a model-free solution to the SOTA problem, and thus it is not confined to any specific distribution model. The reason we choose normal distribution as the edges' distribution models, is that in this way, we are able to calculate, analytically, a given policy's true on-time arrival probability, and hence evaluate the correctness and optimality of all functional modules within SEGAC.



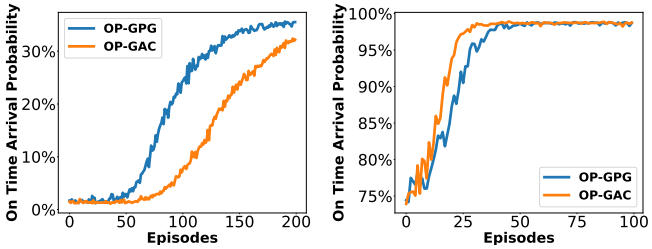
(a) The simple network

Edge	μ	σ^2	Distribution
1	10	0.1	Normal
2	10	4	Normal
3	10	0.1	Normal
4	10	4	Normal

(b) Travel-time distribution

Fig. 2. The Simple Network for Unit Test of SEGAC.

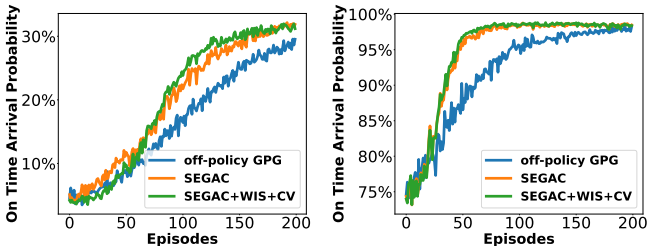
In Fig. 2(a), the ego vehicle is supposed to navigate from Node 1 to Node 3, with time budgets $T = 19$ and $T = 21$, respectively. With the given edges' distribution models, we can calculate that: (1) for $T = 19$, the ego vehicle should take Edge 2 and Edge 4 as the optimal path, which has an SOTA probability of $\Phi(\frac{19-20}{\sqrt{8}}) = 0.36$; (2) for $T = 21$, the ego vehicle should take Edge 1 and Edge 3 as the optimal path, which has an SOTA probability of $\Phi(\frac{21-20}{\sqrt{0.2}}) = 0.99$. Here, $\Phi(x)$ refers to the cumulative distribution function (CDF) of the standard normal distribution. Next, we evaluate OP-GPG, OP-GAC, off-policy GPG, SEGAC and variance-reduced SEGAC to see whether they ultimately converge to the optimal path[5]. For all off-policy modules within SEGAC, we fix the behavior policy $\mu_{\boldsymbol{\theta}}$ as selecting available actions/edges from the uniform randomness, *e.g.*, at Node 1, $\mu_{\boldsymbol{\theta}}$ selects Edge 1 and Edge 2 with equal probabilities of 0.5.



(a) $T = 19$ (b) $T = 21$

Fig. 3. Learning Processes of On-Policy Modules in SEGAC.

Fig. 3(a) and Fig. 3(b) show the learning processes of on-policy modules within SEGAC for $T = 19$ and $T = 21$, respectively. In the figure, we observe that OP-GPG learns faster than OP-GAC on the difficult task, *i.e.*, $T = 19$, which has less than 40% on-time arrival probability, while OP-GAC converges faster than OP-GPG for the easy task, *i.e.*, $T = 21$. We conjecture the underlying reason is that when facing the difficult task, OP-GPG will put emphasize on the sparse reward, and 'drag' the policy out of bad performance. While for the easy task, since there are already a lot of positive rewards, OP-GPG is 'satisfied' with the current policy, which makes the policy slow to change with the self-rewarding mechanism. For OP-GAC, even when facing a lot of positive rewards, the baseline subtraction might still deem the policy as non-optimal, and update the policy towards better ones.



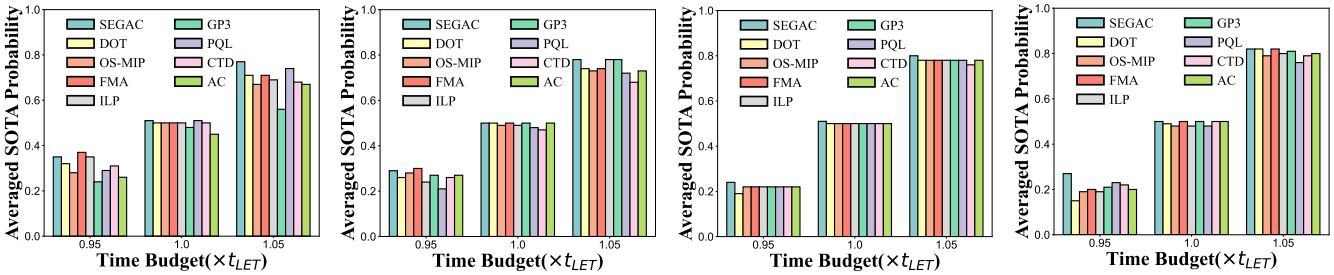
(a) $T = 19$ (b) $T = 21$

Fig. 4. Learning Processes of Off-Policy Modules in SEGAC.

[3]https://github.com/YoZo-X/SEGAC

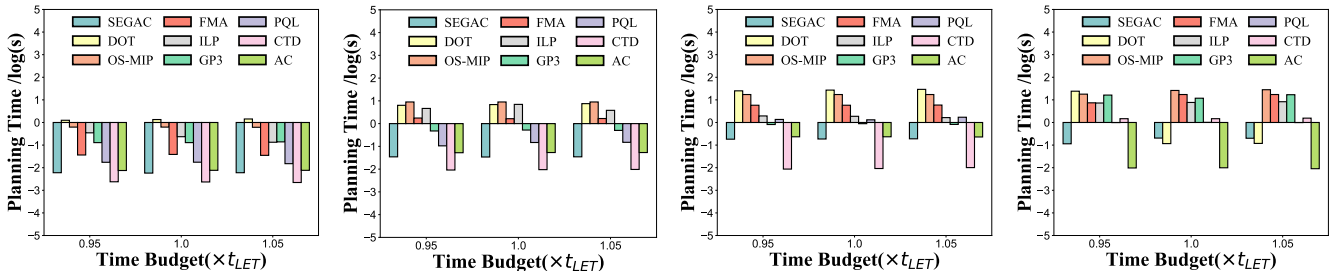
[4]Note that we skip SEGAC's generalization evaluation in SFN, in that the size of the network is too small, and one can simply enumerate all OD pairs and perform training separately.

