TSUNAMI EVACUATION GUIDANCE USING REINFORCEMENT LEARNING ALGORITHM

E. Mas [1], L. Moya[2,3], S. Koshimura[4]

[1] *Associate Professor, IRIDeS, Tohoku University, mas@irides.tohoku.ac.jp*
[2] *Researcher, IRIDeS, Tohoku University, lmoyah@irides.tohoku.ac.jp*
[3] *Principal Investigator, CISMID, National University of Engineering, Peru, lmoyah@uni.pe*
[4] *Professor, IRIDeS, Tohoku University, koshimura@irides.tohoku.ac.jp*

Abstract

In previous tsunami disasters it was reported that many residents used vehicles to evacuate from the coast. Consequently traffic congestions delayed evacuation times reducing survival chances in the area. Similarly, when evacuating as pedestrian, safe evacuations can be compromised by overcrowding conditions due to high population concentration, narrow streets and limited space for mobility. Thus, finding the best evacuation route considering less congestion and higher chances to reach safe haven is critical for the evacuation process.

We envision a future of smart and autonomous evacuations in a post-information age, where smart cities become fully interconnected and people or vehicles can be intelligently guided in real time with precise information aiming for an optimum behavior and a safe evacuation.

With this in mind, we built a model of evacuation with limited evacuee decision-making, however with learning-skilled connections within the network that would guide the flow of evacuees in directions that result best to reduce congestion and increase the chances to survive. One can imagine also smart and interconnected digital tsunami sign boards placed at intersections guiding evacuees accordingly to current congestion conditions in the network.

Here we used reinforcement learning to find the best policy (routing) constrained to low density in roads and successful guidance behavior of nodes in the network.

Keywords: tsunami evacuation, reinforcement learning, evacuation simulation

1. Introduction

Saving human lives in the case of a tsunami is the ultimate objective of disaster risk management. In this regard, one of the methods that is considered as the most important and effective is evacuation [1]. Thus, evacuation planning is one of the main activities conducted by local disaster managers and stakeholders. Unfortunately, In previous tsunami disasters it was reported that many residents used vehicles to evacuate from the coast and due to traffic congestions their evacuation was delayed leading to fatal outcomes. In small communities, were vehicles are seldom used in evacuation, pedestrian evacuations can also be compromised by overcrowding conditions due to high population concentration, narrow streets and limited space for mobility. Thus, finding the best evacuation route considering less congestion and higher chances to reach safe haven is critical for the evacuation process.

In the future, when smart and autonomous vehicles can perform evacuations for us, when we reach the post-information age, smart cities will become fully interconnected and people or vehicles can be intelligently guided in real time with precise information that leads the system to an optimum behavior and a safe evacuation.

Preparing ourselves for such future, we built a model of evacuation with limited evacuee decision-making and highly learning-skilled connections in the road network that would guide the flow of evacuees in directions towards less congestion and higher chances to survive.

We used reinforcement learning to find the best policy (routing) constrained to low density in roads and rewarding successful guidance. This paper reports on preliminary results and is organized as follows: a brief description of related work on the field, an introduction to the reinforcement learning framework and our evacuation model; the description of the numerical experiment and its results. Finally a discussion and conclusions on the current study.

2. Related work

Reinforcement Learning (RL) has been applied in various fields for purposes related to this work. For instance, [2] applied reinforcement learning to let pedestrians in the simulation learn the crowd motion obtained from classified video data in order to navigate avoiding obstacles. The RL approach here is inserted in the evacuee agent architecture of path planning. In other words, the evacuee learn to assess the environment of crowd motion to then design his evacuation strategy and local collision avoidance behavior. In contrast, [3] used deep reinforcement learning to model the fire evacuation in a building. In this case, the also reward/penalize the evacuee agent based on the action taken and the condition of fire spreading. Similarly, [4] developed a model using RL for path selection in disaster response management. They looked for the best strategy to go from a rescue team location to an affected area through a safe and shortest path. The RL algorithm was suitable to this case due to its ability to interact with the environment without needing any prior knowledge of it.

To the authors' knowledge, most of the efforts to incorporate RL in the evacuation modeling are concentrated on incorporating the learning process within the agent decision-making. This is understandable, since in multi-agent systems and evacuation modeling, intelligent and sophisticated behavior are necessary for the agent architecture when autonomous decision-making is expected. On the other hand, related work to our particular approach has not been found in the literature of evacuation modeling. Nevertheless, studies using a similar philosophy to ours can be found in the information science discipline. For instance, [5] developed a RL module embedded in nodes of a communication network to learn a routing policy for transmission of packets of information. Also, [6] compared this approach to shortest path algorithms and found that adaptive (learning) approaches perform better than tradition non-adaptive approaches.

