S-Nav: Safety-Aware IoT Navigation Tool for Avoiding COVID-19 Hotspots

Sudip Misra, *Senior Member, IEEE*, Pallav Kumar Deb, *Student Member, IEEE*, Naimisha Koppala, Anandarup Mukherjee, *Student Member, IEEE*, and Shiwen Mao, *Fellow, IEEE*

Abstract—In this paper, we present a Q-learning-enabled safe navigation system – S-Nav – that recommends routes in a road network by minimizing traveling through categorically demarcated COVID-19 hotspots. S-Nav takes the source and destination as inputs from the commuters and recommends a safe path for traveling. The S-Nav system dodges hotspots and ensures minimal passage through them in unavoidable situations. This feature of S-Nav reduces the commuter's risk of getting exposed to these contaminated zones and contracting the virus. To achieve this, we formulate the reward function for the reinforcement learning model by imposing zone-based penalties and demonstrate that S-Nav achieves convergence under all conditions. To ensure real-time results, we propose an Internet of Things (IoT)-based architecture by incorporating the cloud and fog computing paradigms. While the cloud is responsible for training on large road networks, the geographically-aware fog nodes take the results from the cloud and retrain them based on smaller road networks. Through extensive implementation and experiments, we observe that S-Nav recommends reliable paths in near real-time. In contrast to state-of-the-art techniques, S-Nav limits passage through red/orange zones to almost 2% and close to 100% through green zones. However, we observe 18% additional travel distances compared to precarious shortest paths.

Index Terms—Path planning, Reinforcement Learning, Hotspots, Q-Learning Model, Fog Computing, Shortest Path, Internet of Things

1 Introduction

The COVID-19 virus has spread throughout all major countries with an explosion in the number of infected individuals. The *small size* and *intangible* nature of the virus has led to an *uncontrollable* spread across the world, resulting in creation of COVID-19 *hotspots*. Since contact tracing in these areas is challenging [1], the concerned authorities identify these hotspots and categorize them as *red*, *orange*, or *green* zones. These zones are classified based on the severity of the spread, and the restrictions in each zone vary accordingly. Although the conditions are adverse, people need to *commute* from one place to the other regularly for work and basic amenities. *Safe Transportation* is a concern

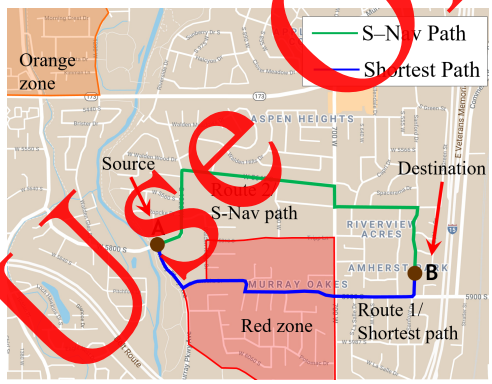


Figure 1: Road route planning in presence of hotspots marked as red, orange, and green zones.

at this moment, and *avoidance* of hotspots is of paramount importance. Conventional route-finding methods such as shortest path and optimized path algorithms [2] do not suffice in finding safe routes in COVID-19 environments. However, a road map consists of a complex network of multiple routes from a particular source (S) to a destination (D). In such situations, intelligent routines that recommend routes that bypass hotspots or minimize passage through them is necessary for ensuring the safety of the people.

In this work, we propose and develop a Q-learning-based smart navigation system – **S-Nav** – that avoids COVID-19 hotspots according to the category of the zones for ensuring the safety of the commuters. S-Nav takes S and D as inputs from the users/commuters and recommends a safe path (S-Nav path) that – 1) is optimal and 2) minimizes traveling through the hotspots. To achieve this, we design the road route recommender system by formulating rewards based on the *categorical* hotspot zones in a road map like the one in Fig. 1. We also ensure minimal exposure by minimizing the travel distance through the zones. In summary, we account for the *category* of the zones and the travel *length* in each zone. In addition to the need for safe routes in road networks, real-time results are also essential in mobile environments. However, Q-learning methods depend on matrix-based operations, which are relatively time-consuming, specifically for large road networks. Since the Internet of Things (IoT)-based solutions have the potential

S. Misra, P. K. Deb, and A. Mukherjee are with the Department of Computer Science and Engineering, Indian Institute of Technology Kharagpur, India. e-mail: (sudipm@iitkgp.ac.in, pallv.deb@iitkgp.ac.in, and anandarupmukherjee@ieee.org)
N. Koppala is with the Department of Mathematics and Computing, Indian Institute of Technology Delhi, India. e-mail: (naimisha.dps@gmail.com)
S. Mao is with the Department of Electrical and Computer Engineering, Auburn University, United States. e-mail: (smao@ieee.org)

to overcome such issues [3], we propose a distributed architecture for our system by adopting *cloud* and *fog* computing paradigms to reduce the computation time in S-Nav. We execute a preliminary road learning routine on the cloud servers for large road networks (cities/states) and forward the trained model to the fog nodes (FNs). The FNs then sight attention to *granular* details and retrain them based on designated *geographical regions*. Since FNs are usually resource-constrained, it is suitable to execute S-Nav on small-scale road networks. Additionally, the FNs are closer to the commuters, which helps in reducing latency. The FNs may also keep track of the changing categories of zones and update its database accordingly. In case an area has no fog nodes, the commuters may receive the recommended routes directly from the cloud with some additional delays. It may be noted that the proposed S-Nav system may extend to any application that involves the need to avoid traveling through certain zones/regions. These may include hazards such as, but not limited to, radiations, poisonous gas leaks, oil tank blasts, pandemics, riots, blockages, and others. In this work, we choose the COVID-19 scenario because of: 1) the need of the hour and the urgency to restrict the spread of the COVID-19 virus, and 2) a proof of concept to show the feasibility of S-Nav.