[5]Note that E-VFA and off-policy E-VFA are merely baseline estimators for OP-GAC and SEGAC, respectively, and thus we cannot independently evaluate the corresponding performance.



(a) Sioux Falls Network (b) Anaheim (c) Winnipeg (d) Chicago Sketch

Fig. 5. The Performance Comparison of SEGAC and State-of-the-art SOTA Solutions.



(a) Sioux Falls Network (b) Anaheim (c) Winnipeg (d) Chicago Sketch

Fig. 6. The Efficiency Comparison (Decision-Making/Planning Time) of SEGAC and State-of-the-art SOTA Solutions (The unit of y-axis is log of seconds).

For the off-policy modules within SEGAC, *i.e.*, off-policy GPG, SEGAC and variance-reduced SEGAC, the related performance evolution process for both T values are reported in Fig. 4(a) and Fig. 4(b), respectively[6]. In both figures, we observe that both SEGAC and variance-reduced SEGAC (SEGAC+WIS+CV in the figure) are converging faster than off-policy GPG, in that they make better use of the behavior policy generated trajectory samples with baseline subtraction and importance sampling ratio normalization. We further observe that there is no significant difference in terms of learning rate between SEGAC and variance-reduced SEGAC for the simple network, in that there are altogether only two decision-making stages, and the importance sampling ratio $\big(\rho(\tau)\big)$ does not fluctuate significantly among different trajectories. However, for large scale networks, variance-reduced SEGAC exhibits low-variance convergence characteristics, and we report the related comparative results in the publicly available github repository.

B. Performance Comparison with State of the Arts

This subsection compares the performance and efficiency of SEGAC with the pre-selected collection of state-of-the-art SOTA solutions in four canonical transportation networks, namely Sioux Falls network (SFN), Anaheim, Winnipeg, and Chicago Sketch. The network details, *i.e.*, the number of nodes and edges, are presented in Table III.

The edges' mean travel times of all the four transportation networks are publicly available [43], and we follow the same procedure as proposed in [44] to generate the standard deviation value of each edge in the network. Specifically, we set $\sigma_{ij} = \text{Uniform}(0, \kappa)\mu_{ij}$, where $\kappa = 0.4$, and assume that all edges' travel time follow the normal distribution. Note that although SEGAC is model free, DOT requires full distribution models; CTD and GP3 require the (multi-variate) Gaussian distribution assumption to be applicable to the SOTA problem. Therefore, we choose normal distribution in the experiments.

TABLE III
TEST NETWORK SUMMARY

Test network	# of edges(m)	# of nodes(n)
Sioux Falls Network (SFN)	76	24
Anaheim	914	416
Winnipeg	2836	1052
Chicago Sketch	2950	933

For performance evaluation, we randomly generate 50 origin-destination (OD) pairs with rejection (reject those OD pairs which are too near to each other) for Anaheim, Winnipeg, and Chicago Sketch and 5 OD pairs for SFN since it is a relatively small network with the fewest number of nodes and edges. Following the work in [15], for each selected OD pair, we also evaluate three use cases, with $T = 0.95 \times t_{\text{LET}}$, $T = 1.0 \times t_{\text{LET}}$ and $T = 1.05 \times t_{\text{LET}}$, respectively. Note that t_{LET} refers to the corresponding OD pair's least expected travel time. Unlike canonical machine learning problems, which gauge the performance of an algorithm with classification accuracy or F1 score, the SOTA navigation problem is essentially a decision-making problem, and we use the evaluating algorithms' averaged SOTA probability over all 50 OD pairs as the performance metric for comparison.

Fig. 5 shows the performance comparisons between SEGAC

[6]Here, we separate the showcase of SEGAC's on-policy and off-policy modules, in that the key characteristic of off-policy modules is sample/data efficiency, instead of faster convergence.

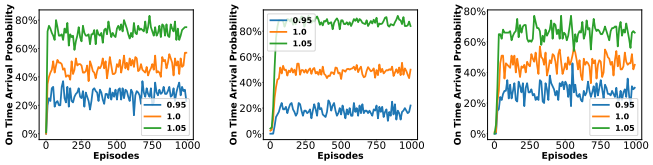
and state-of-the-art SOTA solutions for all four transportation networks. In the figure, we can observe that (1) almost all algorithms' performance patterns are consistent, *i.e.*, the SOTA probability increases as we increase the planning budgets; (2) SEGAC achieves, on average, the best performance across the four transportation networks for all evaluated budget values; (3) all algorithms including SEGAC are yielding quite similar performance metrics when we set $T = 1 \times t_{\text{LET}}$, in that when $T = 1 \times t_{\text{LET}}$, most of the time, the optimal routing policy is to take the LET path with 50% SOTA probability; and (4) AC merely outputs the LET path, which corresponds to a mediocre performance among others. We believe that the underlying reason of SEGAC achieving better SOTA performance than other machine learning (ML)-based algorithms, *i.e.*, PQL and CTD, mainly attributes to SEGAC's off-policy nature, which enables it to reuse the past trajectories for the additional training process. While both PQL and CTD are essentially on-policy learning algorithms, which essentially discards all past trajectory data for the current policy's training process.

For a well-trained SEGAC network, the decision-making time depends only on how fast we can compute a given actor network's output, which merely involves simple algebraic computations. Therefore, SEGAC is able to make decisions in real time, *i.e.*, less than one second for the referred hardware setup. Fig. 6 shows the decision-making time comparisons for all the algorithms. In the figure, we can see that all learning algorithms, *i.e.*, SEGAC, CTD, PQL and AC are able to make online decisions within one second, while DP-based and MP-based planning methods incur relatively more time before reaching the respective SOTA solutions. Here, we wish to note that since CTD is essentially a ***tabular*** solution to the SOTA problem, its decision making time is the shortest among all benchmark algorithms. However, CTD presumes that the travel time follows the single variate normal distribution, and thus cannot be applied to other travel-time distribution models or offer a model-free solution. Moreover, SEGAC possesses a unique 'all-to-all' navigation characteristic among all SOTA solutions, which will be evaluated in the next subsection.

