IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received October 7, 2019, accepted October 22, 2019, date of publication October 29, 2019, date of current version November 11, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2950055

Reinforcement Learning Based Stochastic Shortest Path Finding in Wireless Sensor Networks

WENWEN XIA, CHONG DI, HAONAN GUO, AND SHENGHONG LI
School of Cyber Security, Shanghai Jiao Tong University, Shanghai 200240, China

Corresponding author: Shenghong Li (shli@sjtu.edu.cn)

This work was supported by the National Natural Science Foundation of China under Grant 61971283.

ABSTRACT Many factors influence the connection states between nodes of wireless sensor networks, such as physical distance, and the network load, making the network's edge length dynamic in abundant scenarios. This dynamic property makes the network essentially form a graph with stochastic edge lengths. In this paper, we study the stochastic shortest path problem on a directional graph with stochastic edge lengths, using reinforcement learning algorithms. we regard each edge length as a random variable following unknown probability distribution and aim to find the stochastic shortest path on this stochastic graph. We evaluate the performance of path-finding algorithms using regret, which represents the cumulative reward difference between the practical path-finding algorithm and the optimal strategy that chooses the global stochastic shortest path every time. We model the path-finding procedure as a Markov decision process and propose two online path-finding algorithms: Q_{SSP} algorithm and $SARSA_{SSP}$ algorithm, both combined with specifically-devised average reward mechanism. We justify the convergence property and correctness of the proposed algorithms theoretically. Experiments conducted on two benchmark graphs illustrate the superior performance of the proposed Q_{SSP} algorithm which outperforms the $SARSA_{SSP}$ algorithm and other competitive algorithms about the regret metric.

INDEX TERMS Stochastic shortest path finding, reinforcement learning, Q-learning, SARSA, convergence proof.

I. INTRODUCTION

Wireless sensor networks (WSN) is an autonomous sensor network with spatially dispersed sensors harvesting the ambient environmental information. The environmental information is mostly transmitted by the distributed sensors in a hop-by-hop manner to a centralized station wirelessly, which processes the gathered information and dispatches control instructions [1]. WSN has been widely applied in various areas, including area monitoring [2], health care monitoring [3], [4], and earth sensoring [4], [5]. Despite regular WSN application, there are a large number of other applications, including battlefield probing, critical region surveillance, and emergency rescue.

In order to support these complicated and critical applications, sensors have to operate reliably with limited energy, bandwidth, and computational resources. Furthermore, the resource-limited sensors must be extensively efficient to ensure low time-delay and long network lifetime. To meet the quality of service requirements in these scenarios, one of the critical issues in WSNs is the data transmission scheme, i.e., the data relay paths. There have been efforts aiming to tackle these challenges by combining WSNs and Software-defined networking (SDN) [6], [7]. SDN [8] is a new networking paradigm which separates the forwarding hardware and the control decisions [9]. Therefore, the upper control plane can dispatch dynamic routing rules to different network devices, endowing the network with programmability and flexibility. The flexibility and dynamics provide novel insight on data transmission schemes for WSNs. Therefore, the data transmission strategy, i.e., the data path selection issue is particularly important.

Machine learning algorithms have been progressively popular for data path selection in WSNs in the past

The associate editor coordinating the review of this manuscript and approving it for publication was Qilian Liang.

decades [10]–[13]. Many protocols primarily utilize Reinforcement Learning (RL) to learning a path selection policy [10], [11], [14], [15]. QELAR [14] and QL-EEBDG [15] both use Q-learning algorithm [16], a classic RL algorithm, to choose the data routing path. They take into account the forwarder node's residual energy and energy distribution of its adjacent nodes in the reward function design, to force the network nodes' residual energy evenly distributed and prolong the whole network lifetime. FROMS [11] is designed for multi-sink data path selection using each node's local information as feedback to its neighbor nodes to update the Q function. However, most of these methods regard the single hop length or cost as a constant, i.e., static, and merely mix one node's extra information in the reward function design. While in abundant practical scenarios, the hop length or network graph edge length itself is not a constant but acts as a random variable, i.e., dynamic. This random variable can be influenced by the location of nodes, the randomness of demand [17], and more commonly, the network load. The dynamics of network connections essentially form a stochastic graph, i.e., a graph with dynamic edges.

Under this circumstance, the important issue becomes finding the stochastic shortest path (SSP) on a stochastic graph, i.e., finding a path with minimal expected total length on a stochastic graph. A stochastic graph has dynamic, instead of static, edge lengths. The traditional shortest path is the foundation of many path selection strategys [18], [19]. However, selecting the same path may lead to congestion or hotpots. We can regard the time delay between two nodes as the ideal dynamic edge length, instead of their physical distance. Under this perspective, once congestion happens on a specific path, it is not the stochastic shortest path anymore, and the stochastic shortest path solution will choose a different path. Hence the solution for the stochastic shortest path problem still works and is especially valuable. Furthermore, the solution of this problem can also be applied in other areas, including emergency response [20], robot navigation [21], and road networks [22]. Therefore, the solution of SSP problem is advantageous in diverse areas, including the wireless sensor networks.

A. RELATED WORK

Frank [23] and Pritsker [24] primarily analyzed the stochastic shortest path problem. Their analyses leverage multiple integrals to represent the probabilistic quantities, necessitating numerical evaluation to estimate the integrals. To overcome the fast-growing evaluation cost, Mirchandani [25] introduced the reliability computation, measuring the connectivity of each link. However, this method suffers the tight constraint that it assumes the edge lengths to be discrete random variables. Fishman [26], Adlakha [27], and Sigal *et al.* [28] adopted Monte-Carlo simulation to estimate the probabilistic quantities. However, the simulation procedure is computationally inefficient.

Also, the methods mentioned above require the prior knowledge of edge length distributions of a stochastic graph, which is not applicable for numerous practical scenarios. Beigy and Meybodi [29] proposed to construct a distributed learning automata (LA) network to find the stochastic shortest path for packet routing. Guo *et al.* [30] improved this LA-based method by modifying the distributed LA architecture to its hierarchical counterpart and adopted the corresponding hierarchical convergence criteria. Liu and Zhao [31] proposed to adopt whole path feedback and devised a forced exploration algorithm in which a random barycentric spanner is used for exploration. The methods mentioned above use bandit feedback (feedback for the whole path) to improve the packet path strategy. He *et al.* [32] chose to use semi-bandit feedback (feedback of each edge) for path planning. This method uses one path for exploitation and possibly another path for exploration. Talebi *et al.* [17] proposed to model the path planning process as a Markov decision process (MDP), which coincides with our model in this paper, and propose the KL-Hop-by-Hop Routing (KL-HHR) algorithm. However, their method has to combine with line search and Bellman-Ford algorithm to choose the next node for a path. Besides, the combinatorial upper confidence bound (CUCB) algorithm [33] and Thompson sampling (TS) algorithm [34] can also be used for this problem. The CUCB algorithm is used initially to solve the multi-armed bandit (MAB) problem, and it takes several simple arms to form a super arm to interact with the environment. The TS algorithm maintains parameters of a Beta distribution for each possible action. It generates a value according to the maintained Bata distribution and chooses the best action according to the generated values each time. For this problem, we can regard the possible paths as actions and then use these two algorithms to solve the SSP problem.

B. CONTRIBUTION OF THE PAPER

In this paper, we treat the stochastic shortest path problem from the reinforcement learning (RL) perspective, and propose two counterpart algorithms, the off-policy Q_{SSP} algorithm and the on-policy $SARSA_{SSP}$ algorithm. The reward function is crucial for an RL algorithm to learn a feasible policy. Hence we specifically devise the reward mechanism using a simple while useful trick which stabilizes and facilitates the learning process. The proposed algorithms converge fast and has low regret w.r.t number of transmissions, compared with competitive algorithms.

We analyze the convergence property of the proposed algorithms, showing that there exists some *appropriate* interval for the discounted factor γ of the RL algorithms, during which the algorithms will converge w.p.1 and find the optimal stochastic shortest path.

We conduct experiments on two benchmark graphs, revealing that the proposed algorithms converge significantly fast and are insensitive to the initial learning rate parameter. Furthermore, compared with other algorithms for the stochastic shortest path problem, the proposed Q_{SSP} algorithm outperforms $SARSA_{SSP}$ and other algorithms.

The remaining of this paper is organized as follows. In section 2, we review the preliminaries related to stochastic graph and RL and formulate the SSP problem. The proposed algorithms and convergence proof lie in section 3. Section 4 illustrates the experimental results and performance analysis. Section 5 concludes this paper and provides several future research directions.