3. Model framework

3.1 Reinforcement Learning Model

The RL deals with the policies of an agent to perform decisions/*actions*. The best policy is learnt from the interaction agent-environment. The *environment* refers to everything outside the agent that interacts with him. The agent knowledge is reinforced through more interaction with the environment. Consider the term *state* as all the information that the agent perceive from the environment at certain location and/or instant. In order to assess a policy, the parameter called *reward* is introduced to quantify the effect of an agent's action.

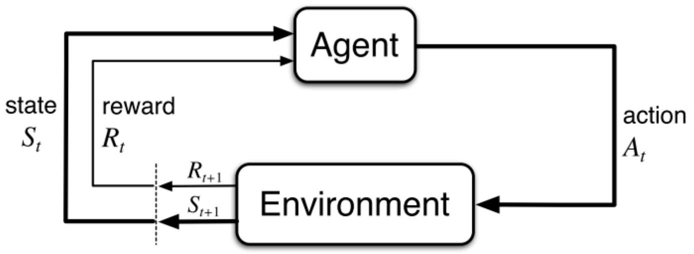


Fig. 1 – The agent from a Reinforcement Learning point of view. Image from [7].

Given these previous definitions, the mathematical structure/model of the agent-environment interaction is depicted in Fig. 1. It is described as a finite Markov Decision problem, from which the interaction is performed within discrete time steps. At certain instant *t*, the agent chooses a policy associated to the state s_t and performs an action A_t. The referred action will have repercussions in the environment, and consequently, it will influence the state at the instant $t+1$, s_{t+1}. Whether the new state is positive or negative to the agent's main objective, it is quantified with a reward r_{t+1}. The RL approach searches for the policy that gives the largest long-term reward, R_t, referred also as *return* and it is defined as follows:

$$R_t = \sum_{k=0}^{T} \gamma^k r_{t+k+1} \tag{1}$$

where γ is a parameter, $0 \leq \gamma \leq 1$, called *discount rate*; *T* denotes the end of the time from which interaction agent-environment finishes. It can be set to ∞, which will only converge to R_T if $\gamma < 1$. The best policy is chosen using the following optimization problem:

$$\max_{\pi} q^{\pi}(s,a) \tag{2}$$

$$q^{\pi}(s,a) = E_{\pi}\{R_t | s_t = s, a_t = a\} \tag{3}$$

where $q^{\pi}(s,a)$, referred as *value function*, is the expected return given that $s_t = s$ and $a_t = a$. Further details on how to solve Eq. (2) can be found in [7] and summarized in [8]. Here, we will describe the RL model accordingly to the aforementioned reference literature.

3.2 Tsunami Evacuation Model

The model developed in this study is based on the Monte Carlo method within the RL approach (MC-RL). Here, we estimate value functions and discover an optimal policy for the system through simulated experience. The system is based on a given road network *G(n,m)* with *n* nodes and *m* edges or links. Each node represents an intersection of two or more links (roads) or the endpoint of a boundary link (See Fig. 2).