C. SEGAC's Generalization Ability Evaluation

One unique characteristic of SEGAC when compared with almost all other SOTA solutions is that SEGAC is able to offer the 'all-to-all' routing service, *i.e.*, the well-trained actor-critic network has the generalization ability to navigate the ego vehicle for a completely new OD pair that has not been trained before. The underlying reason is that the real-valued node embedding vector endows SEGAC with the generalized decision-making ability. Since state-of-the-art SOTA algorithms offer either 'one-to-one' solution or 'all-to-one' solution, and cannot supply the 'all-to-all' routing service, we evaluate SEGAC's generalization ability without making comparisons with the aforementioned baseline algorithms in this subsection.

For Anaheim, Winnipeg, and Chicago Sketch, we randomly generate 100 OD pairs with rejection (reject those OD pairs which are too near to each other), and train SEGAC with the first 50 OD pairs, and test SEGAC's performance with the remaining 50 OD pairs. For each training/testing OD pair, we set three budget values, *i.e.*, $T = 0.95 \times t_{\text{LET}}$, $T = 1 \times t_{\text{LET}}$ and $T = 1.05 \times t_{\text{LET}}$. Note that in this subsection, we omit the Sioux Falls network, in that (1) the size of SFN is too small (24 nodes altogether as shown in Table III) to generate 100 different OD pairs with rejection requirements; (2) the node embedding dimension ($\tilde{d} = 32$) is larger than the total number of nodes in SFN (24), which makes the embedding process and thus generalization ability less meaningful. In this case, one may simply enumerate all the destination nodes, and perform the training process separately.



(a) Anaheim (b) Winnipeg (c) Chicago Sketch

Fig. 7. The Generalization Ability Test of SEGAC.

Fig. 7 shows the learning processes of SEGAC's generalization ability for the three transportation networks. In the figure, we can see that SEGAC is able to output an up-to-standard performance for completely un-trained OD pairs. For example, for $T = t_{\text{LET}}$, SEGAC is able to reach 50% SOTA probability after around 100 training episodes. Here, we wish to note that in order for SEGAC to have a 'meaningful' generalization ability, the testing OD pairs need to have been encoded properly during the training process, *i.e.*, the testing OD pairs have been visited by SEGAC's routing policy during training.

D. Ablation Study of SEGAC

In the last three subsections, we perform the 'unit test' of SEGAC in a simple network, benchmark SEGAC's performance and efficiency with state-of-the-art SOTA solutions in four realistic transportation networks, and evaluate SEGAC's generalization ability in Anaheim, Winnipeg and Chicago Sketch, respectively. This subsection performs the ablation study by comparing the performance of GPG, GPG$^+$ (GPG with a constant baseline subtraction), GPG+E-VFA, OP-GPG, OP-GPG$^+$ (OP-GPG with a constant baseline subtraction) and SEGAC (OP-GPG+off-policy E-VFA). Two key considerations are as follows: (1) The constant subtraction baseline for GPG$^+$ and OP-GPG$^+$ is set at 0.5. (2) The first three algorithms, namely, GPG, GPG$^+$, and GPG+E-VFA, are categorized as on-policy RL algorithms. On the other hand, the last three algorithms, which include OP-GPG, OP-GPG$^+$ and SEGAC, fall under the domain of off-policy RL algorithms. It's worth mentioning that the primary advantage of off-policy RL algorithms lies in their sample efficiency. To highlight this aspect, we have set a relatively smaller maximum iteration number ($T_{\max} = 1e3$). This allows for on-policy RL algorithms to *not* reach the ultimate convergence, thus showcasing the benefits of off-policy RL algorithms.

Table IV presents the results of our ablation study, with several noteworthy findings: (1) For small testing networks like SFN and Anaheim, both on-policy and off-policy algorithms

TABLE IV
ABLATION STUDY OF THE SEGAC ALGORITHM

Testing Network	T ($\times t_{\text{LET}}$)	GPG	GPG$^+$	GPG+ E-VFA	OP-GPG	OP-GPG$^+$	SEGAC
SFN	0.95	0.3686	0.3703	0.3784	0.3744	0.3756	**0.3795**
	1.0	0.4986	0.5040	**0.5050**	0.5028	0.5029	0.5045
	1.05	0.6592	0.6659	0.6689	0.6565	0.6631	**0.6700**
Anaheim	0.95	0.2612	0.2695	0.2705	0.2688	**0.2719**	0.2718
	1.0	0.5067	0.5056	0.5116	0.5040	0.5060	**0.5155**
	1.05	0.7580	0.7603	0.7607	0.7592	0.7610	**0.7616**
Winnipeg	0.95	0.2405	0.2413	0.2427	0.2493	0.2505	**0.2591**
	1.0	0.5010	0.5049	0.5059	0.5070	0.5110	**0.5187**
	1.05	0.7591	0.7609	0.7699	0.7846	0.7850	**0.7991**
Chicago Sketch	0.95	0.2512	0.2525	0.2597	0.2653	0.2750	**0.2809**
	1.0	0.5016	0.5044	0.5134	0.5158	0.5202	**0.5291**
	1.05	0.7706	0.7750	0.7860	0.7795	0.7969	**0.8106**

exhibit reasonably good and barely discernible performance in terms of the SOTA probability across different time budgets. (2) for large-scale testing networks, such as Winnipeg and Chicago Sketch, the advantages of incorporating a baseline estimator (E-VFA) and utilizing off-policy modules become more evident. Specifically, GPG+E-VFA outperforms the other two on-policy algorithms, and off-policy algorithms consistently achieve better SOTA performance than their counterparts, primarily due to their sample efficiency. (3) Notably, SEGAC which integrates both an off-policy actor network (OP-GPG) and an off-policy value network (off-policy E-VFA), delivers the best overall performance in Winnipeg and Chicago Sketch.