II. PRELIMINARIES AND PROBLEM FORMULATION

A. STOCHASTIC GRAPH

A stochastic graph can be represented by a triple $G = (V, E, \mathcal{F})$, where $V = \{1, 2, \cdots, n\}$ represents the node set, $E \subset V \times V$ denotes the edge set, and $\mathcal{F}_{n\times n}$ indicates the probability distribution of each edge length. Note that for a general stochastic graph, there may exist both uni-directional and bi-directional edges. The length $L(v_i, v_j)$ of edge (v_i, v_j) is considered as an random variable with the probability density function f_{ij}, stored in $\mathcal{F}_{ij}$.

B. REINFORCEMENT LEARNING

Reinforcement learning has illustrated great potential in various challenging tasks, including game playing [35]–[37], real-time ads bidding [38], neural network architecture searching [39], [40] and iterative recommender systems [41], [42]. Reinforcement learning is conventionally modeled as a Markov decison process (MDP) [43]. MDP is described as a tuple $(\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma)$. $\mathcal{S}$ is the state space while $\mathcal{A}$ is the action space. $\mathcal{P} : \mathcal{S}\times\mathcal{A}\times\mathcal{S} \rightarrow [0, 1]$ is the state transition function. $\mathcal{R} : \mathcal{S}\times\mathcal{A}\times\mathcal{S} \rightarrow \mathbb{R}$ is the reward function determined by the environment. γ is the discounted factor discounting the future rewards. At each timestep, the RL agent will choose an action to interact with the environment following the policy π_θ according to current state, i.e., $a_t = \pi_\theta(s_t)$, then the environment will give the agent an immediate reward feedback r_t, and transfer to the next state s_{t+1}. The interaction procedure is depicted in figure1. The goal of the agent is to find an optimal policy π^* to maximize the expected cumulative reward from any state $s \in \mathcal{S}$, i.e., $\pi^* = \max_{\pi_\theta} V^{\pi_\theta}(s)$, where $V^{\pi_\theta}(s) = \mathbb{E}_{\pi_\theta}[\sum_{k=0}^{\infty} \gamma^k r_{t+k} | s_t = s]$ is the state value function of policy π_θ, representing the expected cumulative discounted reward from state s, following policy π_θ. Equivalently, the optimal policy can be defined by the state-action value function, i.e. Q function: $\pi^* = \max_{\pi_\theta} Q(s, a)$ at any given state s after choosing an action a, where $Q^{\pi_\theta}(s, a) = \mathbb{E}_{\pi_\theta}[\sum_{k=0}^{\infty} \gamma^k r_{t+k} | s_t = s, a_t = a]$, measuring the expected cumulative discounted reward at state s after choosing action a then following policy π_θ.

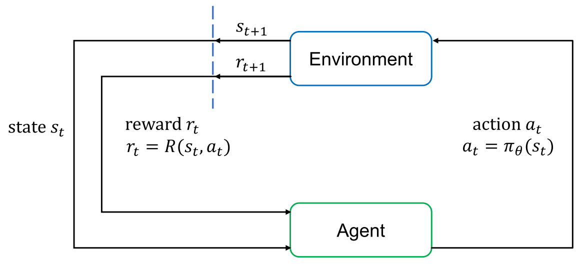


FIGURE 1. The RL interaction sketch.

C. STOCHASTIC SHORTEST PATH PROBLEM FORMULATION

The goal of stochastic shortest path problem is to find a path on a stochastic graph with minimal expacted total length. A stochastic path, indexed by i, on a stochastic graph from node v_s to node v_d can be represented by $p_i = \{v_{i,1}(v_s), v_{i,2}, \cdots, v_{i,n_i}, v_{i,n_i+1}(v_d)\}$, with n_i edges, and $p_{i,j}$ is the index of the jth edge on path p_i, i.e. edge $(v_{i,j}, v_{i,j+1})$. Each stochastic path p_i has an expected length $\sum_{j=1}^{n_i} e_{p_{i,j}}$, where $e_{p_{i,j}}$ denotes the expected length of edge $p_{i,j}$, i.e., $e_{p_{i,j}} = \mathbb{E}[L(v_{i,j}, v_{i,j+1})]$. Finding the stochastic shortest path can be formalized as finding the optimal solution p^* in path space $P = \{p_i, i = 1, 2, 3 \cdots k\}$, such that $p* = \min_{p_i} \sum_{j=1}^{n_i} e_{p_{i,j}}$.

III. REINFORCEMENT LEARNING BASED SSP ALGORITHMS

A. PROBLEM MODELING AND ALGORITHMS

We model the stochastic shortest path-finding process as an MDP. RL updating rules are adopted by the algorithm/agent to learn a feasible value function. Specifically, the agent chooses an action to interact with the stochastic graph (i.e., the environment) to arrive at the next node periodically, to obtain a path from the source node to the destination node, aiming to maximize the expected cumulative reward given by the stochastic graph environment. We describe the MDP as follows:

- **States** $\mathcal{S}$. Each node on the stochastic graph is model as a state. The current state is the current node in the path-finding procedure on the stochastic graph. The possible transition states of a state (node) consist of its out-link neighbor nodes.
- **Actions** $\mathcal{A}$. The action set of a state is formed by its out-link neighbor nodes, which is identical to its transition state set. Note that different nodes have diverse adjacent nodes, thereby different states have unequal action sets. This property differs from traditional MDPs, in which all states have identical action sets.
- **Transitions** $\mathcal{P}$. As we model the state as the current node on the stochastic graph, once the agent samples an action, the next state is naturally determined by the action chosen, i.e., we have a deterministic state transformation function.
- **Reward** $\mathcal{R}$. We set the reward for a state-action pair (s, a) as the negative length sampled on edge (s, a) of the stochastic graph. The stochasticity of edge length means we have a stochastic reward function.

- **Discounted factor** γ. The discounted factor $\gamma \in [0, 1]$ is used to discount future rewards. $\gamma = 0$ means the agent considers the immediate reward exclusively, while $\gamma = 1$ reveals the agent regards future rewards equivalently important as the immediate reward.

To find the stochastic shortest path, the agent initially observes the start state s_0, i.e., the source node, and then chooses an action following the current policy, until the agent arrives at the final state (the destination node) or a state at which there have no selectable actions. As described above, the action set consists of all the selectable contiguous outlink nodes of the current state. Thereby, the state transition is deterministic, i.e., the next state equals the action chosen by the agent at the current state.

The state space and action space are naturally discrete in this problem, making tabular-based and value-based methods more appropriate. For value-based methods, the state-value function can be evaluated by Monte-Carlo simulation or TD methods. Monte Carlo methods can only update the state-value function and the policy in an episode-by-episode sense, i.e., only after the completion of an episode. Hence, we adopt the TD methods to update the value functions, which update the value function after each interaction. We propose to devise Q-learning [16] and SARSA [43] based algorithms for the SSP problem. They are two representative TD methods, updating the Q function in an off-policy and an on-policy manner, respectively. The update rule of Q-learning is:

$$Q(S_t, A_t) = Q(S_t, A_t) + \alpha_t (R_t + \gamma \max_a Q(S_{t+1}, a) - Q(S_t, A_t)) \tag{1}$$

At each state S_t, Q-learning chooses an action A_t following its behavioral policy and updates the Q function using a greedy policy, i.e., choosing an action with the maximal state-action value at state S_{t+1}. While SARSA firstly determines an action A_{t+1} at state S_{t+1}, and then uses the A_{t+1} for both updating the Q function and interacting with the environment at next timestep. The update rule of SARSA is:

$$Q(S_t, A_t) = Q(S_t, A_t) + \alpha_t (R_t + \gamma Q(S_{t+1}, A_{t+1}) - Q(S_t, A_t)) \tag{2}$$

Another central concern for RL is the balance between exploration and exploitation. We adopt the standard ϵ-greedy policy for the behavioral policy of Q-learning and the SARSA policy, and the greedy policy for Q-learning's target policy. Formula3 describes the ϵ-greedy policy strategy, i.e., we choose the optimal action with probability $1-\epsilon$ and a random action with probability ϵ. We set the epsilon to 0.1 for both algorithms. Note that the stochastic shortest path should be acyclic, i.e., there are no duplicate nodes in the optimal path. Hence, the nodes appearing in the path should not be chosen as actions at any future state at all. The action set for every state will dynamically shrink during the interactions between the agent and the stochastic graph.

$$\pi(S) = \begin{cases} argmax_a Q(S, a), & 1-\epsilon \\ \forall a, & \epsilon \end{cases} \tag{3}$$

More importantly, the ϵ-greedy ensures that each state-action pair will be visited with infinite times, i.e., every state-action pair will be updated infinitely often, which is necessary for the convergence of the Q-table iteration [44].