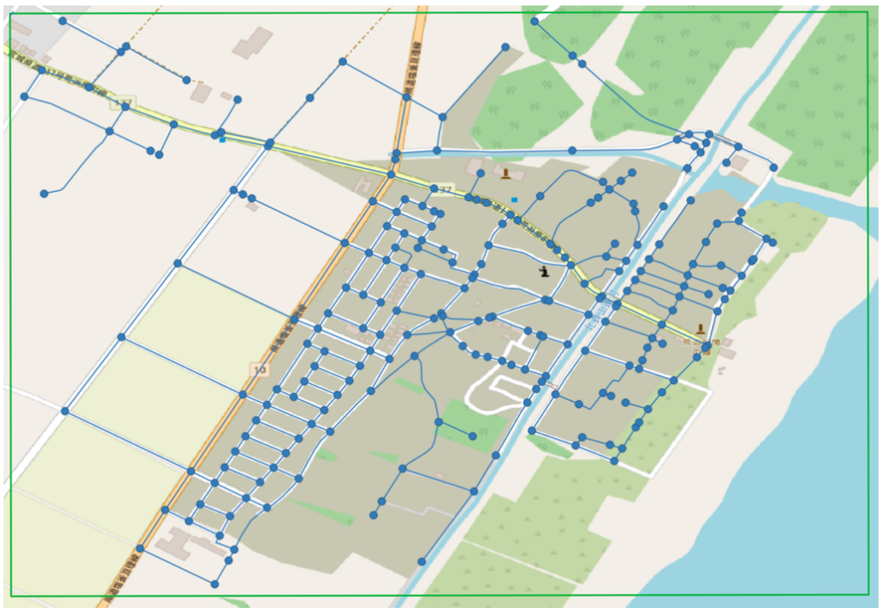


Fig. 2 – Example of a road network graph used in this study. The blue points are nodes and the blue lines are the links. The background image was retrieved from the Open Street Map archive.

Contrarily to other RL approaches in disaster research where the machine learning-based algorithm is located at the core of the moving and thinking agent (i.e. the robot, the evacuee, the responder or rescue team, etc.) [9]; in this case, we aim for the environment (the network) to be the smartest in the system and guide the evacuation process. Therefore, we set the nodes of the network as the learning agent – from here on *the node agent* - and the evacuee as part of the environment that imposes the constraints on the learning and the reward process for each node.

Let us explain in detail the components of the model. As stated before, the graph or network is the main component, and the population that consists on a fix number of evacuees trying to reach any available shelter is only part of the environment. Population spatial distribution is based on census data with original resolution of 100m grid. These data are disaggregated into the buildings and residential houses located within its corresponding grid. Finally, the individual population is reaggregated into the nearest node agent which is assumed as the initial position for evacuation.

As for the timing of evacuation, a Rayleigh distribution is used [10] to randomly allocated a starting time to each evacuee agent. When the model starts its clock, the evacuee agent will assess its timing for evacuation and when the time comes will start moving towards the next node.

We need to stress here that our model does not account for any decision-making by the evacuee agent nor any global knowledge of the entire system. The evacuee agent just follows the orders of the node agent.

In addition, crowding congestion is taken into account for speed adjustment based on values from [11]. Based on the graph of Figure 3, we divided three stages of level of congestion based on the density of evacuees at a link. Each level defines the moving speed in m/s that the evacuee will have at the entrance of a new link towards the next node. At the current stage, the moving speed is constant throughout the link and update again based on the information provided by the next node agent.

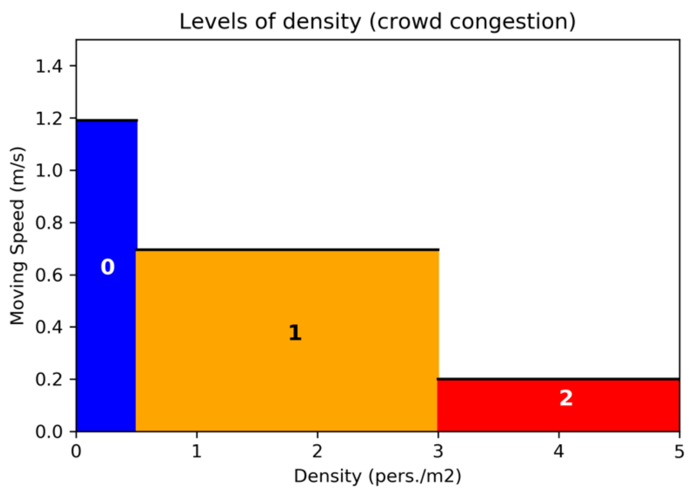


Fig. 3 – Moving speed and three levels of density reflecting crowding congestion conditions.

In the RL approach the main features of the environment are: (i) objective; (ii) state; (iii) action; (iv) return; (v) termination [12]. In our case we can describe these as follows:

- **Objective**: Guide evacuees at each node to the best direction to arrive safely at a shelter within the available time.
- **State**: An array of variable length which represents the density level at each link connected to the agent node. For example, [0,2,1] represents a node with three links with each of them at a 0, 2 and 1 level of crowd condition. The levels are defined in Fig. 3.
- **Action**: Selection of the next node based on the *Q(n)* matrix. For example, [0.50, 0.60, 0.75] represents the action-values where if an *exploit* behavior is selected, the maximum is chosen and therefore the evacuee agent follows the third link direction. On the other hand if and *explore* behavior is selected, a random link direction is chosen and new possible routes might be found or ignored based on trial-and-error throughout the MC simulation.
- **Reward**: No reward is given to the node/state within the episode; therefore the terminal rewards are also the returns.
- **Return**: From all the evacuees that crossed a node-agent, a return of +1 for each person that arrives at a shelter; otherwise, a return of zero is assigned. +1 for every node/state visited by the evacuee agent who reached safe haven. This decision-rule is the consequence of using $\gamma = 1$, and a reward/return $r = 1$ only if the action makes a person arrives at a shelter, otherwise, a reward/return of zero is assigned.
- **Termination**: A pre-set time *T* based on the expected tsunami arrival time.