VI. PERFORMANCE EVALUATION IN METROPOLITAN TRANSPORTATION NETWORKS: CHENGDU AND BEIJING

In the last section, we perform the 'unit test' of SEGAC in a self-constructed simple network; compare SEGAC's performance and decision-making time with state-of-the-art SOTA solutions in a range of canonical transportation networks, and showcase SEGAC's generalization ability to new OD pairs. However, for all the evaluation networks, we presume that the edges' travel times follow the normal distribution, and use it to get the required data samples for SEGAC's training and testing. While SEGAC is a 'model-free' solution to the SOTA problem, which does not pose any travel-time distribution modeling constraint. Therefore, in this section, we deploy SEGAC to two metropolitan transportation networks, namely Chengdu and Beijing, with real traffic data.

The two metropolitan transportation networks, as illustrated in Fig. 8, represents the main transportation networks of Chengdu, China and Beijing, China, respectively. Travel speed samples of the edges are measured through loop detectors, which, together with the link lengths, are provided in [45] and [22], respectively.

Since the travel time distributions are different in different times slots of the day, *e.g.*, peak hour, off-peak hour, and across different types of the day, *e.g.*, weekday, weekend, we separate the real traffic data into four different groups, namely, weekday peak hour[7], weekday off-peak hour, weekend peak hour, weekend off-peak hour. In the following, we will evaluate how SEGAC and other state-of-the-art SOTA algorithms perform in different types of time slots *i.e.*, weekday, weekend, peak hour, off-peak hour, and different path planning budgets (T) *e.g.*, tight budget ($T = 0.95 \times t_{\text{LET}}$) and loose budget ($T = 1.05 \times t_{\text{LET}}$). For each use case, *e.g.*, off-peak hour, weekday, tight budget, we randomly generate 50 OD pairs, and train the respective SOTA algorithms, and report the averaged arrival on-time probability for Chengdu and Beijing, in Table V and Table VI, respectively. Note that we eliminate several state-of-the-art SOTA solutions in that (1) we cannot apply ILP and OS-MIP to Chengdu and Beijing, in that the inherent integer programming will cause the program to quit; (2) GP3 and CTD assume that the underlying travel-time distribution is (multi-variate) Gaussian, which is not the case for real traffic data, therefore, we did not include the respective algorithms' performance; (3) the canonical AC algorithm, when applied to the path planning domain, is essentially searching for the LET path connecting the origin node with the destination node. AC's training time is huge in metropolitan transportation networks, and its ultimate performance is exactly the same as that of an LET path. Therefore, in the experiment, we just straightforwardly evaluate the LET path's performance in different use cases, and replace the algorithm AC as LET.

(a) Chengdu (b) Beijing

Fig. 8. Illustration of the two metropolitan transportation networks

Examining Table V and Table VI, we can see that (1) SEGAC achieves the best performance in the peak hour use cases, *i.e.*, weekday peak hour and weekend peak hour for both two networks; (2) for off-peak hour use cases, the LET path already offers up-to-standard service in most configurations. For (1), we conjecture that in peak hour, there are many 'good' path selection options, and it is difficult to *calculate* the exact optimal one with planning-based methods. Therefore, SEGAC excels in learning to make the optimal decision and achieves the best overall performance. While for (2), in off-peak hour, since there is not much traffic, the optimal path is relatively simple, and in most cases, it is exactly the LET path. In this case, all algorithms are yielding similar performance, and LET has the best overall performance in most cases.

VII. CONCLUSION AND FUTURE WORK

This paper proposes the sample efficient generalized actor critic (SEGAC) algorithm for the stochastic on-time arrival (SOTA) problem in transportation networks. While canonical SOTA solutions either incur high-complexity convolution computation (DP-based solution) or call up off-the-shelf integer linear programming solvers (MP-based solution), which

[7]peak hours are defined as the following two time slots: (1) between 7:30am to 9:30am, and (2) between 5:00pm to 7:00pm.

TABLE V
PERFORMANCE COMPARISON OF SEGAC, FMA DOT, PQL AND LET FOR DIFFERENT TRAVEL TIME BUDGETS IN CHENGDU NETWORK

Period of analysis	T ($\times t_{\text{LET}}$)	SEGAC	FMA	DOT	PQL	LET
Weekday (Peak hour)	0.95	**0.3238**	0.3168	0.3076	0.3131	0.3151
	1.0	**0.5034**	0.4999	0.4946	0.4857	0.4999
	1.05	**0.6871**	0.6854	0.6757	0.6179	0.6848
Weekday (Off-peak hour)	0.95	0.2824	0.2825	0.2526	0.2798	**0.2862**
	1.0	0.4812	0.4943	0.4979	0.4769	**0.4999**
	1.05	0.7032	0.7083	0.7063	0.7028	**0.7137**
Weekend (Peak hour)	0.95	**0.3328**	0.3258	0.3185	0.2965	0.3204
	1.0	**0.5023**	0.5000	0.4945	0.4881	0.5000
	1.05	**0.6925**	0.6812	0.6754	0.6677	0.6795
Weekend (Off-peak hour)	0.95	0.2772	0.2803	0.2539	0.2564	**0.2807**
	1.0	0.4907	0.4949	0.4959	0.4557	**0.5000**
	1.05	0.7104	**0.7241**	0.7160	0.6791	0.7192