The reward function design is specifically crucial for the RL agent. Because different rewards stimulate RL agent to learn different policies. As mentioned above, under this problem modeling, the reward given by the environment is set to be the negative length sampled on the stochastic graph on the corresponding edge. Considering the stochasticity of the edge length, directly using the negative length as the reward feedback for the agent leads to instability on the value updating stage, as the reward for an identical state-action pair varies at different timesteps, leading to inconstant update directions. Thus, we introduce a simple while efficient trick, phrased as *reward-averaging*, which stabilizes the received reward, and facilitates the convergence. Reward-averaging maintains the average historical reward for each state-action pair $R_{avg}(s, a)$, and $Num(s, a)$ represents the number that (s, a) has been chosen. After receiving a reward feedback r_t from the environment at timestep t, for the state-action pair (s_t, a_t), the new average reward $R_{avg}^{new}(s_t, a_t)$ will be calculated according to the $R_{avg}(s, a)$ and r_t via Eq.4, and then the $R_{avg}^{new}(s_t, a_t)$ is regarded as R_t in Eq.1 and Eq.2. Finally, the $R_{avg}(s_t, a_t)$ is updated by $R_{avg}^{new}(s_t, a_t)$. Experiments have justified the effectiveness of this technique.

$$R_{avg}^{new}(s_t, a_t) = \frac{Num(s_t, a_t) \times R_{avg}(s_t, a_t) + r_t}{Num(s_t, a_t) + 1} \tag{4}$$

The learning rate α_t is an hyper-parameter that affects the convergence speed sharply. More importantly, it influences the convergence property of the algorithm. To ensure the convergence of the Q table, the α_t has to satisfy $\sum_t \alpha_t = \infty$ and $\sum_t \alpha_t^2 < \infty$, and we will discuss this in detail in sectionIII-B. Hence, we adopt α_t satisfying the properties mentioned above as follows

$$\alpha_t = \begin{cases} \alpha_0, & t \leq 100 \\ \frac{1}{\lceil \frac{1}{\alpha_0} + \frac{t-100}{10} \rceil}, & t > 100 \end{cases} \tag{5}$$

The parameter γ is another crucial hyper-parameter that affects the performance and correctness of the algorithm. In this problem, $\gamma < 1$ means we discount the future edge lengths when estimating the expected total length of a path. However, future edges lengths contribute equally with the immediate edge length for a specific path. Hence we should set $\gamma = 1$ naturally. In this case, the $\max_a Q^*(v_s, a)$ indicates the negative shortest expected length. In other words, under the $\gamma = 1$ setting, $|Q^*(x, a)|$ represents the shortest expected length from node x to destination if choosing next node as a, as is shown in formula6. Hence the greedy strategy ensures the correct stochastic shortest path under the optimal Q^*.

$$|Q^*(x,a)| = \mathbb{E}\left[\sum_{s=x}^{destination} |R(s,\pi(s))|\right] \tag{6}$$

However, we find that, for some specific $\gamma < 1$, the greedy strategy could still obtains the correct path. We define the *appropriate* γ to be the $\gamma \in [0, 1)$ that still ensures the greedy strategy obtains the correct stochastic shortest path under Q^*_γ, i.e., if

$$\sum_{i=1}^{n_m} e_{p_{m,i}} \leq \sum_{i=1}^{n_k} e_{p_{k,i}}, \quad \forall k \in \mathbb{N}^+,\ k \leq N \tag{7}$$

where p_m is the stochastic shortest path. The appropriate γ guarantees the following inequation.

$$\sum_{i=1}^{n_m}(\gamma)^{i-1} e_{p_{m,i}} \leq \sum_{i=1}^{n_k}(\gamma)^{i-1} e_{p_{k,i}}, \quad \forall k \in \mathbb{N}^+,\ k \leq N \tag{8}$$

where N is the number of total loop-free paths, and $e_{p_{k,j}}$ denotes the expected length of jth edge of kth path p_k.

Since we have the appropriate $\gamma < 1$, we can prove that the Q table, according to the update rule1 or 2 converges to the optimal Q^*_γ, with the infinite state-action pair visitation guarantee. Moreover, the greedy policy with Q^*_γ finds the correct stochastic shortest path. We prove and compute a precise interval of the appropriate γ in sectionIII-B.

Algorithm 1 The Q$_{SSP}$ Algorithm

```
1: Initialize the Q table, Q$_{Counter}$ table, Q$_{MeanReward}$ table,
   according to the stochastic graph
2: Initialize the path set $\Phi$
3: T = $v_d$
4: episode = 0
5: for episode<EpisodeNum do
6:   S = $v_s$
7:   $\Phi$ = S
8:   while S $\neq$ T do
9:     ActionSet = adjecent nodes of node S
10:    U = ActionSet/$\Phi$
11:    if U is empty set then
12:      break
13:    end if
14:    A = $\epsilon$-greedy(S) in U
15:    R, S$'$ = Env.Step(S, A)
16:    TotalReward = $R^{new}_{avg}(S,A) \times Num(S,A)$+R
17:    TotalNum = $Num(S,A) + 1$
18:    R$_{avg}$ = TotalReward/TotalNum
19:    $R^{new}_{avg}(S,A)$ = R$_{avg}$
20:    $Num(S,A)$ = TotalNum
21:    Q(S, A) = Q(S, A) + $\alpha$(R$_{avg}$ + $\gamma$ max$_a$Q(S$'$, a) -
       Q(S, A))
22:    S = S$'$
23:    $\Phi = \Phi \cup$ S
24:  end while
25:  episode = episode + 1
26: end for
27: FinalPath = greedy path w.r.t (Q, $v_s$, $v_d$)
28: return FinalPath
```

Algorithm 2 The SARSA$_{SSP}$ Algorithm

```
1: Initialize the Q table, Q$_{Counter}$ table, Q$_{MeanReward}$ table,
   according to the stochastic graph
2: Initialize the path set $\Phi$
3: T = $v_d$
4: episode = 0
5: for episode<EpisodeNum do
6:   S = $v_s$
7:   $\Phi$ = S
8:   ActionSet = adjecent nodes of node S
9:   U = ActionSet/$\Phi$
10:  A = $\epsilon$-greedy(S) in U
11:  while S $\neq$ T do
12:    R, S$'$ = Env.Step(S, A)
13:    ActionSet = adjecent nodes of node S$'$
14:    U = ActionSet/$\Phi$
15:    if U is empty set then
16:      break
17:    end if
18:    A$'$ = $\epsilon$-greedy(S$'$) in U
19:    TotalReward = $R^{new}_{avg}(S,A) \times Num(S,A)$+R
20:    TotalNum = $Num(S,A) + 1$
21:    R$_{avg}$ = TotalReward/TotalNum
22:    $R^{new}_{avg}(S,A)$ = R$_{avg}$
23:    $Num(S,A)$ = TotalNum
24:    Q(S, A) = Q(S, A) + $\alpha$(R$_{avg}$ + $\gamma$Q(S$'$, A$'$) - Q(S,
       A))
25:    S = S$'$
26:    A = A$'$
27:    $\Phi = \Phi \cup$ S
28:  end while
29:  episode = episode + 1
30: end for
31: FinalPath = $\Phi$
32: return FinalPath,
```

Based on the analysis and settings mentioned above, we propose two practical algorithms: the Q$_{SSP}$ algorithm, and the SARSA$_{SSP}$ algorithm. The details of Q$_{SSP}$ algorithm (resp. SARSA$_{SSP}$ algorithm) are shown in Algo.1 (resp. Algo.2). Note that the environment modeling is merely used for experimental simulations, while the algorithms operate in an online and model-free manner.

B. CONVERGENCE AND CORRECTNESS OF PROPOSED ALGORITHMS

Since the Q$_{SSP}$ updates the Q table in an off-policy manner, which makes the convergence proof more easy to handle, in this section we will prove the convergence and correctness of the proposed Q_{SSP} algorithm with an appropriate γ and prove the existence of such an appropriate γ.

1) CONVERGENCE OF Q$_{SSP}$ ALGORITHM WITH $\gamma < 1$

Theorem 1: A random process $\Delta_{t+1}(x) = (1-\alpha_t(x))\Delta_t(x) + \alpha_t(x)F_t(x)$ converges to zero w.p.1 under the following assumptions:

- $0 \le \alpha_t(x) \le 1$, $\sum_t \alpha_t(x) = \infty$, $\sum_t \alpha_t(x)^2 < \infty$;
- $\|\mathbb{E}[F_t(x)|\mathcal{F}_t]\|_W \le \gamma\|\Delta_t\|_W$, with $\gamma < 1$, $\|\cdot\|_W$ is some weighted maximum norm, and $\mathcal{F}_t$ denotes the past information $\{X_{n-1},\cdots;F_{n-1},\cdots;\alpha_{n-1},\cdots\}$.
- $\mathbf{var}[F_t(x)|\mathcal{F}_t] \le C(1+\|\Delta_t\|_W^2)$, for $C > 0$.

Proof: The proof can be found in [44] ■

Theorem 2: For the proposed Q$_{SSP}$ algorithm, the Q table converges to the optimal Q table with $\gamma < 1$.