The process of MC-RL can be summarized in the following pseudocode (See Algorithm 1). A predefined number of simulations (*N*) is defined. Here the term simulation is used to refer to episodes or repetitions which are common argot within the RL and the Monte Carlo approaches, respectively. First, a *State Matrix*, called the *Q(n)* matrix is defined. Where, *n* is the current simulation number. The state matrix contains all the states experienced by the evacuee agents. Recall that a state perceived by a node agent in this context is the level of congestion at each link that connects to himself. This is the most important matrix. It stores the action-value functions (an value calculated from state experience and rewards), which is used to judge which policy (next node to take) is the best option. Since *Q(0)* does not exist, unless a previous result is imported, the first state matrix is populated with random values. Otherwise the current *Q* matrix is imported from the previous simulation. A number of steps per simulation defined by a parameter *T*, related to the arrival time of tsunami, are set. For each evacuee, at each time step, the individual start time of evacuation (*STE*) is verified. Recall that initially we have set on each evacuee agent a random value pooled from a Rayleigh distribution with a certain mean value. When the time *t* equals the *STE* of the evacuee, this is classified as a 'moving agent' who consequently becomes a candidate for updating position and velocity in consecutive time steps. Finally, at time *T* the simulation stops and all evacuee agents who have reached safe haven reward the states visited in their path.

Algorithm 1 MC-RL pseudocode

```
 1: procedure MC-RL
 2:     $N \leftarrow$ Number of Simulations
 3: $PRECONDITION$: $n$ counter starts at 0
 4:     while $n < N$ do
 5:         if $n = 0$ then
 6:             Set $Q(0)$ matrix to random values
 7:         else
 8:             Import $Q(n-1)$ matrix
 9:         $T \leftarrow$ Number of of steps in one simulation
10: $PRECONDITION$: $t$ counter starts at 0
11:         while $t < T$ do
12:             for each evacuee agent $(A_i)$ do
13:                 if (*)$STE_{A_i} = t$ then
14:                     Set as moving evacuee agent
15:                 if moving evacuee agent? then
16:                     Update $(x, y)$ agent's position
17:                     if $(x, y) = (X, Y)$ of next node then
18:                         Update velocity and next node based on $Q(0)$
19:         Update values in $Q(n)$ matrix
20:         Export $Q(n)$ matrix
```

(*) STE = Start Time of Evacuation - A random number pooled from a Rayleigh distribution with a mean value that represents the time in mins for at least 50% of the population deciding to start evacuation

At the end of an episode/simulation, the components of *Q(n)* are updated in the following form:

$$Q_{i,j} \leftarrow \frac{Q_{i,j} \cdot N_i + R}{N_i + 1} = \frac{R - Q_{i,j}}{N_i + 1} + Q_{i,j} \tag{4}$$

Where $Q_{i,j}$ is the component of *Q(n)* matrix associated to a state *i* and an action *j*; N_i is the frequency or number of occurrences of rewards to this state; and *R* is the return value, in our case *R=+1*. Eq. (4) is performed for every node-agent and for every person that crossed it. The final step is to export the state matrix *Q(n)* of the current episode or simulation to become the input of a next run in the MC loop. Then, a subsequent episode will start learning from previous experiences already logged in the state matrix.

In the next section a numerical experiment is presented to discuss the feasibility of using the MC-RL approach to optimize the evacuation process by guiding agents through directions with higher chance of survival.

4. Numerical experiments

4.1 Configuration

We first use the census data provided by the Statistical Bureau of Japan on a spatial resolution of 100m grid. These data are disaggregated following an iterative proportional fitting method were the re-aggregated sums of initial estimations of residents per households and in buildings are adjusted iteratively to fit the original data and ancillary data such as the number of households and its number of family members or gender within the same spatial cell. Figure. 4 illustrates the original data format in the left and the result of the disaggregation on the right.