TABLE VI
PERFORMANCE COMPARISON OF SEGAC, FMA DOT, PQL AND LET FOR DIFFERENT TRAVEL TIME BUDGETS IN BEIJING NETWORK

Period of analysis	T ($\times t_{\text{LET}}$)	SEGAC	FMA	DOT	PQL	LET
Weekday (Peak hour)	0.95	**0.3486**	0.3184	0.3457	0.3418	0.3466
	1.0	**0.5155**	0.4812	0.5012	0.5000	0.5000
	1.05	**0.6534**	0.6511	0.6528	0.6432	0.6518
Weekday (Off-peak hour)	0.95	0.3486	0.3409	0.3439	0.3459	**0.3493**
	1.0	0.4997	0.4931	0.4966	0.4992	**0.5000**
	1.05	0.6572	0.6517	0.6529	0.6551	**0.6578**
Weekend (Peak hour)	0.95	**0.3542**	0.2914	0.3416	0.3409	0.3414
	1.0	**0.5181**	0.4926	0.5018	0.4999	0.5000
	1.05	**0.6536**	0.6400	0.6435	0.6263	0.6530
Weekend (Off-peak hour)	0.95	0.3415	0.3408	**0.3433**	0.3370	**0.3433**
	1.0	**0.5005**	0.4912	0.4999	0.5000	0.5000
	1.05	0.6648	0.6565	0.6570	0.6659	**0.6666**

inherently possess non-polynomial computational complexity, SEGAC treats the SOTA problem from reinforcement learning's perspective. Within SEGAC, we propose the *generalized* policy gradient (GPG) and the *extended* value function approximation (E-VFA) modules to cope with the non-additive nature of the SOTA problem, and also upgrade both modules to the respective off-policy versions for sample efficiency. Two variance reduction techniques (weighted importance sampling and control variate) are also proposed to reduce the variance of SEGAC. Compared with state-of-the-art SOTA solutions, SEGAC offers a (1) model-free, (2) sample-efficient, (3) all-to-all, and (4) fast decision-making solution to the SOTA problem, and we validate those characteristics with a series of experiments in several canonical transportation networks.

In the future, we would like to extend SEGAC's application to spatial and temporal correlated travel-time distribution use cases. Furthermore, we are also keen on proposing an *offline* version of SEGAC, which is able to be trained with human-driver collected travel-time samples and in this case, SEGAC does not need a pre-set behavior policy to collect data samples. Another interesting future work direction is to apply modern machine learning based approaches, *e.g.*, neural heuristic analysis, to solving the SOTA problem.

ACKNOWLEDGMENT

The authors would like to thank the Editor-in-Chief, Associate Editor, and three anonymous reviewers for reviewing the manuscript and providing the invaluable comments and suggestions.

REFERENCES

[1] H. Guo, X. Hou, Z. Cao, and J. Zhang, "GP3: Gaussian process path planning for reliable shortest path in transportation networks," *IEEE Transactions on Intelligent Transportation Systems (T-ITS)*, vol. 23, no. 8, pp. 11 575–11 590, 2022.

[2] H. Guo, Z. He, C. Gao, and D. Rus, "Navigation with time limits in transportation networks: A fourth moment approach," *IEEE Transactions on Intelligent Transportation Systems (T-ITS)*, vol. 23, no. 11, pp. 1–16, 2022.

[3] Y. Liu, S. Blandin, and S. Samaranayake, "Stochastic on-time arrival problem in transit networks," *Transportation Research Part B: Methodological (TR-B)*, vol. 119, pp. 122–138, 2019.

[4] Y. Fan, R. Kalaba, and J. Moore, "Arriving on time," *Journal of Optimization Theory and Applications*, vol. 127, no. 3, pp. 497–513, 2005.

[5] A. Khani and S. D. Boyles, "An exact algorithm for the mean–standard deviation shortest path problem," *Transportation Research Part B: Methodological*, vol. 81, pp. 252–266, 2015.

[6] S. Biswal, G. Ghorai, and S. Mohanty, "α-reliable shortest path problem in uncertain time-dependent networks," *International Journal of Applied and Computational Mathematics*, vol. 8, no. 4, p. 164, 2022.

[7] M. Filipovska and H. S. Mahmassani, "Reliable least-time path estimation and computation in stochastic time-varying networks with spatio-temporal dependencies," in *IEEE International Conference on Intelligent Transportation Systems (ITSC)*, 2020, pp. 1–6.

[8] Z. Zang, X. Xu, K. Qu, R. Chen, and A. Chen, "Travel time reliability in transportation networks: A review of methodological developments," *Transportation Research Part C: Emerging Technologies (TR-C)*, vol. 143, p. 103866, 2022.

[9] H. Guo, X. Hou, and Q. Peng, "CTD: Cascaded temporal difference learning for the mean-standard deviation shortest path problem," *IEEE Transactions on Intelligent Transportation Systems (T-ITS)*, vol. 23, no. 8, pp. 10 868–10 886, 2022.

[10] A. Agafonov, V. Myasnikov, and A. Maksimov, "Approximation of the road segments travel time using Lévy distributions in the reliable shortest path problem," in *Journal of Physics: Conference Series*, vol. 1368, no. 3. IOP Publishing, 2019, p. 032008.

[11] A. Flajolet, S. Blandin, and P. Jaillet, "Robust adaptive routing under uncertainty," *Operations Research*, vol. 66, no. 1, pp. 210–229, January 2018.

[12] M. Niknami and S. Samaranayake, "Tractable pathfinding for the stochastic on-time arrival problem," in *Experimental Algorithms*, A. V. Goldberg and A. S. Kulikov, Eds. Springer International Publishing, 2016, pp. 231–245.

[13] L. Shen, H. Shao, L. Zhang, and J. Zhao, "The global optimal algorithm of reliable path finding problem based on backtracking method," *Mathematical Problems in Engineering*, vol. 2017, pp. 1–10, 2017.

[14] F. Manseur, N. Farhi, C. N. Van Phu, H. Haj-Salem, and J.-P. Lebacque, "Robust routing, its price, and the tradeoff between routing robustness and travel time reliability in road networks," *European Journal of Operational Research (EJOR)*, vol. 285, no. 1, pp. 159–171, 2020.

[15] C. Gao, H. Guo, and W. Sheng, "GP4: Gaussian process proactive path planning for the stochastic on time arrival problem," *IEEE Transactions on Vehicular Technology (T-VT)*, vol. 70, no. 10, pp. 9849–9862, 2021.

[16] S. Lim, C. Sommer, E. Nikolova, and D. Rus, "Practical route planning under delay uncertainty: Stochastic shortest path queries," in *Robotics: Science and Systems (RSS)*, 2012, pp. 249–256.

[17] Y. M. Nie and X. Wu, "Reliable a priori shortest path problem with limited spatial and temporal dependencies," in *Transportation and Traffic Theory 2009: Golden Jubilee*. Springer, 2009, pp. 169–195.

[18] A. Arun Prakash, "Algorithms for most reliable routes on stochastic and time-dependent networks," *Transportation Research Part B: Methodological (TR-B)*, vol. 138, pp. 202–220, 2020.