Proof: The proof here is similar with that in [45], however, [45] adopts a stochastic state transformation function setting and a deterministic reward function assumption. While the MDP for the SSP problem has a deterministic state transformation function and a stochastic reward function, which can not be directly tackled by the methods of [45].

In the SSP problem, the reward $r(x, a, y)$ at timestep t is regraded as a bounded random varibale $R(x, a)$ (the reward is only releated with x and a, i.e., the edge (x, a), hence y can be ignored). Defining operator $\mathcal{T}$ as

$$\begin{aligned}\mathcal{T}Q(x,a) &= \sum_{y\in\mathcal{X}} P_a(x,y)(r(x,a,y)+\gamma\max_{b\in\mathcal{A}}Q(y,b)) \\ &= \mathbb{E}[R(x,a)] + \gamma\max_{b\in\mathcal{A}}Q(a,b) \qquad (9)\end{aligned}$$

Then we have:

$$\begin{aligned}\|\mathcal{T}Q_1-\mathcal{T}Q_2\|_\infty &= \max_{x,a}\|\mathbb{E}[R_1(x,a)]+\gamma\max_{b\in\mathcal{A}}Q_1(a,b) \\ &\quad -\mathbb{E}[R_2(x,a)]-\gamma\max_{b\in\mathcal{A}}Q_2(a,b)\| \\ &= \max_{x,a}\gamma\|\max_{b\in\mathcal{A}}Q_1(a,b)-\max_{b\in\mathcal{A}}Q_2(a,b)\| \\ &\le \max_{x,a}\gamma\max_{b\in\mathcal{A}}\|Q_1(a,b)-Q_2(a,b)\| \\ &= \gamma\max_{a,b}\|Q_1(a,b)-Q_2(a,b)\| \\ &= \gamma\|Q_1-Q_2\|_\infty \qquad (10)\end{aligned}$$

Accodring to the update rule of formula1:

$$\begin{aligned}Q_{t+1} &= Q_t(x_t,a_t)+\alpha_t(x_t,a_t)(R_t(x_t,a_t) \\ &\quad +\gamma\max_{b\in\mathcal{A}}Q_t(x_{t+1},b)-Q_t(x_t,a_t)) \qquad (11)\end{aligned}$$

We can substract $Q^*(x_t, a_t)$ and let

$$\Delta_t(x,a) = Q_t(x,a)-Q^*(x,a) \qquad (12)$$

then we have

$$\begin{aligned}\Delta_{t+1} &= (1-\alpha_t(x_t,a_t))\Delta_t+\alpha_t(x_t,a_t)(R_t(x_t,a_t) \\ &\quad +\gamma\max_{b\in\mathcal{A}}Q_t(x_{t+1},b)-Q^*(x_t,a_t)) \qquad (13)\end{aligned}$$

And we can write

$$F_t(x,a) = R_t(x,a)+\gamma\max_{b\in\mathcal{A}}Q_t(y,b)-Q^*(x,a) \qquad (14)$$

where y is the next state followed by x after choosing action a.

It's easily to verify that $\{\Delta_t\}$ fits the first assumption of theorem1, according to formula5.

Note that in this problem, the next state is not sampled from a Markov chain but determined by the action a, Hence hence we have:

$$\begin{aligned}\mathbb{E}[F_t(x,a)|\mathcal{F}_t] &= \mathbb{E}[R_t(x,a)+\gamma\max_{b\in\mathcal{A}}Q_t(y,b)-Q^*(x,a)] \\ &= \mathbb{E}[R_t(x,a)]+\gamma\max_{b\in\mathcal{A}}Q_t(a,b)-Q^*(x,a) \\ &= \mathcal{T}Q_t(x,a)-Q^*(x,a) \\ &= \mathcal{T}Q_t(x,a)-\mathcal{T}Q^*(x,a) \qquad (15)\end{aligned}$$

The last equation is because for the optimal $Q^*(x, a)$, we have $\mathcal{T}Q^*(x,a) = Q^*(x,a)$.

Finally we have

$$\begin{aligned}\|\mathbb{E}[F_t(x,a)|\mathcal{F}_t]\|_\infty &= \|\mathcal{T}Q_t(x,a)-\mathcal{T}Q^*(x,a)\|_\infty \\ &\le \gamma\|Q_t(x,a)-Q^*(x,a)\|_\infty \\ &= \gamma\|\Delta_t(x,a)\|_\infty \qquad (16)\end{aligned}$$

Note that the maximum norm $\|\cdot\|_\infty$ is a special case of $\|\cdot\|_W$. Hence the $\{\Delta_t\}$ fits the second assumption in theorem1

According to formula22, the exptectation of $F_t(x, a)$ under $\mathcal{F}_t$ is $\mathcal{T}Q_t(x,a) - Q^*(x,a)$, hence the variance of $F_t(x, a)$ under $\mathcal{F}_t$ is

$$\begin{aligned}&var[F_t(x,a)|\mathcal{F}_t] \\ &\quad = \mathbb{E}[(R_t(x,a)+\gamma\max_{b\in\mathcal{A}}Q_t(y,b)-Q^*(x,a) \\ &\qquad -\mathcal{T}Q_t(x,a)+Q^*(x,a))^2] \\ &\quad = \mathbb{E}[(R_t(x,a)+\gamma\max_{b\in\mathcal{A}}Q_t(a,b)-\mathcal{T}Q_t(x,a))^2] \qquad (17)\end{aligned}$$

The $\gamma\max_{b\in\mathcal{A}}Q_t(a,b) - \mathcal{T}Q_t(x,a)$ term is constant and $R_t(x,a)$ is bounded, hence we have $var[F_t(x,a)|\mathcal{F}_t] < \infty$, e.g. there exists a constant C, such that

$$var[F_t(x,a)|\mathcal{F}_t] \le C(1+\|\Delta_t\|_W^2) \qquad (18)$$

holds. Eq.18 shows that $\{\Delta_t\}$ fits the third assumption of theorem1.

According to theorem1, $\{\Delta_t\}$ converges to zero w.p.1, revealing that Q_t converges to the optimal Q^* w.p.1, for the Q$_{SSP}$ algorithm. ■

2) CONVERGENCE OF SARSA$_{SSP}$ ALGORITHM WITH $\gamma < 1$

Theorem 3: For the proposed SARSA$_{SSP}$ algorithm, the Q table converges to the optimal Q table with $\gamma < 1$.

Proof: The convergence proof of SARSA$_{SSP}$ algorithm is similar with that of the Q$_{SSP}$ algorithm, as the SARSA$_{SSP}$ differs from Q$_{SSP}$ about the action adopted for the Q table update. We will show that these three assumptions of theorm1 still hold for the SARSA$_{SSP}$ algorithm.

According to the update rule of formula2,

$$\begin{aligned}Q_{t+1} &= Q_t(x_t,a_t)+\alpha_t(x_t,a_t)(R_t(x_t,a_t) \\ &\quad +\gamma Q_t(x_{t+1},a_{t+1})-Q_t(x_t,a_t)) \qquad (19)\end{aligned}$$

and the definition of Δ_t (Eq.12) in the previous section, we modify the definition of Δ_{t+1} as

$$\Delta_{t+1} = (1-\alpha_t(x_t,a_t))\Delta_t + \alpha_t(x_t,a_t)(R_t(x_t,a_t) + \gamma Q_t(x_{t+1},a_{t+1}) - Q^*(x_t,a_t)) \quad (20)$$

and modify the definition of $F_t(x,a)$ as

$$F_t(x,a) = R_t(x,a) + \gamma Q_t(y,b) - Q^*(x,a) \quad (21)$$

Recall the definition of opeator $\mathcal{T}$ and $\mathbb{E}[F_t(x,a)|\mathcal{F}_t]$ in the previous subsection, we can deduce that

$$\begin{aligned}
&\mathbb{E}[F_t(x,a)|\mathcal{F}_t]\\
&\quad= \mathbb{E}[R_t(x,a) + \gamma Q_t(y,b) - Q^*(x,a)]\\
&\quad= \mathbb{E}[R_t(x,a)] + \gamma \sum_{b\in\mathcal{A}} P(b|a)Q_t(a,b) - Q^*(x,a)\\
&\quad\le \mathbb{E}[R_t(x,a)] + \gamma \max_{b'\in\mathcal{A}} Q_t(a,b') - Q^*(x,a)\\
&\quad= \mathcal{T}Q_t(x,a) - Q^*(x,a)\\
&\quad= \mathcal{T}Q_t(x,a) - \mathcal{T}Q^*(x,a) \qquad (22)
\end{aligned}$$

Similarly, we have that $\|\mathbb{E}[F_t(x,a)|\mathcal{F}_t]\|_\infty \le \|\mathcal{T}Q_t(x,a) - \mathcal{T}Q^*(x,a)\|_\infty \le \gamma\|\Delta_t(x,a)\|_\infty$, i.e. the second assumption of theorem1 holds.