Example Scenario: We illustrate the working of the proposed S-Nav system by considering a commuter who needs to travel from location A to B in the map in Fig. 1. Conventional road route planning techniques may recommend an optimal path (Route 1 in Fig. 1) based on the distance and traffic conditions. However, this path may pass through one of the categorical hotspots, which exposes the commuter to the threat of contracting the COVID-19 virus. In such scenarios, the path recommended by S-Nav (Route 2 in Fig. 1) helps in avoiding the risky zones and ensures safe travel, reducing the risk of exposure. Additionally, as we design the execution and response to navigation requests from the FNs close to the commuters, the S-Nav system recommends routes in near real-time. Additionally, pre-trained models from the cloud also help ensure low retraining time at the FNs and fast local convergence (geographical region).

1.1 Difference of S-Nav from commercial navigation tools

Some of the most commonly used applications are *Google Maps*, *Apple Maps*, *BackCountry Navigator*, *HERE WeGo Maps*, and others. Apart from their uniqueness in features such as online/offline, personalization, and visualization, these applications typically focus on road traffic, blockage, speed, and distance for finding the shortest/optimized route from source to destination. Such methods are not suitable for use in the current scenario of COVID-19. The commuters need to avoid traveling through hotspots to reduce exposure. S-Nav recommends paths that bypass traveling through hotspots. In unavoidable situations, S-Nav ensures minimal travel through them.

1.2 Contribution

In this work, we propose and develop an IoT-based system – S-Nav – for recommending safe road routes in a road network by avoiding COVID-19 hotspots and ensuring the safety of the commuters. Towards this, our specific set of contributions are:

- ***S-Nav System:*** The proposed S-Nav is a Q-based reinforcement learning road route recommender system for ensuring commuters' safety from the COVID-19 virus while travelling.
- ***Rewards:*** To ensure complete/partial avoidance of the hotspot zones, we impose penalties based on the category of the zones. We also ensure minimal travel through the zones in case of unavoidable situations.
- ***IoT-based Architecture:*** To minimize the training time and the time for delivering results, we adopt a fog-cloud architecture for the S-Nav system.
- ***Evaluation:*** To demonstrate the performance of the proposed S-Nav system, we perform experiments exhaustively and present results.

It may be noted that although we performed our experiments on datasets based on real road maps, we annotated the hotspots and their categories randomly before training and implementation.

We organize the rest of the paper as follows. We present some of the existing literature in Section 2, followed by the proposed method in Section 3. We then present our implementation setup in Section 4 and the observed results in Section 5, and finally conclude in Section 6.

2 Related Work

Road route/path planning has been an area of great interest among researchers. Apart from parameters such as distance, road route planning techniques also consider traffic, data derived from GPS systems, and others to determine efficient paths. In this section, we categorize and briefly describe some of the existing route planning methods in literature.

2.1 Road route/path planning techniques

Silva *et al.* [4] proposed a method for localizing the robots/devices and support them for navigation using Kinect sensor and Convolution Neural Networks (CNN). Similarly, the authors in [5] proposed a localization and navigation tool using Radio Frequency Identification (RFID) and petri net technologies. The authors in [6] designed a routing model by considering intersection signals and real-time velocity of the vehicle. The authors ensured reduced travel time and energy consumption. The authors in [7] developed a model based on Dempster-Shafer theory to calculate the uncertainty (road conditions) in cost function while using Dijkstra's algorithm. In [8], the authors built a two-step model (using K-path and shuffled frog leaping algorithm) for dynamic path planning by considering real-time traffic data and travel speed as parameters. Customer-centric routes have a unique impact than conventional ones. It is essential to model the routes according to the requests. The authors in [9] proposed a log C-means clustering algorithm (LFCM) to form clusters based on driving style. They then used the Ant Colony algorithm to calculate the shortest

path. Sun *et al.* [10] proposed a graph-based method to minimize travel time by either recommending the shortest path or the best starting time using conventional and extended Bellman-Ford algorithms. The authors considered the traffic pattern and its history to recommend the starting time and reduce the travelling time. Horvath *et al.* [11] used a traditional four-step model for dynamically assigning parameters (traffic and user specifications) in two matrices to estimate an efficient route. The authors in [12] designed an Optimized Path Algorithm Based on Reinforcement Learning (OPABRL) using prior reinforcement learning (RL) and improvised searched A* algorithm.

2.2 Fog computing and IoT solutions

The authors in [13] presented an IoT-based method for taking decisions based on the past and present scenario in a smart city as a connected community. The fog-cloud architecture helps in load sharing for performing machine learning operations on resource-constrained FNs [14]. Ahmad *et al.* [15] proposed a route recommender system for solid garbage collection by waste carrier vehicles. They built the recommender system by profiling the areas based on waste production rates. The authors in [16] demonstrated how IoT systems have the potential to monitor these routes and vehicles by using off-the-shelf sensors for communicating with street lights and other stationary units installed on the street. Xu *et al.* [17] proposed a crowd evacuation recommender system in case of disasters. IoT solutions also help guide and plan optimized routes for data traffic in applications such as smart homes [18]. The authors in [19] exploited the features of Q-learning methods to schedule transmissions to ensure reliable exchange of data. Such methods play an important role in sending decisions in real-time IoT environments.