[19] B. Y. Chen, C. Shi, J. Zhang, W. H. Lam, Q. Li, and S. Xiang, "Most reliable path-finding algorithm for maximizing on-time arrival probability," *Transportmetrica B: Transport Dynamics*, vol. 5, no. 3, pp. 248–264, 2017.

[20] G. Sabran, S. Samaranayake, and A. Bayen, "Precomputation techniques for the stochastic on-time arrival problem," in *Proceedings of the Meeting on Algorithm Engineering and Experiments (ALENEX)*. Society for Industrial and Applied Mathematics, 2014, pp. 138–146.

[21] Z. Cao, H. Guo, J. Zhang, D. Niyato, and U. Fastenrath, "Finding the shortest path in stochastic vehicle routing: A cardinality minimization approach," *IEEE Transactions on Intelligent Transportation Systems (T-ITS)*, vol. 17, no. 6, pp. 1688–1702, 2016.

[22] Z. Cao, H. Guo, J. Zhang, D. Niyato, and U. Fastenrath, "Improving the efficiency of stochastic vehicle routing: A partial Lagrange multiplier method," *IEEE Transactions on Vehicular Technology (T-VT)*, vol. 65, no. 6, pp. 3993–4005, 2015.

[23] L. Yang and X. Zhou, "Optimizing on-time arrival probability and percentile travel time for elementary path finding in time-dependent transportation networks: Linear mixed integer programming reformulations," *Transportation Research Part B: Methodological (TR-B)*, vol. 96, pp. 68–91, 2017.

[24] Z. Cao, H. Guo, W. Song, K. Gao, Z. Chen, L. Zhang, and X. Zhang, "Using reinforcement learning to minimize the probability of delay occurrence in transportation," *IEEE Transactions on Vehicular Technology (T-VT)*, vol. 69, no. 3, pp. 2424–2436, 2020.

[25] Y. Pan, "A nonlinear mixed integer programming method to find the reliable shortest path in a stochastic network," in *16th COTA International Conference of Transportation Professionals*, 2016, pp. 71–77.

[26] Z. Cao, Y. Wu, A. Rao, F. Klanner, S. Erschen, W. Chen, L. Zhang, and H. Guo, "An accurate solution to the cardinality-based punctuality problem," *IEEE Intelligent Transportation Systems Magazine (MITS)*, vol. 12, no. 4, pp. 78–91, 2020.

[27] K. K. Srinivasan, A. Prakash, and R. Seshadri, "Finding most reliable paths on networks with correlated and shifted log–normal travel times," *Transportation Research Part B: Methodological (TR-B)*, vol. 66, no. C, pp. 110–128, 2014.

[28] B. Y. Chen, W. H. K. Lam, A. Sumalee, Q. Li, and M. L. Tam, "Reliable shortest path problems in stochastic time-dependent networks," *Journal of Intelligent Transportation Systems*, vol. 18, no. 2, pp. 177–189, 2014.

[29] M. Ruß, G. Gust, and D. Neumann, "The constrained reliable shortest path problem in stochastic time-dependent networks," *Operations Research*, vol. 69, no. 3, pp. 709–726, 2021.

[30] K. Ahmadi and V. Allan, "Congestion-aware stochastic path planning and its applications in real world navigation," in *Proceedings of the 13th International Conference on Agents and Artificial Intelligence - Volume 2: ICAART,*, INSTICC. SciTePress, 2021, pp. 947–956.

[31] H. Guo, W. Sheng, C. Gao, and Y. Jin, "DRL-Router: Distributional reinforcement learning-based router for reliable shortest path problems," *IEEE Intelligent Transportation Systems Magazine (MITS)*, pp. 2–19, 2023.

[32] W. Kool, H. van Hoof, and M. Welling, "Attention, learn to solve routing problems!" in *International Conference on Learning Representations*, 2018.

[33] J. Li, Y. Ma, R. Gao, Z. Cao, A. Lim, W. Song, and J. Zhang, "Deep reinforcement learning for solving the heterogeneous capacitated vehicle routing problem," *IEEE Transactions on Cybernetics*, vol. 52, no. 12, pp. 13 572–13 585, 2022.

[34] Y. Wu, W. Song, Z. Cao, J. Zhang, and A. Lim, "Learning improvement heuristics for solving routing problems," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 33, no. 9, pp. 5057–5069, 2021.

[35] Y.-D. Kwon, J. Choo, B. Kim, I. Yoon, Y. Gwon, and S. Min, "POMO: Policy optimization with multiple optima for reinforcement learning," in *Advances in Neural Information Processing Systems*, vol. 33, 2020, pp. 21 188–21 198.

[36] R. Zhang, C. Zhang, Z. Cao, W. Song, P. S. Tan, J. Zhang, B. Wen, and J. Dauwels, "Learning to solve multiple-TSP with time window and rejections via deep reinforcement learning," *IEEE Transactions on Intelligent Transportation Systems*, pp. 1–12, 2022.

[37] G. Andonov and B. Yang, "A new formulation of the shortest path problem with on-time arrival reliability," *ArXiv*, vol. abs/1804.07829, 2018.

[38] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. MIT press, 2018.

[39] C. J. C. H. Watkins and P. Dayan, "Q-learning," *Machine Learning*, vol. 8, no. 3, pp. 279–292, 1992.

[40] V. Konda and J. Tsitsiklis, "Actor-critic algorithms," in *Advances in Neural Information Processing Systems (NeurIPS)*, S. Solla, T. Leen, and K. Müller, Eds., vol. 12. MIT Press, 1999, pp. 1008–1014.

[41] K. Kumar, P. Passban, M. Rezagholizadeh, Y. Lau, and Q. Liu, "From fully trained to fully random embeddings: Improving neural machine translation with compact word embedding tables," in *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, vol. 36, no. 10, Jun. 2022, pp. 10 930–10 937.

[42] D. E. Knuth, "Big omicron and big omega and big theta," *ACM Sigact News*, vol. 8, no. 2, pp. 18–24, 1976.