Next, the third assumption holds. The variance of $F_t(x,a)$ under $\mathcal{F}_t$ is

$$\begin{aligned}
var[F_t(x,a)|\mathcal{F}_t] &= var[(R_t(x,a) + \gamma Q_t(y,b) - Q^*(x,a)]\\
&= var[(R_t(x,a) + \gamma Q_t(a,b)] \qquad (23)
\end{aligned}$$

as the $R_t(x,a)$ and $Q_t(a,b)$ are all bounded random variables, hence we have $var[F_t(x,a)|\mathcal{F}_t] < \infty$. Hence, the third assumption of theorem1 holds. Thereby, the $\{\Delta_t\}$ converges to zero w.p.1, revealing that Q_t converges to the optimal Q^* w.p.1, for the SARSA$_{SSP}$ algorithm. ∎

3) THE EXISTENCE OF APPROPRIATE $\gamma < 1$

We can prove that there exists a $\gamma < 1$, such that the algorithm can still find the correct optimal path without violating the $\gamma < 1$ constraint in previous convergence proofs.

Theorem 4: There exists a $\gamma < 1$, such that the discounted stochastic shortest path is precisely the undiscounted stochastic shortest path.

Proof: Without loss of generality, we assume that the path p_1 is the stochastic shortest path, with n_1 edges. And there has total N paths, represented as $\{p_1, p_2, \cdots, p_N\}$. What we need to prove is, if

$$\sum_{i=1}^{n_1} e_{p_{1,i}} \le \sum_{i=1}^{n_k} e_{p_{k,i}} \quad \forall k \in \mathbb{N}^+,\ k \le N \quad (24)$$

where $p_{i,j}$ indicates the index of the j-th edge on path p_i. Then there exists an $\gamma < 1$, such that:

$$\sum_{i=1}^{n_1} \gamma^{i-1} e_{p_{1,i}} \le \sum_{i=1}^{n_k} \gamma^{i-1} e_{p_{k,i}}, \quad \forall k \in \mathbb{N}^+,\ k \le N \quad (25)$$

We define the minimal expected length difference between all paths as Δ.

$$\Delta = \min_{i,j} |\sum_{k=1}^{n_i} e_{p_{i,k}} - \sum_{k=1}^{n_j} e_{p_{j,k}}| \quad (26)$$

It's clear to verify that

$$\lim_{\gamma\to 1} \sum_{i=1}^{n_k} \gamma^{i-1} e_{p_{k,i}} = \sum_{i=1}^{n_k} e_{p_{k,i}}, \quad \forall k \in \mathbb{N}^+,\ k \le N \quad (27)$$

Eq.27 means that, for $\forall\epsilon > 0, \exists\delta > 0$, such that if $\gamma > 1-\delta$, we have

$$\sum_{i=1}^{n_k} e_{p_{k,i}} - \sum_{i=1}^{n_k} \gamma^{i-1} e_{p_{k,i}} = \sum_{i=1}^{n_k} (1-\gamma^{i-1}) e_{p_{k,i}} < \epsilon \quad (28)$$

Letting $\epsilon = \Delta$, then there exists a $\gamma_\Delta < 1$, such that

$$\sum_{i=1}^{n_k} (1-\gamma_\Delta^{i-1}) e_{p_{k,i}} < \Delta, \quad \forall k \in \mathbb{N}^+,\ k \le N \quad (29)$$

Scaling up the number of n_k to N, and denoting the maximum expected length of all edges as e_{max}, we yield a stronger inequality,

$$\sum_{i=1}^{|E|} (1-\gamma_\Delta^{i-1}) e_{max} < \Delta \quad (30)$$

Further, we can scale up the power of γ_Δ, yielding the following inequality

$$\sum_{i=1}^{|E|} (1-\gamma_\Delta^{|E|-1}) e_{max} < \Delta \quad (31)$$

e.g.,

$$e_{max}|E|(1-\gamma_\Delta^{|E|-1}) < \Delta \quad (32)$$

As is shown above, InEq.32 is stronger than the original InEq.25. Hence we can solve InEq.32 and obtain a smaller appropriate interval of γ_Δ,

$$1 > \gamma_\Delta > (1 - \frac{\Delta}{e_{max}|E|})^{\frac{1}{|E|-1}} \quad (33)$$

∎

Theorem 4, cooperated with theorem2 and theorem3 reveal that we can choose a valid γ in the interval $((1-\frac{\Delta}{e_{max}|E|})^{\frac{1}{|E|-1}}, 1)$, such that the Q$_{SSP}$ and SARSA$_{SSP}$ algorithm can converge as well as find the correct path, without violating the constraint of $\gamma < 1$, which is necessary for the convergence proofs.

IV. EXPERIMENTAL RESULTS AND ANALYSIS

In this section, we compare the performance of proposed algorithms with KL-Hop-by-Hop routing (KL-HHR) algorithm [17], combinatorial upper confidence bound (CUCB) algorithm [33], and Thompson sampling (TS) algorithm [34] applied for this problem. The details of KL-HHR algorithm can be found in [17]. The CUCB algorithm chooses

TABLE 1. Edge length distributions of stochastic graph 1.

Edges	Lengths				Probabilities			
(1, 2)	3	5.3	7.4	9.4	0.2	0.2	0.3	0.3
(1, 3)	3.5	6.2	7.9	8.5	0.3	0.3	0.2	0.2
(1, 4)	4.2	6.1	6.9	8.9	0.2	0.3	0.2	0.3
(2, 5)	2.6	4.1	5.5	9.0	0.2	0.2	0.4	0.2
(2, 6)	5.8	7.0	8.5	9.6	0.3	0.3	0.2	0.2
(3, 2)	1.5	2.3	3.6	4.5	0.2	0.2	0.3	0.3
(3, 7)	6.5	7.2	8.3	9.4	0.5	0.2	0.2	0.1
(3, 8)	5.9	7.8	8.6	9.9	0.4	0.3	0.1	0.2
(4, 3)	2.1	3.2	4.5	6.8	0.2	0.2	0.3	0.3
(4, 9)	1.1	2.2	3.5	4.3	0.2	0.3	0.4	0.1
(5, 7)	3.2	4.8	6.7	8.2	0.2	0.2	0.3	0.3
(5, 10)	6.3	7.8	8.4	9.1	0.2	0.2	0.4	0.2
(6, 3)	6.8	7.7	8.5	9.6	0.4	0.1	0.1	0.4
(6, 5)	0.6	1.5	3.9	5.8	0.2	0.2	0.3	0.3
(6, 7)	2.1	4.8	6.6	7.5	0.2	0.4	0.2	0.2
(7, 6)	4.1	6.3	8.5	9.7	0.2	0.3	0.4	0.1
(7, 8)	1.6	2.8	5.2	6.0	0.2	0.3	0.3	0.2
(7, 9)	3.5	4.0	5.0	7.7	0.1	0.2	0.4	0.3
(7, 10)	1.6	3.4	8.2	9.3	0.2	0.3	0.3	0.2
(8, 4)	7.0	8.0	8.8	9.4	0.2	0.2	0.2	0.4
(8, 7)	2.1	4.6	8.5	9.6	0.4	0.2	0.2	0.2
(8, 9)	1.7	4.9	6.5	7.8	0.2	0.2	0.2	0.4
(9, 10)	4.6	6.4	7.6	8.9	0.4	0.1	0.2	0.3

a whole path for the nth packet according to the following strategy [33]

$$p(n) \in argmin_{p\in\mathcal{P}} \sum_{i\in p} \frac{1}{\hat{e}_i(n) + \sqrt{1.5log(n)/t_i(n)}} \tag{34}$$

The TS algorithm chooses the shortest path according to the sampled length of each edge based on the posterior distribution of edge parameters.

A. EXPERIMENTAL SETUP AND METRIC

We adopt the benchmark stochastic graphs utilized in [29] and [30] to compare the performance of proposed algorithms and competitive algorithms. Fig.2 depicts the network topology of benchmark graphs, on which the source node (resp. destination node) is colored red (resp. blue). The first graph has 10 nodes and 23 edges, and the stochastic shortest path is $\phi^* = (1, 4, 9, 10)$. The second graph is more complicated, with 15 nodes and 42 edges, and the stochastic shortest path is $\phi^* = (1, 2, 5, 15)$. There are 68 possible loop-free paths on graph 1, and 720 possible loop-free paths on graph 2. The edge length distributions of graph 1 and graph 2 are shown in table1 and table2, respectively.