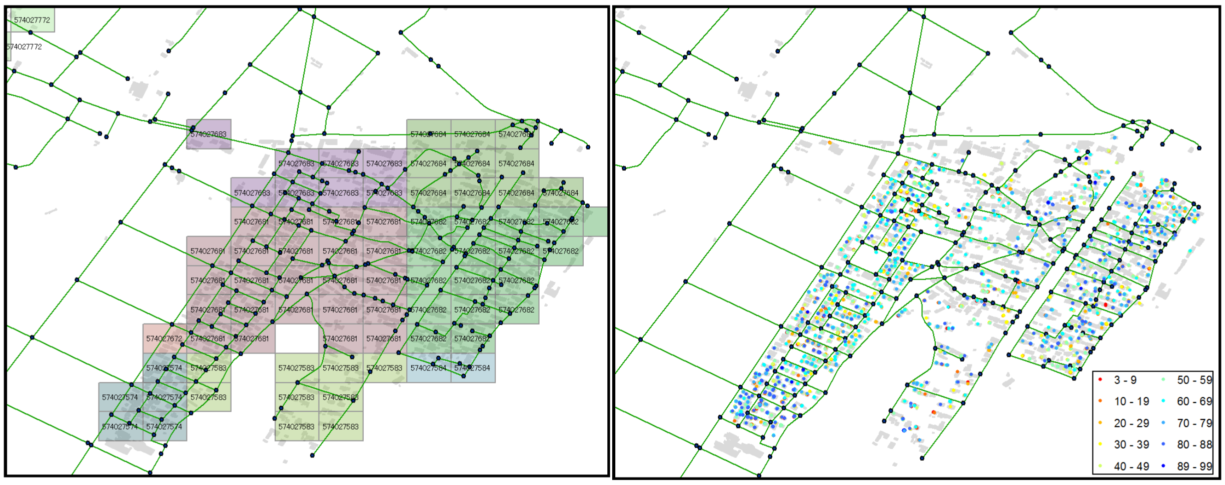


Fig. 4 – Population census data disaggregated into individuals at each residential building in the area of study.

As for other parameters and settings in the experiment, these are detailed in Table 1.

Parameter	Value	Unit	Reference
No. of Evacuee Agents	2,723	Pers.	Census Data
No. of Node Agents	240	Nodes	Road Network Data
No. of Links	312	Links	Road Network Data
Evacuee STE*	$\mathcal{R}(t,7)$	min.	Rayleigh Distribution [4]
Simulation time	67	min.	March 11, 2011 Japan Tsunami arrival time to the area
Number of Repetitions	1,000	run	Based on convergence of accumulated average values of evacuees in shelter per run

*STE: Start Time of Evacuation

4.2 Results analysis

The model was run 1,000 times to let the road network (node agents) to learn from positive outcomes and reinforce the guiding behavior on specific states. Figures 5 and 6 show the snapshots of the first and last run in the MC-RL simulation and evidence the learning effect in evacuation clearance. Figure 7 shows the accumulated average number of survivors from available runs (in red) and the number of survivors per run (in gray). In this experiment a 90% of the time the node guides the evacuee towards the maximum Q value (the best policy) and the remaining 10% of the time it shows an aleatory direction to explore the environment. Such balance between exploiting and exploring behaviors leads to the outcome shown in the figures.