[43] H. Bar-Gera, "Transportation network test problems," Last Accessed May 11 2023. [Online]. Available: https://github.com/bstabler/TransportationNetworks

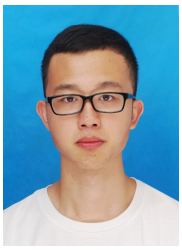
[44] Y. Zhang and A. Khani, "An algorithm for reliable shortest path problem with travel time correlations," *Transportation Research Part B: Methodological*, vol. 121, no. C, pp. 92–113, 2019.

[45] F. Guo, D. Zhang, Y. Dong, and Z. Guo, "Urban link travel speed dataset from a megacity road network," *Scientific Data*, vol. 6, no. 61, pp. 1–8, 05 2019.

List of Acronyms

AC	actor critic
CDF	cumulative distribution function
CTD	cascaded temporal difference
CV	control variate
DOT	decreasing order of time
DP	dynamic programming
E-VFA	extended value function approximation
FMA	fourth moment approach
GP3	Gaussian process path planning
GP4	Gaussian process *proactive* path planning
GPG	generalized policy gradient
ILP	integer linear programming
ITS	intelligent transportation system
LET	least expected time
MC	Monte Carlo
MDP	Markov decision process
MILP	mixed integer linear programming
ML	machine learning
MLP	multi-layer perceptron
MP	mathematical programming
OD	origin-destination
OP-GAC	on-policy generalized actor critic
OP-GPG	on-policy generalized policy gradient
OS-MIP	one-step mixed integer programming
PG	policy gradient
PQL	practical Q-learning
RL	reinforcement learning
RV	random variable
SA	successive approximation
SEGAC	sample efficient generalized actor critic
SFN	Sioux Falls network
SLN	shifted log-normal
SOTA	stochastic on-time arrival
TD	temporal difference
VRP	vehicle routing problem
WIS	weighted importance sampling

Hongliang Guo received his BE and ME in Beijing Institute of Technology, China, in 2005 and 2007 respectively. He holds a PhD degree from Stevens Institute of Technology, USA. He has been serving as postdoc researchers in Nanyang Technological University (NTU) and Massachusetts Institute of Technology (MIT), respectively. In 2021, He joins the institute of infocomm research (I2R) in A*STAR, Singapore, as a senior scientist. His research interests include reliable planning and learning under uncertainties, and multi-robot exploration and coordination in unknown environments.



Zhi He received his Bachelor of Engineering with the first class honor in School of GLASGOW College, University of Electronic Science and Technology of China (UESTC) in 2020, and he is currently pursuing his master's degree majoring in Control Science and Engineering at UESTC. His research interest includes learning and planning in uncertain environments, and time-constrained mobile robot navigation.

Wenda Sheng received his Bachelor's degree with the first class honour in School of Automation Engineering, University of Electronic Science and Technology of China (UESTC) in 2020, and he is currently pursuing his master's degree majoring in Control Science and Engineering at UESTC. His research interest includes distributional reinforcement learning for reliable shortest path, and reinforcement learning for safe autonomous navigation in dynamic environments.

Zhiguang Cao received the Ph.D. degree from Interdisciplinary Graduate School, Nanyang Technological University in 2016. He received the B.Eng. degree in Automation from Guangdong University of Technology, Guangzhou, China, and the M.Sc. in Signal Processing from Nanyang Technological University, Singapore, respectively. He was a Research Fellow with the Energy Research Institute @ NTU (ERI@N), a Research Assistant Professor with the Department of Industrial Systems Engineering and Management, National University of Singapore, and a Scientist with the Agency for Science Technology and Research (A*STAR), Singapore. He joins the School of Computing and Information Systems, Singapore Management University, as an Assistant Professor. His research interests focus on learning to optimize (L2Opt).

Yingjie Zhou (M'14) received his Ph.D. degree in the School of Communication and Information Engineering from University of Electronic Science and Technology of China (UESTC), China, in 2013. He is currently an associate professor in the College of Computer Science at Sichuan University (SCU), China. He is the Director of Institute of Network and Intelligent Systems at SCU. He has been a visiting scholar in Department of Electrical Engineering of Columbia University, New York and Bell Laboratories, Murray Hill, New Jersey. His current research interests include network management, behavioral data analysis, and resource allocation. He has served as a Program Vice-Chair of IEEE International Conference on High Performance Computing and Communications (IEEE HPCC), a Local Arrangement Chair of IEEE International Symposium on Broadband Multimedia Systems and Broadcasting (IEEE BMSB), and the TPC member for various IEEE conferences.

Weinan Gao received the B.Sc. degree in Automation from Northeastern University, Shenyang, China, in 2011, the M.Sc. degree in Control Theory and Control Engineering from Northeastern University, Shenyang, China, in 2013, and the Ph.D. degree in Electrical Engineering from New York University, Brooklyn, NY, USA in 2017. He is currently a Professor with the State Key Laboratory of Synthetical Automation for Process Industries at Northeastern University, Shenyang, China. Previously, he was an Assistant Professor of Mechanical and Civil Engineering at Florida Institute of Technology, Melbourne, FL, USA in 2020-2022, an Assistant Professor of Electrical and Computer Engineering at Georgia Southern University, Statesboro, GA, USA in 2017-2020, and a Visiting Professor of Mitsubishi Electric Research Laboratory (MERL), Cambridge, MA, USA in 2018. His research interests include reinforcement learning, adaptive dynamic programming (ADP), optimal control, cooperative adaptive cruise control (CACC), intelligent transportation systems, sampled-data control systems, and output regulation theory. He is the recipient of the best paper award in IEEE Data Driven Control and Learning Systems Conference (DDCLS) in 2023, IEEE International Conference on Real-time Computing and Robotics (RCAR) in 2018, and the David Goodman Research Award at New York University in 2019. Dr. Gao is an Associate Editor of IEEE Transactions on Neural Networks and Learning Systems, IEEE/CAA Journal of Automatica Sinica, Control Engineering Practice, Neurocomputing, and IEEE Transactions on Circuits and Systems II: Express Briefs, a member of Editorial Board of Neural Computing and Applications, and a Technical Committee member in IEEE Control Systems Society on Nonlinear Systems and Control and in IFAC TC 1.2 Adaptive and Learning Systems.