We can evaluate the performance of SSP algorithms by the metric of *regret*, which is defined as the cumulative path length difference between the path chosen by the policy π and that by the optimal policy π^*, which chooses the optimal path every time. More precisely, the regret of policy π up to the N-th episode is defined as the expected difference of rewards for the first N episodes, denoted by $R^\pi(N)$.

$$R^\pi(N) = \mathbb{E}\left[Nr^{\pi^*} - \sum_{i=1}^{N} r^\pi(i)\right] \tag{35}$$

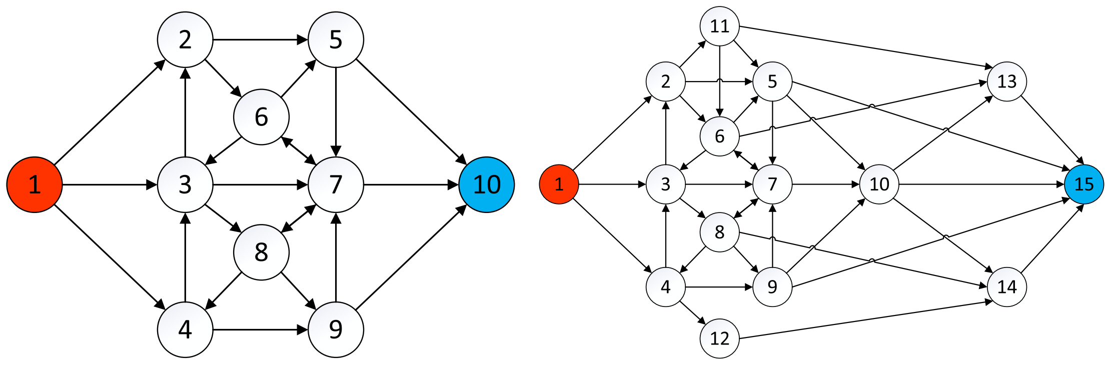
TABLE 2. Edge length distributions of stochastic graph 2.

Edges	Lengths				Probabilities			
(1, 2)	16	25	36		0.6	0.3	0.1	
(1, 3)	21	24	25	39	0.5	0.2	0.2	0.1
(1, 4)	11	13	26		0.4	0.4	0.2	
(2, 11)	24	28	31		0.5	0.3	0.2	
(2, 5)	11	30			0.7	0.3		
(2, 6)	13	37	39		0.6	0.2	0.2	
(3, 2)	11	20	24		0.6	0.3	0.1	
(3, 7)	23	30	34		0.4	0.3	0.3	
(3, 8)	14	23	34		0.5	0.4	0.1	
(4, 3)	22	30			0.7	0.3		
(4, 9)	35	40			0.6	0.4		
(4, 12)	16	25	37		0.5	0.4	0.1	
(5, 13)	28	35	37	40	0.4	0.3	0.2	0.1
(5, 15)	25	32			0.7	0.3		
(5, 10)	27	33	40		0.4	0.3	0.3	
(5, 7)	15	17	19	26	0.3	0.3	0.3	0.1
(6, 5)	18	25	29		0.5	0.3	0.2	
(6, 13)	21	23	36		0.5	0.5		
(6, 7)	11	31	37		0.5	0.4	0.1	
(6, 3)	18	24			0.7	0.3		
(7, 10)	19	23	37		0.6	0.2	0.2	
(7, 8)	12	25	22	24	0.3	0.3	0.2	0.2
(7, 6)	12	23	31		0.5	0.3	0.2	
(8, 7)	14	34	39		0.6	0.2	0.2	
(8, 14)	14	15	27	32	0.3	0.3	0.2	0.2
(8, 9)	13	31	32		0.8	0.1	0.1	
(8, 4)	13	23	34		0.4	0.3	0.3	
(9, 7)	10	17	20		0.6	0.3	0.1	
(9, 10)	16	18	36	39	0.3	0.3	0.2	0.2
(9, 15)	12	13	25	32	0.4	0.3	0.2	0.1
(9, 14)	19	24	29		0.4	0.3	0.3	
(10, 13)	14	20	25	32	0.3	0.3	0.2	0.2
(10, 15)	15	19	25		0.4	0.3	0.3	
(10, 14)	23	34			0.9	0.1		
(11, 13)	13	31	35		0.6	0.3	0.1	
(11, 5)	18	19	20	33	0.63	0.3	0.3	0.1
(11, 6)	10	19	39		0.5	0.4	0.1	
(12, 8)	15	36	39		0.5	0.3	0.2	
(12, 9)	16	22			0.7	0.3		
(12, 14)	10	13	18	34	0.3	0.3	0.3	0.1
(13, 15)	12	31			0.9	0.1		
(14, 15)	14	19	32		0.5	0.3	0.2	

B. RESULTS AND ANALYSIS

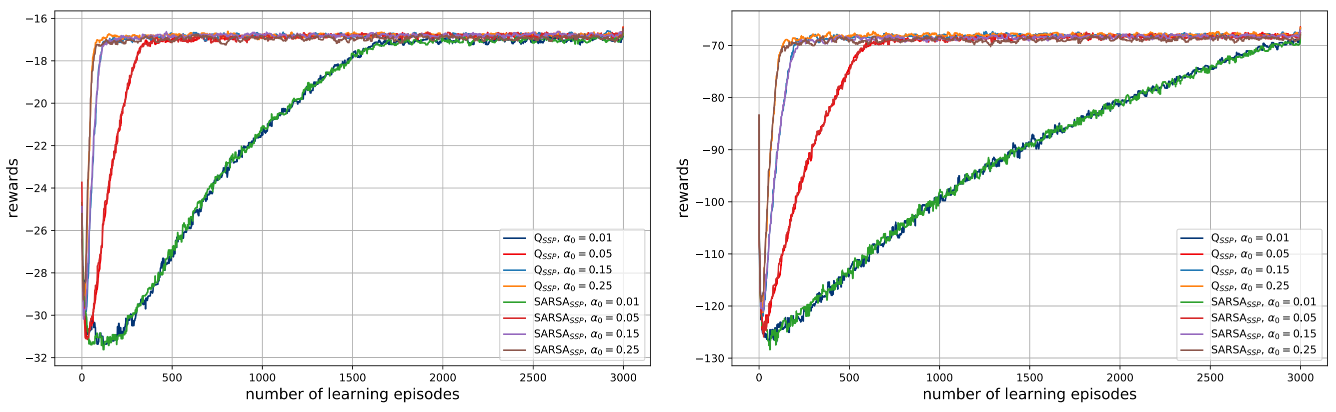
The reward curve w.r.t learning episodes is intuitive to illustrate the performance of an RL algorithm. Fig.3 shows the rewards of Q$_{SSP}$ and SARSA$_{SSP}$ with diverse initial hyper-parameter α_0, to verify the influence of this parameter. The reward curve shows α_0 will affect the convergence speed significantly, especially on more sophisticated graph 2. $\alpha_0 = 0.01$ leads to the slowest convergence speed, while $\alpha_0 = 0.25$ shows the fastest convergence. However, both algorithms with different α_0 converge to approximately same reward level, indicating that α_0 has a limited effect on the converged value of the Q table.

The reward curve also illustrates the fast convergence speed of the proposed algorithms. The convergence speed of the proposed RL-based algorithms can be affected by the learning parameters and the environment/graph. Hence it may be infeasible to derive a closed-form algorithm complexity. To compare the performance and convergence speed of Q$_{SSP}$ and SARSA$_{SSP}$, we illustrate the accuracy of both algorithms in Fig.5, on the more sophisticated graph 2, which has 720 possible paths with only one correct



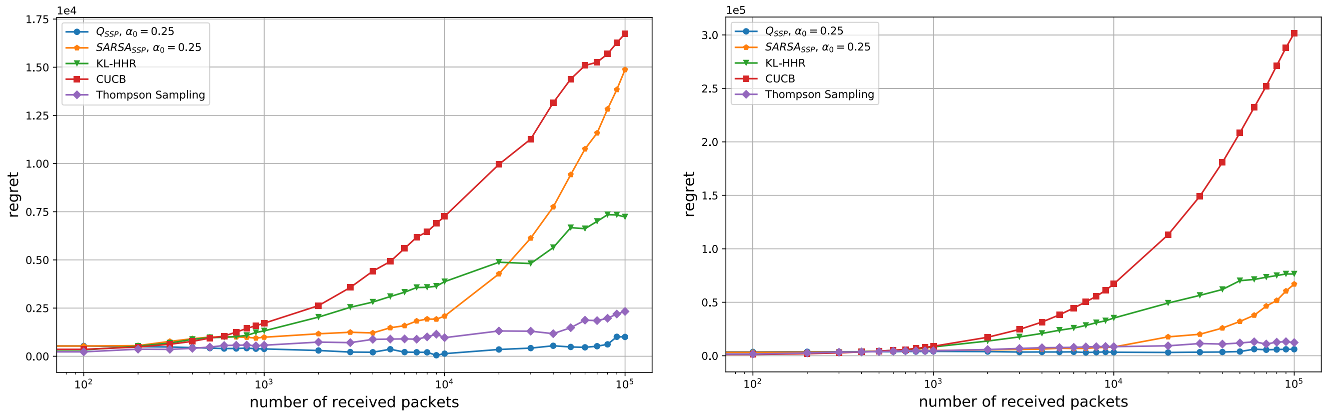
(a) Network topology of stochastic graph 1 (b) Network topology of stochastic graph 2

FIGURE 2. Network topology of benchmark stochastic graphs.