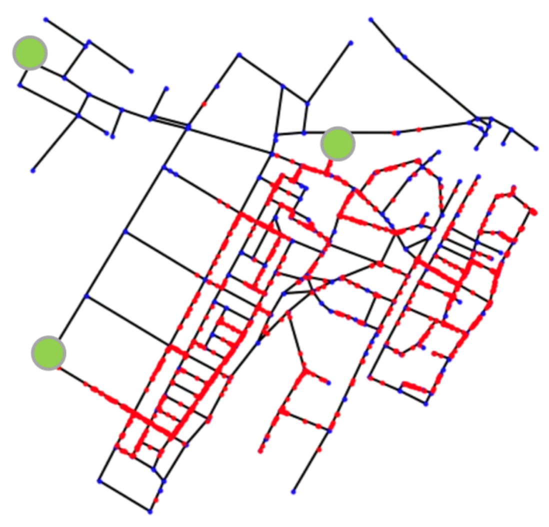
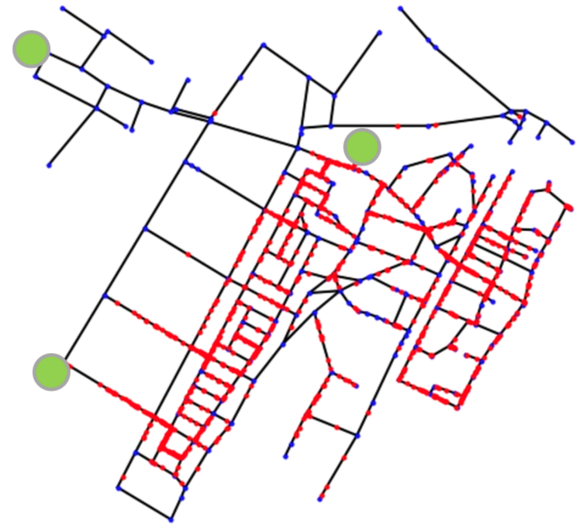
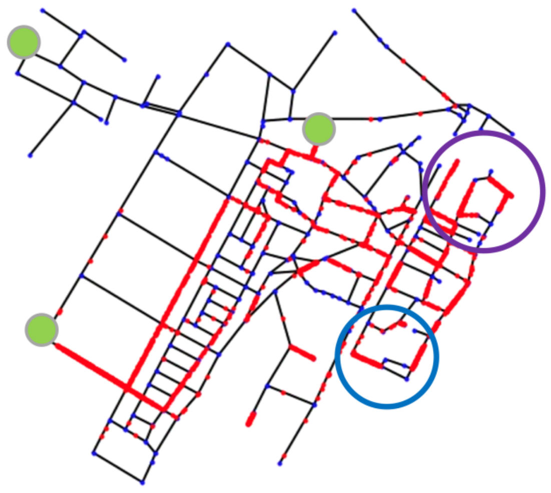
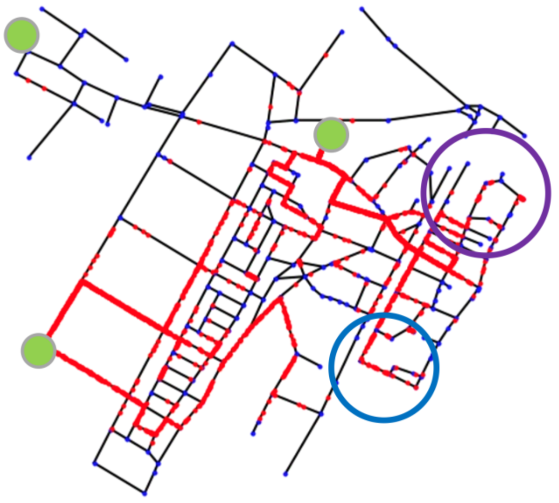
t (min)	**First Run (n=1)**	**Last Run (n=1,000)**
5		
15		

Fig. 5 – Snapshots of the first simulation and the 1,000th simulation at 5 and 15 minutes of evacuation process. Notice the lower congestion and use of alternative routes in the last run (right) compared to the first run (left)