2.3 Synthesis

Route planning based on different parameters and under various conditions is a well-explored field in the research community. As discussed in the previous sections, the existing literature offers efficient route planning solutions. However, there is a lacuna in the study. The existing solution techniques typically focus on finding the shortest/optimal paths between the source and destinations. Such methods are not suitable for use in the current scenario due to the threat of the COVID-19 virus. The commuters need to maintain social distancing and avoid traveling through zones that may be part of the optimized routes to reduce exposure. Further, training machine learning models takes significant time, which opens the scope for distributed learning in fog/edge computing platforms.

3 System Model

In this section, we first present our network architecture. We then briefly explain the need for RL in lieu of other solution techniques for determining the road routes in COVID-19 situations. We then present our formulations and algorithms towards training and deployment of S-Nav.

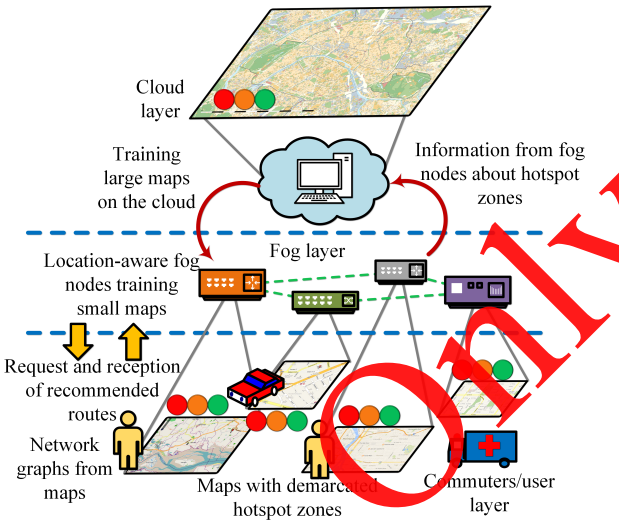


Figure 2: Network architecture for the S-Nav system.

3.1 Network architecture

We consider an IoT-based network architecture, as shown in Fig. 2 for realising S-Nav. In this work, we consider a scenario with a remote cloud server $\mathcal{C}$ and a set of FNs $\mathcal{F} = \{f_1, f_2, ..., f_n\}$. The cloud is responsible for computing the paths for a wide area (cities/states) and for areas that are devoid of FNs. On the other hand, we propose the use of location-aware FNs for S-Nav. In other words, for a set of geographical regions $\mathcal{G} = \{g_1, g_2, ..., g_m\}$ each of the FNs is responsible for their own geographical area. This strategy reduces the size of the map on the FNs, which is suitable for the resource-constrained nature of the FNs. It may be noted that we consider networking devices such as *switches*, *routers*, and others for assuming the role of FNs. Although the communication among the FNs for information sharing is beyond the scope of this work, we envision that each geographical region is associated with only one fog node, i.e., for f_{pq} to be the p^{th} FN associated with q^{th} geographical region, $\sum_q f_{pq} \leq 1$. As $\mathcal{C}$ tries to train for a large region, and the FNs feed granular details with respect to the hotspots, the separation of the tasks among the fog and cloud helps distribute the load along with easy data management. Further, once the model is ready, the FNs respond to requests from the set of users/commuters $\mathcal{U} = \{u_1, u_2, ...u_k\}$. Since the FNs are closer to the users, the delay in obtaining the recommended routes is minuscule [20]. It may be noted that the red zones are converted to orange when there are no COVID-19 positive cases for 14 continuous days, and its conversion to the green zone needs at least 28 continuous days with no positive reports. In this work, we depend on some user intervention in this regard and allow only concerned authorities to update the databases. In the extension of this work, we plan to incorporate web crawlers to update the databases from reliable sites periodically.

3.2 The Motivation of Using Reinforcement Learning

On the one hand, as the COVID-19 virus is spreading rapidly, concerned authorities are successful in limiting the spread on the other. Due to this hand-off, the status of the hotspots keeps changing at specific intervals. The criteria for road routes need to change per the changing states of the hotspots. In such scenarios, RL-based solutions give us the scope for imposing dynamic penalties for the hotspots. The model then trains based on the updated penalties before recommending the safe paths with the motive to maximize the rewards. RL also allows the model to learn from the actions based on the outcome of past decisions. The feedback loop plays a significant role in avoiding past mistakes and making smart decisions. Further, RL allows straightforward updates in the system as the state of the zones change. Due to these salient features, we use RL, particularly Q-learning, for realizing S-Nav. Q-learning is an off-policy value-based RL technique that does not depend on the conventional greedy methods and estimates its reward for the future. In summary, because of the *feedback loop*, *Q-learning*, *changing status of the hotspots*, and *non-greedy* selection technique, we use RL instead of other possible solution techniques. Another attractive feature of using reinforcement learning is that it is an online learning approach, which allows training the model across devices in parts.

3.3 Road network and graph generation

In this section, we explain the graph generation and then navigation with respect to the network architecture explained in Section 3.1. For a geographic region (small/large), we extract the road information and form a road network graph. We represent it as $\mathcal{G} =< V, E >$, such that V is the set of vertices representing the intersection points, and E is the set of edges representing the roads. It may be noted that the creation of the network map from the maps is beyond the scope of this work, and we rely on publicly available datasets for the same. For the set of vertices $V = \{v_1, v_2, ..., v_a\}$ and corresponding edges $E = \{e_1, e_2, ..., e_b\}$, the final vertex-edge pair representing the road route is $< V^*, E^* > \subseteq < V, E >$. The $< V^*, E^* >$ pair is user-centric as it depends on the commuter's source and destination. As mentioned in Section 3.1, $\mathcal{C}$ is responsible for $\mathcal{G}$ that spans over a large area, while the FNs focus on only on its region of interest (ROI)/sector. In case the source and the destination belong to different ROIs, we execute the route-finding routine on $\mathcal{C}$. In the future, we plan to extend this work by enabling the FNs to communicate with one another and perform virtual map *stitching*. We believe processing in the FNs will incur less delay as compared to $\mathcal{C}$.