(a) Rewards of Q_{SSP} and $SARSA_{SSP}$ on graph 1. (b) Rewards of Q_{SSP} and $SARSA_{SSP}$ on graph 2.

FIGURE 3. Rewards versus number of learning episodes of Q_{SSP} algorithm and SARSA-$_{SSP}$ algorithm, under different α_0 settings.



(a) Regret of proposed algorithms and comparison algorithms on graph 1, with $\alpha_0 = 0.25$. (b) Regret of proposed algorithms and comparison algorithms on graph 2, with $\alpha_0 = 0.25$.

FIGURE 4. Regret comparison versus number of packets/paths under Q_{SSP}, $SARSA_{SSP}$, KL-HHR, CUCB, and TS algorithms.

stochastic shortest path. The accuracy is defined as the probability that an algorithm outputs the correct stochastic path, estimated in 100 repeated experiments. Fig. 5 demonstrates that Q_{SSP} convergences with almost 1.0 accuracy with less than 500 episodes, while $SARSA_{SSP}$ attains 0.9 accuracy in 2000 learning episodes, with minor fluctuation, which may be caused by the action-selection strategy. Nevertheless, the result reveals that the proposed Q_{SSP} algorithm has a relatively feasible convergence speed and high accuracy compared with $SARSA_{SSP}$.

We also illustrate the regret w.r.t the number of episodes under different algorithms/policies in Fig.4. We can see that

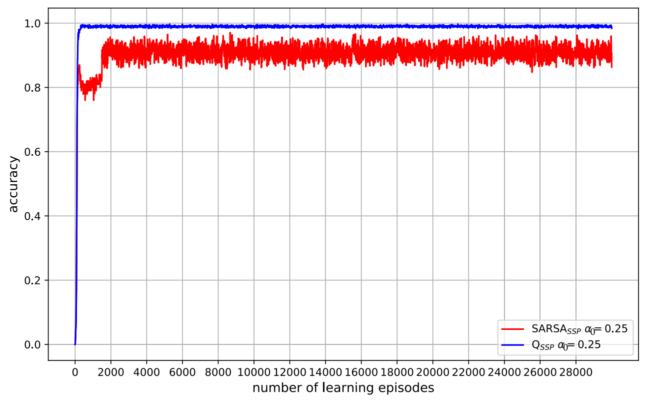


FIGURE 5. **Accuracy comparison between Q_{SSP} and $SARSA_{SSP}$ w.r.t number of learning episodes.**

the proposed Q_{SSP} algorithm and the TS algorithm have better performance than the KL-HHR, CUCB, and $SARSA_{SSP}$ algorithm on both graphs. The Q_{SSP} performs the best than other competitors. The $SARSA_{SSP}$ algorithm is inferior to the Q_{SSP} algorithm, which may be caused by the action selection strategy. When choosing a path, the off-policy Q_{SSP} algorithm adopts the greedy policy according to the current Q table, while using a ϵ-greedy policy for learning. In contrast, the $SARSA_{SSP}$ algorithm uses the ϵ-greedy policy for both learning and path planning. We can induce that even the $SARSA_{SSP}$ algorithm learns the optimal Q table, the probability of finding the correct path is approximate $(1 - \epsilon)^n$, where n is the number of edges on the optimal path. In short, the proposed algorithms, especially Q_{SSP}, converge faster and are non-sensitive to the initial parameter setting. The Q_{SSP} algorithm outperforms other competitors significantly, with fast learning procedure and lower long-term regret.

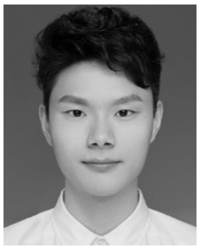
V. CONCLUSION

In this paper, we tackle the stochastic shortest path problem using reinforcement learning schemes by modeling the path searching procedure as an appropriate discounted Markov decision process. Specifically, we devise the off-policy Q_{SSP} algorithm and the on-policy $SARSA_{SSP}$ algorithm. The proposed algorithms learn and find the stochastic shortest path in an online manner, utilizing every timestep's feedback to adjust state-action value functions, instead of the whole path's feedback. The specially devised negative reward-averaging technique stabilizes and facilitates the learning process, leading to fast and accurate convergence. We theoretically prove the existence of an appropriate discounted factor $\gamma < 1$, which ensures the convergence as well as the correctness. We compare the regret of proposed algorithms and other algorithms on two benchmark graphs. The experimental results have shown that Q_{SSP} performs better than competitive algorithms and the $SARSA_{SSP}$ algorithm. The reward curves and accuracy curve indicate that Q_{SSP} and $SARSA_{SSP}$ algorithm have fast convergence speed. Furthermore, the proposed solutions can be easily scaled to more massive stochastic graphs, with just minor modifications to the Q table, making them scalable to more complicated graphs.

REFERENCES

[1] P. Suriyachai, U. Roedig, and A. Scott, "A survey of MAC protocols for mission-critical applications in wireless sensor networks," *IEEE Commun. Surveys Tuts.*, vol. 14, no. 2, pp. 240–264, 2nd Quart., 2012.
[2] A. Mainwaring, D. Culler, J. Polastre, R. Szewczyk, and J. Anderson, "Wireless sensor networks for habitat monitoring," in *Proc. 1st ACM Int. Workshop Wireless sensor Netw. Appl.*, Sep. 2002, pp. 88–97.
[3] H. Alemdar and C. Ersoy, "Wireless sensor networks for healthcare: A survey," *Comput. Netw.*, vol. 54, no. 15, pp. 2688–2710, Oct. 2010.
[4] M. Othman and K. Shazali, "Wireless sensor network applications: A study in environment monitoring system," *Procedia Eng.*, vol. 41, pp. 1204–1210, Jul. 2012.
[5] K. Martinez, R. Ong, and J. Hart, "Glacsweb: A sensor network for hostile environments," in *Proc. 1st Annu. IEEE Commun. Soc. Conf. Sensor Ad Hoc Commun. Netw.*, Oct. 2004, pp. 81–87.
[6] T. Luo, H.-P. Tan, and T. Q. S. Quek, "Sensor OpenFlow: Enabling software-defined wireless sensor networks," *Commun. Lett.*, vol. 16, no. 11, pp. 1896–1899, Nov. 2012.
[7] J. Zhou, H. Jiang, J. Wu, L. Wu, C. Zhu, and W. Li, "SDN-based application framework for wireless sensor and actor networks," *IEEE Access*, vol. 4, pp. 1583–1594, 2016.
[8] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "Openflow: Enabling innovation in campus networks," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 38, no. 2, pp. 69–74, Apr. 2008.
[9] B. A. A. Nunes, M. Mendonca, X.-N. Nguyen, K. Obraczka, and T. Turletti, "A survey of software-defined networking: Past, present, and future of programmable networks," *IEEE Commun. Surveys Tuts.*, vol. 16, no. 3, pp. 1617–1634, 3rd Quart., 2014.
[10] R. Sun, S. Tatsumi, and G. Zhao, "Q-MAP: A novel multicast routing method in wireless ad hoc networks with multiagent reinforcement learning," in *Proc. IEEE Region 10 Conf. Comput., Commun., Control Power Eng.*, vol. 1, Oct. 2002, pp. 667–670.
[11] A. Forster and A. L. Murphy, "FROMS: Feedback routing for optimizing multiple sinks in WSN with reinforcement learning," in *Proc. 3rd Int. Conf. Intell. Sensors, Sensor Netw. Inf.*, Dec. 2007, pp. 371–376.
[12] R. Arroyo-Valles, R. Alaiz-Rodriguez, A. Guerrero-Curieses, and J. Cid-Sueiro, "Q-probabilistic routing in wireless sensor networks," in *Proc. 3rd Int. Conf. Intell. Sens. Sensor Netw. Inf.*, Dec. 2007, pp. 1–6.
[13] C. Guestrin, P. Bodik, R. Thibaux, M. Paskin, and S. Madden, "Distributed regression: An efficient framework for modeling sensor network data," in *Proc. 3rd Int. Symp. Inf. Process. Sensor Netw.*, Apr. 2004, pp. 1–10.
[14] T. Hu and Y. Fei, "QELAR: A machine-learning-based adaptive routing protocol for energy-efficient and lifetime-extended underwater sensor networks," *IEEE Trans. Mobile Comput.*, vol. 9, no. 6, pp. 796–809, Jun. 2010.
[15] N. Javaid, O. A. Karim, A. Sher, M. Imran, A. U. H. Yasar, and M. Guizani, "Q-learning for energy balancing and avoiding the void hole routing protocol in underwater sensor networks," in *Proc. 14th Int. Wireless Commun. Mobile Comput. Conf. (IWCMC)*, Jun. 2018, pp. 702–706.
[16] C. J. C. H. Watkins, "Learning from delayed rewards," Ph.D. dissertation, Dept. King's College Univ. Cambridge, Univ. Cambridge, Cambridge, U.K., 1989.
[17] M. S. Talebi, Z. Zou, R. Combes, A. Proutiere, and M. Johansson, "Stochastic online shortest path routing: The value of feedback," *IEEE Trans. Autom. Control*, vol. 63, no. 4, pp. 915–930, Apr. 2018.
[18] J. Moy. (Apr. 1998). *OSPF Version 2*. [Online]. Available: https://www.ietf.org/rfc/rfc2328.txt
[19] Y. Rekhter and T. Li. (Jul. 1995). *A Border Gateway Protocol 4 (BGP-4)*. [Online]. Available: https://www.rfc-editor.org/rfc/rfc1654.txt
[20] M. Hajian, E. Melachrinoudis, and P. Kubat, "Modeling wildfire propagation with the stochastic shortest path: A fast simulation approach," *Environ. Model. Softw.*, vol. 82, pp. 73–88, Aug. 2016.
[21] B. Bonet and H. Geffner, "Planning with incomplete information as heuristic search in belief space," in *Proc. 5th Int. Conf. Artif. Intell. Planning Syst.*, Apr. 2000, pp. 52–61.
[22] G. Andonov and B. Yang, "Stochastic shortest path finding in path-centric uncertain road networks," in *Proc. 19th IEEE Int. Conf. Mobile Data Manage. (MDM)*, Jun. 2018, pp. 40–45.