APPENDIX A
NOTATION SYMBOLS

TABLE VII
LIST OF MAJOR NOTATIONS USED IN THE PAPER

Notations	Descriptions
$\mathcal{G}(\mathscr{N},\mathscr{E})$	The stochastic transportation network
i,j,o,d	Node index, with o representing the origin, d representing the destination, $i,j,o,d\in\mathscr{N}$
ij	Edge index, $ij\in\mathscr{E}$
c_{ij}	Edge ij's travel time (an RV)
$\mathscr{P}_{od}$	Complete set of feasible paths connecting o to d
$c_{\boldsymbol{x}}$	Path x's total travel time (an RV)
T	The pre-departure budget from node o
T_i	The remaining budget at node i
$\mathcal{S}$	RL agent's state space
$\mathcal{A}$	RL agent's action space
$\mathbb{P}[s'\|s,a]$	State-transition probability
$\mathbb{P}[r\|s,a]$	Reward function
$\tau,\tau^{(j)}$	One sampled trajectory (on-policy trajectory)
$R(\tau^{(j)})$	Total travel time of $\tau^{(j)}$
$\pi_{\boldsymbol{\theta}}$	The routing policy, *i.e.*, the actor/policy network
$v^{\pi}_{\boldsymbol{\phi}}$	The value function, *i.e.*, critic network's output
$G^{\pi_{\boldsymbol{\theta}}}(s_k,a_k)$	The estimated cumulative reward from $(s_k,\,a_k)$
$t^{\pi}(o,d)$	Total travel time (an RV) following π from o to d
$J_s(\pi_{\boldsymbol{\theta}})$	Evaluation policy's on-time arrival probability
$b^{\pi}_{\boldsymbol{\phi}}(s)$	Estimated on-time arrival probability for π at s
$\mu_{\boldsymbol{\theta}}$	The behavior policy (for data generation)
$\tau^{(j)}_{\mu}$	The j^{th} sampled trajectory from behavior policy $\mu_{\boldsymbol{\theta}}$
H_j	The length of the sampled trajectory $(\tau^{(j)}_{\mu})$
$\bar{H}$	The averaged length of the sampled trajectories
$\rho(\tau)$	The importance sampling ratio of π to μ w.r.t. τ
$\tilde{\rho}(\tau)$	The weighted importance sampling ratio w.r.t. τ
M	Total number of collected trajectories from π or μ
K	Mini-batch sample size
ψ	One-hot embedding function
φ	Real-valued embedding function
$\tilde{d}$	The dimension of real-valued node embedding
T_{max}	max. training episodes of SEGAC

APPENDIX B
PROOF OF THEOREM 1

Proof.

$$\begin{aligned}\nabla_{\boldsymbol{\theta}} J_s(\pi_{\boldsymbol{\theta}}) &= \nabla_{\boldsymbol{\theta}}\mathbb{P}[t^{\pi}(o,d)\le T]\\ &= \nabla_{\boldsymbol{\theta}}\int_{\tau}\mathbb{P}[R(\tau)\le T]\pi_{\boldsymbol{\theta}}(\tau)\,d\tau\\ &= \int_{\tau}\mathbb{P}[R(\tau)\le T]\nabla_{\boldsymbol{\theta}}\pi_{\boldsymbol{\theta}}(\tau)\,d\tau\\ &= \int_{\tau}\mathbb{P}[R(\tau)\le T]\frac{\nabla_{\boldsymbol{\theta}}\pi_{\boldsymbol{\theta}}(\tau)}{\pi_{\boldsymbol{\theta}}(\tau)}\pi_{\boldsymbol{\theta}}(\tau)\,d\tau\\ &= \int_{\tau}\mathbb{P}[R(\tau)\le T]\nabla_{\boldsymbol{\theta}}\log\pi_{\boldsymbol{\theta}}(\tau)\pi_{\boldsymbol{\theta}}(\tau)\,d\tau\\ &= \mathbb{E}_{\tau\sim\pi_{\boldsymbol{\theta}}}[\mathbb{P}[R(\tau)\le T]\nabla_{\boldsymbol{\theta}}\log\pi_{\boldsymbol{\theta}}(\tau)].\end{aligned}$$

□

In the proof process, $\pi_{\boldsymbol{\theta}}(\tau)$ refers to the probability that trajectory τ is generated by $\pi_{\boldsymbol{\theta}}$.

APPENDIX C
PROOF OF THEOREM 2

Proof.

$$\begin{aligned}\nabla_{\boldsymbol{\theta}} J_s(\pi_{\boldsymbol{\theta}}) &= \nabla_{\boldsymbol{\theta}}\mathbb{P}[t^{\pi}(o,d)\le T]\\ &= \nabla_{\boldsymbol{\theta}}\int_{\tau}\mathbb{P}[R(\tau)\le T]\pi_{\boldsymbol{\theta}}(\tau)\,d\tau\\ &= \int_{\tau}\mathbb{P}[R(\tau)\le T]\nabla_{\boldsymbol{\theta}}\pi_{\boldsymbol{\theta}}(\tau)\,d\tau\\ &= \int_{\tau}\mathbb{P}[R(\tau)\le T]\frac{\nabla_{\boldsymbol{\theta}}\pi_{\boldsymbol{\theta}}(\tau)}{\pi_{\boldsymbol{\theta}}(\tau)}\frac{\pi_{\boldsymbol{\theta}}(\tau)}{\mu_{\boldsymbol{\theta}}(\tau)}\mu_{\boldsymbol{\theta}}(\tau)\,d\tau\\ &= \int_{\tau}\mathbb{P}[R(\tau)\le T]\nabla_{\boldsymbol{\theta}}\log\pi_{\boldsymbol{\theta}}(\tau)\rho(\tau)\mu_{\boldsymbol{\theta}}(\tau)\,d\tau\\ &= \mathbb{E}_{\tau\sim\mu_{\boldsymbol{\theta}}}[\mathbb{P}[R(\tau)\le T]\big(\nabla_{\boldsymbol{\theta}}\log\pi_{\boldsymbol{\theta}}(\tau)\big)\rho(\tau)].\end{aligned}$$

□