3.4 Hotspot category-aware rewards

As explained in Section 3.3, we obtain $\mathcal{G}$ from the maps to generate a reward-matrix and process it on $\mathcal{C}$ or $\mathcal{F}$ to produce the safety-aware path (P_{S-Nav}). For $\mathcal{G} =< V, E >$, we represent the length of the i^{th} edge as $e_i^{dist} = l_i$ and the set of all distances as $\mathcal{L} = \{l_1, l_2, ..., l_c\}$. We represent the maximum, minimum, and average of the distances in $\mathcal{L}$ as l_{max}, l_{min}, and l_{avg}, respectively. Since it is challenging to create zones in $< V, E >$, we introduce a *containment factor* α to represent the category of the containment zone that a road passes through or belongs to. Thus, for a road/edge e_i, its containment factor is α_i and the set of all containment factors is $\alpha = \{\alpha_1, \alpha_2, \alpha_3, ..., \alpha_b\}$, such that $|E| = |\alpha|$. For instance, the government has divided the regions into zones according to three categories: 1) Red zone (α^r): regions that have a large number of COVID-19 positive cases, 2) Orange zone (α^o): regions with a relatively fewer number of COVID-19 cases compared to the red zones, and 3) Green zone (α^g): regions with no reported COVID-19 cases. In this work, we assign low α values to edges belonging to zones with lower severity, implying that $\alpha^g < \alpha^o < \alpha^r$. In context of the mentioned parameters, we formulate the reward of a path based on two major components: 1) path length or distance and 2) intensity of containment. For r_{e_i} as the reward associated with e_i, we calculate it as:

$$r_{e_i} = \frac{l_{max}}{1 + e^{\frac{l_{max} - l_i}{l_{avg}}}} \times \alpha_{e_i} \tag{1}$$

$$\text{where } r_{e_1} > r_{e_2} \text{ when } l_1 < l_2 \text{ and } \alpha_{e_1} = \alpha_{e_2} \tag{2}$$

$$\text{and } r_{e_1} > r_{e_2} \text{ when } l_1 = l_2 \text{ and } \alpha_{e_1} < \alpha_{e_2} \tag{3}$$

There may be situations where traversal through a hotspot is unavoidable. We ensure minimum passage through them by accounting the $l_x \in \mathcal{L}$ values in addition to α to reduce the risk of contracting the virus. In equation 2, in case $\alpha_{e_1} = \alpha_{e_2}$, we assign higher rewards for edges with smaller distances. For instance, r_{e_i} is higher for $min(l_1, l_2)$. In case $l_1 = l_2$, r_{e_i} is higher for those with lower containment factor ($min(\alpha_1, \alpha_2)$ in equation 3). We create the reward-matrix $\mathcal{R}$ of size $a \times a$, such that the indices of the columns and rows represent the set $\mathcal{V}$. Mathematically, the entries of $\mathcal{R}$ corresponding to the e_{ij} connecting v_i and v_j is:

$$\mathcal{R}[v_i, v_j] = \begin{cases} r_{e_i}, & \text{if } \exists e_{ij} \neq 0 \\ \text{-1}, & \text{otherwise.} \end{cases} \tag{4}$$

The first condition in equation 4 assigns positive values to the matrix elements in case there exists an edge connecting the vertices v_i and v_j. In case there is no edge connecting them, we assign a negative value, specifically -1 (second condition).

3.5 S-Nav: Safety-aware smart navigation

S-Nav operates in two steps. First, we train the devices involved in finding P_{S-Nav} with respect to the corresponding $\mathcal{G}$ and then calculate P_{S-Nav}. For the training phase, we create a Q-matrix (Q) on obtaining $\mathcal{R}$, which acts as the memory for the model. We initialize the entries by setting them all to be 0. We use row_i to denote the set of entries in i^{th} row of Q and max_{row_i} to be the maximum of row_i. Also, we use c_s to denote the current state, and n_s for the next state from c_s. $\mathcal{A}_{(c_s, n_s)}$ is the action of moving from c_s to n_s. We update the reward entries for $\mathcal{A}_{(c_s, n_s)}$ in the Q as:

$$Q[c_s, n_s] = \mathcal{R}[c_s, n_s] + \gamma . max_{row_i} \tag{5}$$

where, γ is the learning parameter. For gamma values close to 0, the model considers immediate reward for its actions. When gamma is close to 1, the model considers cumulative reward for its actions and chooses to delay it if necessary. The reward entries in the Q matrix may be represented as $r_{\mathcal{A}_{(c_s,n_s)}} = Q[c_s, n_s]$. Finally, the S-Nav system calculates P_{S-Nav} by maximizing the sum of the Q-matrix rewards. Mathematically, our objective function is:

$$max \sum_{i=S,j=max_i}^{D,D} r_{\mathcal{A}_{(i,j)}} \tag{6}$$

where, i represents the current state and j is the best next state from i. The state j is found by traversing through all the entries at the i^{th} row of the Q-matrix. Mathematically, $Q[i, max_i] = max(Q[i, v_k])$ where $v_k \epsilon \mathcal{V}$.