[23] H. Frank, "Shortest paths in probabilistic graphs," *Oper. Res.*, vol. 17, no. 4, pp. 583–599, Aug. 1969.
[24] A. A. B. Pritsker, "Application of multichannel queueing results to the analysis of conveyor systems," *J. Ind. Eng.*, vol. 17, no. 1, pp. 14–21, Jan. 1966.
[25] P. B. Mirchandani, "Shortest distance and reliability of probabilistic networks," *Comput. Oper. Res.*, vol. 3, no. 4, pp. 347–355, Dec. 1976.
[26] G. S. Fishman, "Estimating critical path and arc probabilities in stochastic activity networks," *Nav. Res. Logistics Quart.*, vol. 32, no. 2, pp. 249–261, May 1985.
[27] V. G. Adlakha, "Note—An improved conditional Monte Carlo technique for the stochastic shortest path problem," *J. Manage. Sci.*, vol. 32, no. 10, pp. 1360–1367, Oct. 1986.
[28] C. E. Sigal, A. A. B. Pritsker, and J. J. Solberg, "The stochastic shortest route problem," *Oper. Res.*, vol. 28, no. 5, pp. 1122–1129, 1980.
[29] H. Beigy and M. R. Meybodi, "Utilizing distributed learning automata to solve stochastic shortest path problems," *Int. J. Uncertainty, Fuzziness Knowl.-Based Syst.*, vol. 14, no. 5, pp. 591–615, Oct. 2006.
[30] Y. Guo, S. Li, W. Jiang, B. Zhang, and Y. Ma, "Learning automata-based algorithms for solving the stochastic shortest path routing problems in 5G wireless communication," *Phys. Commun.*, vol. 25, pp. 376–385, Dec. 2017.
[31] K. Liu and Q. Zhao, "Adaptive shortest-path routing under unknown and stochastically varying link states," in *Proc. 10th Int. Symp. Model. Optim. Mobile, Ad Hoc Wireless Netw. (WiOpt)*, May 2012, pp. 232–237.
[32] T. He, D. Goeckel, R. Raghavendra, and D. Towsley, "Endhost-based shortest path routing in dynamic networks: An online learning approach," in *Proc. IEEE INFOCOM*, Apr. 2013, pp. 2202–2210.
[33] W. Chen, Y. Wang, and Y. Yuan, "Combinatorial multi-armed bandit: General framework and applications," in *Proc. Int. Conf. Mach. Learn.*, Feb. 2013, pp. 151–159.
[34] D. J. Russo, B. Van Roy, A. Kazerouni, I. Osband, and Z. Wen, "A tutorial on thompson sampling," *Found. Trends Mach. Learn.*, vol. 11, no. 1, pp. 1–96, 2018.
[35] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton, Y. Chen, T. Lillicrap, F. Hui, L. Sifre, G. van den Driessche, T. Graepel, and D. Hassabis, "Mastering the game of go without human knowledge," *Nature*, vol. 550, no. 7676, p. 354, Oct. 2017.
[36] D. Silver, T. Hubert, J. Schrittwieser, I. Antonoglou, M. Lai, A. Guez, M. Lanctot, L. Sifre, D. Kumaran, and T. Graepel, "Mastering chess and shogi by self-play with a general reinforcement learning algorithm," 2017, *arXiv:1712.01815*. [Online]. Available: https://arxiv.org/abs/1712.01815
[37] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, p. 529, 2015.
[38] H. Cai, K. Ren, W. Zhang, K. Malialis, J. Wang, Y. Yu, and D. Guo, "Real-time bidding by reinforcement learning in display advertising," in *Proc. 10th ACM Int. Conf. Web Search Data Mining*, Feb. 2017, pp. 661–670.
[39] H. Cai, J. Yang, W. Zhang, S. Han, and Y. Yu, "Path-level network transformation for efficient architecture search," 2018, *arXiv:1806.02639*. [Online]. Available: https://arxiv.org/abs/1806.02639
[40] H. Cai, T. Chen, W. Zhang, Y. Yu, and J. Wang, "Efficient architecture search by network transformation," in *Proc. 32nd AAAI Conf. Artif. Intell.*, Apr. 2018, pp. 1–8.
[41] F. Liu, R. Tang, X. Li, W. Zhang, Y. Ye, H. Chen, H. Guo, and Y. Zhang, "Deep reinforcement learning based recommendation with explicit user-item interactions modeling," 2018, *arXiv:1810.12027*. [Online]. Available: https://arxiv.org/abs/1810.12027
[42] H. Chen, X. Dai, H. Cai, W. Zhang, X. Wang, R. Tang, Y. Zhang, and Y. Yu, "Large-scale interactive recommendation with tree-structured policy gradient," 2018, *arXiv:1811.05869*. [Online]. Available: https://arxiv.org/abs/1811.05869
[43] R. S. Sutton and A. G. Barto, *Reinforcement Learning—An Introduction*. Cambridge, MA, USA: MIT Press, 2018.
[44] T. Jaakkola, M. I. Jordan, and S. P. Singh, "Convergence of stochastic iterative dynamic programming algorithms," in *Proc. Adv. Neural Inf. Process. Syst.*, 1994, pp. 703–710.
[45] M. S. Francisco, "Convergence of Q-learning: A simple proof," Inst. Syst. Robot., Zurich, Switzerland, Tech. Rep, 2001, pp. 1–4.



WENWEN XIA received the B.S. degree from Wuhan University, Wuhan, China, in 2018. He is currently pursuing the Ph.D. degree with the School of Electronic Information and Electrical Engineering, Shanghai Jiao Tong University, Shanghai, China. His research interests include reinforcement learning, multiagent reinforcement learning and their applications, and wireless sensor networks.

CHONG DI received the B.S. degree from Xidain University, in 2016. He is currently pursuing the Ph.D. degree with the School of Electronic Information and Electrical Engineering, Shanghai Jiao Tong University. His research interests include machine learning, reinforcement learning, and learning automata and their applications.

HAONAN GUO received the B.Sc. degree in electronic information science and technology from Xidian University, Xi'an, China, in 2009. He is currently pursuing the Ph.D. degree with the Department of Electrical Engineering, Shanghai Jiao Tong University. His research interests include machine learning, deep learning, and data mining.

SHENGHONG LI received the B.S. and M.S. degrees in electrical engineering from the Jilin University of Technology, China, in 1993 and 1996, respectively, and the Ph.D. degree in radio engineering from the Beijing University of Posts and Telecommunications, China, in 1999.

Since September 1999, he has been with Shanghai Jiao Tong University, China, as a Research Fellow, an Associate Professor, and a Professor, successively. In 2010, he was a Visiting Scholar with Nanyang Technological University, Singapore. He is currently a Professor with the School of Cyber Space Security, Shanghai Jiao Tong University. He published more than 80 articles, coauthored four books, and holds ten granted patents. His research interests include information security, signal and information processing, and artificial intelligence. In 2003, he received the 1st Prize of Shanghai Science and Technology Progress in China. In 2006 and 2007, he was elected for New century talent of Chinese Education Ministry and Shanghai Dawn Scholar.