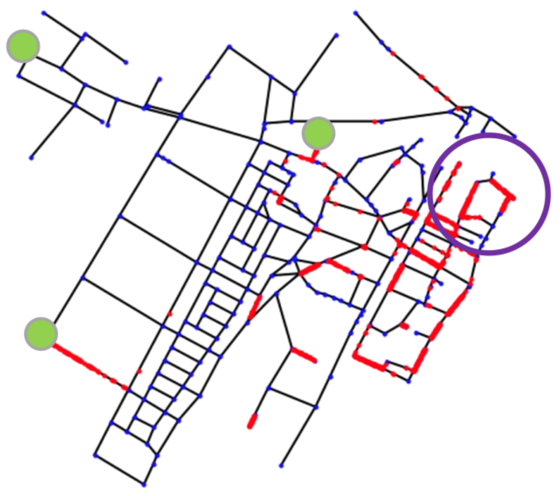
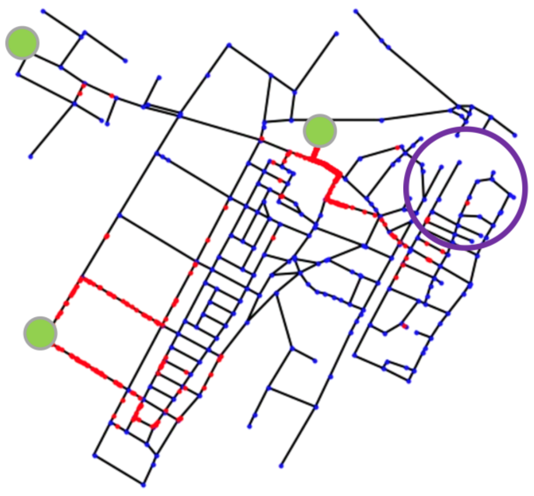
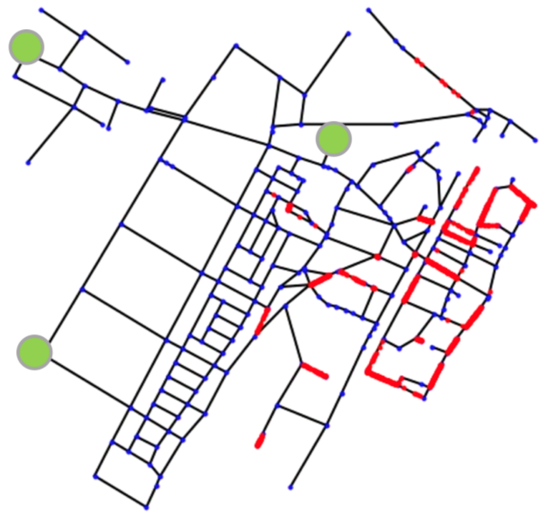
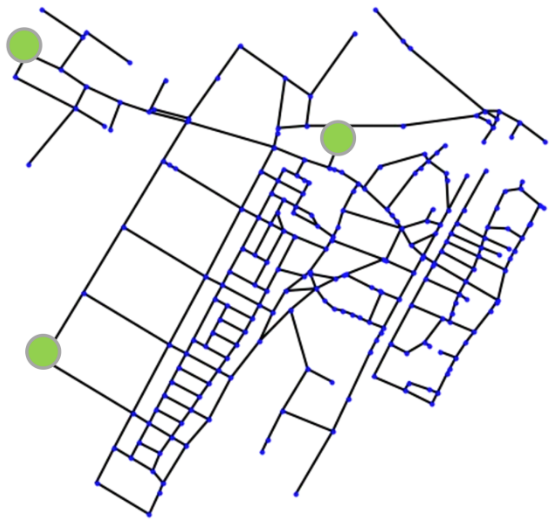
t (min)	First Run (n=1)	Last Run (n=1,000)
30		
60		

Fig. 6 – Snapshots of the first simulation and the 1,000th simulation at 15 and 30 minutes of evacuation process. Notice the lower congestion and use of alternative routes in the last run (right) compared to the first run (left). Also, the network has learned to guide evacuees faster with less congestion in the available time.

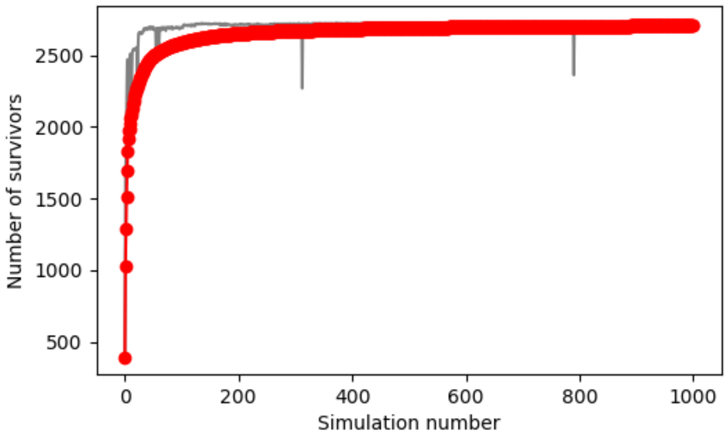


Fig. 7 – The number of survivors increments as the simulation runs progresses, This is, the network learns the best policy for each possible state and reinforces such guidance behavior. Finally a stable policy is found such that all evacuees can reach shelter in the available time. The gray line shows the survivors at each simulation while the red line shows the accumulated average of survivors from available runs.

5. Discussion

Based on Figures 5 and 6, we can argue that the network is effectively learning from previous experience, since a better policy is found at the last run. In addition, it is worth notice that the algorithm does not lead to the shortest path as the solution for the evacuation process. Due to the architecture of the model, where the speed is controlled by the congestion condition, and the reward is related to arriving or not, indirectly, we are rewarding paths that lead to safe haven regardless these are fast, slow, long or short. In other words, the system is trying to take advantage as much as possible of all the options (paths) available when the state (congestion) suggests that the best option is a particular direction (next node), even if this is not in line with the shortest path route.