In summary, S-Nav operates on the roads from $\mathcal{G}$ and the corresponding $\mathcal{R}$ matrix. The rewards for each edge in equation 1 takes care of assigning higher rewards to low containment factors and distances. The low containment factors ensure taking routes through low-risk regions. The distance parameter ensures minimal traversal through the containment zones if there is no safer alternative. Upon training the Q matrix based on these rewards, S-Nav considers the path that renders the maximum reward according to equation 6.

Lemma 1. *The Sigmoid function is concave for values greater than zero.*

Lemma 2. *The sum of two concave functions is a concave function.*

Since the Lemmas 1 and 2 are straightforward and based on basic mathematical principles, we refrain from providing proofs in this paper and maintain simplicity.

Theorem 3. *The reward associated with the selection of the edges in the road network map for reducing the passage through the categorical COVID-19 hotspots represented by equation 6 is a concave function.* □

Proof. The equation 6 is the summation of sigmoid functions in equation 1. Since the parameters in the reward function for the cells in the matrices are dependent on the distances between two vertices, the parameters for the sigmoid function are greater than zero. According to Lemma 1, equation 1 is a concave function, implying that equation 6 is a sum of concave functions. Using Lemma 2, we prove that the proposed objective function for S-Nav is concave. □

On training the model, we calculate P_{S-Nav} starting from source S and move to the next step. We represent the next intermediate steps as $S^{int}_{\mathcal{N}_{ind}}$, where ind represents the hop/step count in the matrix. It may be noted that ind is not the distance but the count of number of the number of vertices involved while calculating P_{S-Nav}. The selection of $S^{int}_{\mathcal{N}_{ind}}$ is based on the rewards on taking action $\mathcal{A}$. The model performs the set of actions while maximising according to equation 6 until it encounters the destination D during Q exploration. We represent P_{S-Nav} as $< V^*, E^* >$, which contains the vertices and edges of the recommended path by the S-Nav model. Algorithm 1 represents the steps involved in training S-Nav and algorithm 2 represents the steps for determining the P_{S-Nav} path.

Algorithm 1: S-Nav - Training

Input: epochs = iterations ; `// Number of iterations set according to number of nodes`
Result: All the possible paths are discovered and knowledge of the best path is attained by the trail and error approach.
for *epochs* **do**
  Select c_s randomly from Q-matrix;
  Select n_s from available states for c_s;
  Update the Q[c_s, n_s];
  `// According to Section 3.5`
end

Algorithm 2: S-Nav - Determination of P_{S-Nav}

Input: Current step (S^{curr}) = S ;
Result: P_{S-Nav} from S to D
Initialization: P_{S-Nav} = [S^{curr}] ; `// Initial point of the path is the source`
while *current_step != Destination* **do**
  add the index(s) of max_{row} in a next_index;
  if *len* ($S^{int}_{\mathcal{N}_{ind+1}}$) >=2 **then**
    choose $S^{int}_{\mathcal{N}_{ind+1}}$ randomly from the list;
  else
    $S^{int}_{\mathcal{N}_{ind}} = S^{int}_{\mathcal{N}_{ind+1}}$;
  end
  $\mathcal{P}_{safer}$.append($S^{int}_{\mathcal{N}_{ind}}$);
  $S^{curr} = S^{int}_{\mathcal{N}_{ind}}$;
  `// The next best state is found and added to` $\mathcal{P}_{S-Nav}$
end

Theorem 4. *The time complexity in determination of the safety-aware path in a geographic region by S-Nav is $\mathcal{O}(N^2)$, where N is the number of nodes.* □

Proof. S-Nav operates by maximizing equation 6. Finding j needs $\mathcal{O}(N)$ time because we iterate over all the entries belonging to the current row (i^{th}) and find the value of $max_i = j$. For N number of nodes in the map/graph (considering the worst case where $S = 1$, $D = N$, edges exists only between i and $i + 1 (i = \{1, 2, 3....N - 1\}$ and $max_i = i + 1$), S-Nav needs to iterate through N steps. Thus, the maximum reward is:

$$R_{max} = \sum_{i=1,j=max_i}^{N,N} Q[i, j] = \sum_{i=1}^{N-1} Q[i, i + 1] + Q[N, N]$$

The total time required in this case is $\mathcal{O}(N \times$ time for finding $j) = \mathcal{O}(N \times N) = \mathcal{O}(N^2)$ □

It may be noted that excluding roads/edges belonging to the COVID-19 hotspots may be a possible solution to avoid traveling through risky areas. However, there may be situations when no alternate route exists from a source to its destination. In such cases, on imposing the mentioned constraint, the S-Nav system will not recommend any route. Thus,

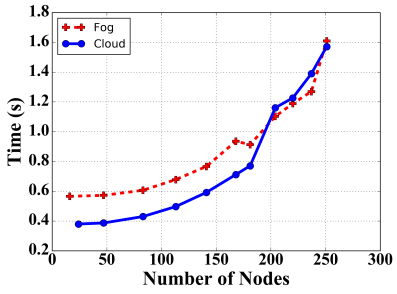


Figure 3: Time taken by FN and cloud devices for training the S-Nav system.

we refrain from applying it as S-Nav ensures no passage through the hotspots during the presence of alternate routes to the destination. Additionally, in cases when all alternate routes pass through hotspots, the zone-based rewards help in ensuring minimal passage through them.

4 Experimental Setup

We performed a series of experiments to evaluate the performance of S-Nav. Towards this, we used the road-network dataset exported from OpenStreetMap [21] and processed the data in a device with 1.8 GHz Dual-Core Intel Core i5 processor. We used different sets of longitudes and latitudes to vary the number of nodes and edges. We used a random distribution process to allocate the categorical hotspot zones in the network map. We used Google Colab to assume the role of cloud and the device mentioned earlier as fog nodes. It may be noted that the FNs may be further resource-constrained. However, we did not scale our results and presented them in its original form. While training the model for S-Nav, we fix the number of epochs/iterations sufficiently high to ensure that the model accurately explores the environment. We draw inferences and safety-aware paths after the training is complete.

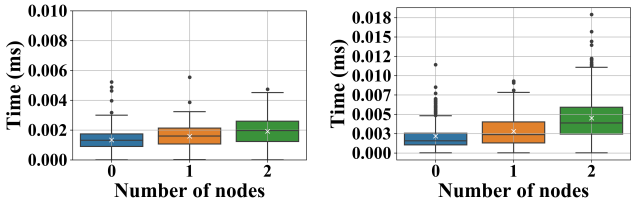
5 Results

In this section, we discuss our observations during the experiments for evaluating the proposed S-Nav system.

5.1 S-Nav training time

We calculate the delay incurred while training and testing S-Nav and present the results separately to get better insights. Fig. 3 illustrates the time necessary for training the S-Nav model in both cloud and fog devices. We vary the number of nodes by ranging it across $50 - 250$ nodes and record the time in each case. We observe that S-Nav training time at the FN is higher than that of the cloud by more than 50%. However, as the number of nodes increases, both the devices demonstrate similar delays. We attribute the lower delays to the superior CPU clock cycles and configurations of the cloud. On the other hand, the demonstration of similar delays as we increase the number of nodes is unlikely to occur in ideal conditions. Intuitively, we justify this phenomenon to the feature of context switching among multiple applications that execute on the cloud. Additionally, as we execute the code on Google Colab, we also attribute the additional delay to network latencies.

Implication: The delays give the impression of biasing the training at the cloud and then transfer the weights and matrices to the FNs. As the S-Nav system is a Q-based RL model, it may be easily trained in parts by the FNs with much lower latencies. This is the reason for adopting the network architecture in Section 3.1 for S-Nav.



(a) Fog node device. (b) Cloud device.

Figure 4: Time taken to calculate the S-Nav path by FN and cloud devices after training the S-Nav model.

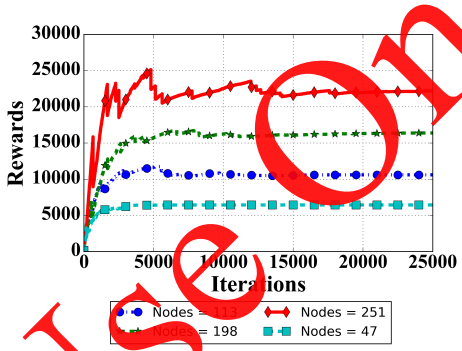


Figure 5: Reward values while training the S-Nav model with varying number of nodes in the map.

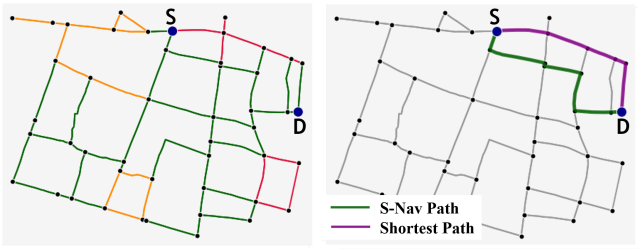
5.2 S-Nav testing time

We calculate the time taken by the FN and cloud devices for finding the path from the S-Nav model after training and present the results in Fig. 4. Interestingly, in the instant of performing our experiment, we observe in Fig. 4a that the FNs need less time to determine the path as compared to the cloud in Fig. 4b. One of the possible reasons for this is that the cloud servers are preoccupied with serving other requests. The response time is larger than the actual execution time, which dominates the overall delay.

Implication: Irrespective of the reasons for demonstrating higher pathfinding delays in the cloud server, we observe that the FNs offer comparative results. Such minuscule delays give us the motivation and justification towards deploying S-Nav in the FNs.

Table 1: Comparison of paths recommended by S-Nav (SN) and Shortest Path (SP) algorithm (Dijkstra's) in terms of travel distances and passage through the demarcated categorical COVID-19 hotspots.

Red Zone		Orange Zone		Green Zone		Extra
SN	SP	SN	SP	SN	SP	Length
0	0	17.73%	46.21%	82.27%	53.79%	17.03%
1.44%	65%	0	0	98.56%	35%	8.80%
44.57%	77.04%	0	0	55.43%	22.96%	17.42%
0	0	0	35.22%	100%	64.78%	17.16%
22.92%	26.39%	44.95%	61.07%	32.14%	12.54%	15.15%
21.39%	53.12%	0	0	78.61%	46.88%	3.94%
83.51%	83.51%	0	0	16.49%	16.49%	0.00%



(a) Road-network with demarcated roads (color-coded).

(b) Paths between Source S and Destination D.

Figure 6: Comparison of paths by S-Nav and shortest path algorithm in presence of COVID-19 hotspots

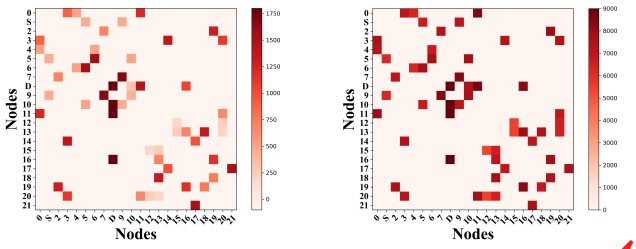
5.3 S-Nav rewards

We keep track of the rewards while training the S-Nav model and present the results in Fig. 5. We vary the number of nodes on the map and start training. We observe convergence in each case. Additionally, as we increase the number of nodes, the value of the rewards keep increasing. Interestingly, we observe that the number of iterations for attaining maximum rewards is proportional to the number of nodes. We measure the reward values by the sum of all the entries in the Q-matrix of the model. We notice the jumps and downs in Fig. 5 as the S-Nav model's training phase uses the trial and error approach. It learns from its mistakes and gains knowledge of the safer path. So, when the number of nodes is more, the number of edges is usually more. The model needs to explore all the possibilities and requires more number of iterations.

Implication: With the results in Fig. 5, we safely comment that the S-Nav system reaches convergence under all conditions. The S-Nav system offers correct solutions with minimum probabilities of making an error. In summary, the S-Nav system is reliable. It always fetches safety-aware paths irrespective of the number of nodes in the map.

5.4 Comparison of S-Nav with conventional techniques

Commonly available applications such as *Google Maps* rely on Dijkstra's algorithm and recommend shortest paths based on parameters explained in Section 1.1. Without loss of generality, we refer to the available methods as shortest path algorithms. In this section, we present a detailed analysis of a comparison between the paths/routes produced



(a) $\mathcal{R}$ matrix.

(b) Q matrix.

Figure 7: R and Q matrices while training the S-Nav system

by Dijkstra's algorithm (Shortest path) and the proposed S-Nav model. As an example, we consider a road-network of Kolkata (north, east, south, west = 22.5862, 88.3765, 22.5764, 88.3645). We assign contamination (hotspot) intensities to different areas of the dataset, as shown in Fig. 6a (color coded). Let S be the source, and D be the destination in the dataset. We execute both S-Nav and shortest path models and present the results in Fig. 6b. We observe that the S-Nav path (green) avoids the red zones/contamination areas and recommends passage through the green zones. On the other hand, the shortest path (pink) takes the route through the hotspots, which increases the commuter's threat of contracting the virus.

Although the S-Nav model finds the safest route possible, it increases the travel distance, implying a trade-off between safety and distance. On setting arbitrary source-destination pairs, we study the increase in travel distance and present our observations in Table 1. We observe higher travel distances by S-Nav in all cases. On average, we see almost 15% additional travel distance in the case of S-Nav as compared to the shortest path. However, we analyze the travel distance through hotspots (red/orange/green) and present the percentage with respect to each case's total recommended path. In both red and orange zones, we observe that S-Nav takes paths as small as 2% (15 m). The shortest path algorithm takes 65% on the same source-destination pair. In the case of green zones, we observe that the S-Nav model travels mostly through them compared to the shortest path model. In some unavoidable conditions like the one in the last row of Table 1, the S-Nav path is the same as the shortest path. We attribute that such a similar route decision occurs due to the absence of better alternative routes. In cases when alternate routes are available, we observe 100% passage through the green zones.

Implication: We safely comment that the S-Nav model ensures safe to travel in all source-destination pairs. However, it increases the travel distance, which is not a concerning factor as it reduces the risk of contracting the virus. In case the users seek shorter routes, we plan to modify S-Nav in our extended work to provide alternate paths (if any) with minimal thoroughfare through the hotspots.

5.5 S-Nav confusion matrix

We consider a map with 22 nodes and 55 edges. We demarcate some areas as red and orange zones to show the confusion matrices. We set the rewards according to

equation 1 and populate the $\mathcal{R}$ matrix of size 22×22. $\mathcal{R}$ is a symmetric matrix as we consider an undirected map. The destination node has the highest reward, and hence we observe dark patches in Fig. 7a. The lightest shade, which covers the majority of the matrix represents non-existing edges among the nodes. The other shades represent the rewards for each edge. The S-Nav system trains the model and populates the Q matrix in Fig. 7b. The patch with the darkest shade in the row represents the best possible next-state from the current state (row index). The values of the Q-matrix updates according to the Bellman equation.

Implication: The S-Nav system correctly updates its matrices, which elevates its reliability, implying that the routes recommended by S-Nav are correct and safe.

6 Conclusion

In this paper, we proposed and developed a reinforcement learning-based model (S-Nav) which recommends safety-aware road routes/paths. The recommended path by S-Nav avoids traveling demarcated categorical hotspots, ensuring safety, and reducing exposure risk for the commuters. To facilitate real-time results, we proposed an IoT-based network architecture by incorporating the cloud and fog computing paradigms. The fog computing platform allows partitioning of the maps and operations on small portions rather than the entire map, which is time-consuming. We also performed extensive experiments on S-Nav using real datasets from OpenStreetMap and presented results. We also performed a detailed comparative analysis of the recommended paths by S-Nav with Dijkstra's algorithm (shortest path). We observed a 15% increase in travel distances by S-Nav. However, its passage through the hotspots is minuscule.

In the future, we plan to extend this work by considering additional factors such as real-time traffic, multi-lane road system, and speed control. Further, we plan to incorporate user protection status by considering full, partial, or unprotected based on the inbound vehicle. We also plan to address the issue of source and destination being in geographical areas served by different FNs. Further, we plan to incorporate disconnected graphs and divide them into various relevant components.

References

[1] A. K. Tripathy, A. G. Mohapatra, S. P. Mohanty, E. Kougianos, A. M. Joshi, and G. Das, "EasyBand: A Wearable for Safety-Aware Mobility during Pandemic Outbreak," *IEEE Consumer Electronics Magazine*, pp. 1–1, May 2020.

[2] G. Ardonov and B. Yang, "Stochastic Shortest Path Finding in Path-Centric Uncertain Road Networks," in *in Proceedings of 19^{th} IEEE International Conference on Mobile Data Management (MDM)*, Jul. 2018, pp. 40–45.

[3] V. Chamola, V. Hassija, V. Gupta, and M. Guizani, "A Comprehensive Review of the COVID-19 Pandemic and the Role of IoT, Drones, AI, Blockchain, and 5G in Managing its Impact," *IEEE Access*, vol. 8, pp. 90 225–90 265, May 2020.

[4] S. P. P. Da Silva, J. S. Almeida, E. F. Ohata, J. J. P. C. Rodrigues, V. H. C. De Albuquerque, and P. P. R. Filho, "Monocular Vision Aided Depth Map from RGB Images to Estimate of Localization and Support to Navigation of Mobile Robots," *IEEE Sensors Journal*, pp. 1–1, Jan. 2020.

[5] F. A. X. Da Mota, M. X. Rocha, J. J. P. C. Rodrigues, V. H. C. De Albuquerque, and A. R. De Alexandria, "Localization and Navigation for Autonomous Mobile Robots Using Petri Nets in Indoor Environments," *IEEE Access*, vol. 6, pp. 31 665–31 676, Jun. 2018.

[6] W. Zhou and L. Wang, "The Energy-Efficient Dynamic Route Planning for Electric Vehicles," *Journal of Advanced Transportation*, vol. 2019, pp. 1–16, Aug. 2019.

[7] G. Szucs, "Decision support for route search and optimum finding in transport networks under uncertainty," *Journal of Applied Research and Technology*, vol. 13, pp. 125–134, Feb. 2015.

[8] J. Zhao, Y. Guo, and X. Duan, "Dynamic Path Planning of Emergency Vehicles Based on Travel Time Prediction," *Journal of Advanced Transportation*, vol. 2017, pp. 1–14, May 2017.

[9] P. Chen, X. Zhang, X. Chen, and M. Liu, "Path Planning Strategy for Vehicle Navigation Based on User Habits," *Applied Sciences*, vol. 8, p. 407, Mar. 2018.

[10] Y. Sun, X. Yu, R. Bie, and H. Song, "Discovering Time-Dependent Shortest Path on Traffic Graph for Drivers Towards Green Driving," *Journal of Network and Computer Applications*, vol. 83, pp. 204–212, Apr. 2017.

[11] M. Horvth, T. Matrai, and L. Tth, "Route Planning Methodology with Four-step Model and Dynamic Assignments," *Transportation Research Procedia*, vol. 27, pp. 1017–1025, Jan. 2017.

[12] X. Liu, D. Zhang, H. Yan, Y. Cui, and L. Chen, "A New Algorithm of the Best Path Selection Based on Machine Learning," *IEEE Access*, vol. 7, pp. 126 913–126 928, Sep. 2019.

[13] Y. Sun, H. Song, A. J. Jara, and R. Bie, "Internet of Things and Big Data Analytics for Smart and Connected Communities," *IEEE Access*, vol. 4, pp. 766–773, Feb. 2016.

[14] Q. Li, L. Zhao, J. Gao, H. Liang, L. Zhao, and X. Tang, "SMDP-Based Coordinated Virtual Machine Allocations in Cloud-Fog Computing Systems," *IEEE Internet of Things Journal*, vol. 5, no. 3, pp. 1977–1988, Apr. 2018.

[15] S. Ahmad, Imran, F. Jamil, N. Iqbal, and D. Kim, "Optimal Route Recommendation for Waste Carrier Vehicles for Efficient Waste Collection: A Step Forward Towards Sustainable Cities," *IEEE Access*, vol. 8, pp. 77 875–77 887, Apr. 2020.

[16] Z. Lv, B. Hu, and H. Lv, "Infrastructure monitoring and operation for smart cities based on iot system," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 3, pp. 1957–1962, Apr. 2020.

[17] X. Xu, L. Zhang, S. Sotiriadis, E. Asimakopoulou, M. Li, and N. Bessis, "CLOTHO: A Large-Scale Internet of Things-Based Crowd Evacuation Planning System for Disaster Management," *IEEE Internet of Things Journal*, vol. 5, no. 5, pp. 3559–3568, Oct. 2018.

[18] D. Shin, K. Yun, J. Kim, P. V. Astillo, J. Kim, and I. You, "A security protocol for route optimization in dmm-based smart home iot networks," *IEEE Access*, vol. 7, pp. 142 531–142 550, Sep. 2019.

[19] J. Zhu, Y. Song, D. Jiang, and H. Song, "A New Deep-Q-Learning-Based Transmission Scheduling Mechanism for the Cognitive Internet of Things," *IEEE Internet of Things Journal*, vol. 5, no. 4, pp. 2375–2385, Aug. 2018.

[20] A. Yousefpour, A. Patil, G. Ishigaki, I. Kim, X. Wang, H. C. Cankaya, Q. Zhang, W. Xie, and J. P. Jue, "FOGPLAN: A Lightweight QoS-Aware Dynamic Fog Service Provisioning Framework," *IEEE Internet of Things Journal*, vol. 6, no. 3, pp. 5080–5096, Jun. 2019.

[21] OpenStreetMap contributors, "Planet dump retrieved from https://planet.osm.org ," https://www.openstreetmap.org, 2018.