We think this is optimum for a system of total compliance, meaning, where evacuees follow orders from nodes without hesitating or ignored them. As this might not be the real case in human behavior, it may be the condition of a futuristic evacuation where autonomous agents or robots will move based on a distributed coordination and organization looking for the optimum outcome for the system. Still, a long way forward in this regard is necessary to evaluate the efficiency, practicality, ethics and morality of such solutions. However, our cities are being developed based on interconnectivity, big data, technology and automation goals for everyday tasks. One of those tasks can be commuting or driving, therefore, a *disaster mode* behavior needs to be considered in advance.

6. Conclusions

We have developed a model of evacuation that uses the reinforcement learning algorithm and the Monte Carlo simulation approach. Our novelty is the consideration of a smart network that can learn from trial and error or simulated experience, the best policy to guide evacuees towards safe haven. Preliminary results show promising outcomes, such as an effective learning process by the network, an optimum solution for evacuation of all population in the area of study, and a distributed evacuation using all possible paths in the road network, leading to less congestion and faster evacuation. Further experiments for verification and improvement of the current model are necessary, including the comparison with traditional evacuation simulation methods to shed lights on the efficiency and applicability of the algorithms to the evacuation problem.

7. Acknowledgements

This study was partly funded by the JSPS Kakenhi Programs (No. 17H06108 and No. 18K18563) and MEXT as within the "Next Generation High-Performance Computing Infrastructures and Applications R&D Program"and the project "R&D of A Quantum-Annealing-Assisted Next Generation HPC Infrastructure and its Applications". Also we acknowledge the National Fund for Scientific, Technological and Technological Innovation Development (Fondecyt – Peru) within the framework of the "Project for the Improvement and Extension of the Services of the National System of Science, Technology and Technological Innovation" (contract number 038-2019). The authors thank the Core Research Cluster of Disaster Science at Tohoku University (a designated national university) for their support.

8. References

[1] Shuto, N. (2005). Tsunamis: Their coastal effects and defense works. In Scientific Forum on Tsunami, Its Impact and Recovery (pp. 1–12). Bangkok, Thailand: Asian Institute of Technology (AIT).

[2] Yao, Z., Zhang, G., Lu, D., & Liu, H. (2019). Data-driven crowd evacuation: A reinforcement learning method. Neurocomputing, 366, 314–327. https://doi.org/10.1016/j.neucom.2019.08.021

[3] Sharma, J., Andersen, P.-A., Granmo, O.-C., & Goodwin, M. (2019). Deep Q-Learning with Q-Matrix Transfer Learning for Novel Fire Evacuation Environment, 1–21. Retrieved from http://arxiv.org/abs/1905.09673

[4] Su, Z. P., Jiang, J. G., Liang, C. Y., & Zhang, G. F. (2011). Path selection in disaster response management based on Q-learning. International Journal of Automation and Computing, 8(1), 100–106. https://doi.org/10.1007/s11633-010-0560-2

[5] Boyan, J. A., & Littman, M. L. (1994). Packet Routing in Dynamically Changing Networks: A Reinforcement Learning Approach. Advances in Neural Information Processing Systems, 6, 671–678.

[6] Tekiner, F., Ghassemlooy, Z., & Srikanth, T. (2004). Comparison of the Q-Routing and Shortest Path Routing Algorithms. Proc. of the 5th Annual, 1–5.

[7] Richard S. Sutton, A. G. B. (2011). Reinforcement Learning An Introduction second edition. https://doi.org/10.1360/zd-2013-43-6-1064

[8] Pack Kaelbling, L., Littman, M. L., & Moore, A. W. (1996). Reinforcement Learning: A Survey. Journal of Artificial Intelligence Research 4, 237–285.

[9] Nguyen, L., Yang, Z., Zhu, J., Li, J., & Jin, F. (2018). Coordinating Disaster Emergency Response with Heuristic Reinforcement Learning.

[10] Mas, E., Koshimura, S., Imamura, F., Suppasri, A., Muhari, A., & Adriano, B. (2015). Recent Advances in Agent-Based Tsunami Evacuation Simulations: Case Studies in Indonesia, Thailand, Japan and Peru. Pure and Applied Geophysics, 172(12), 3409–3424. https://doi.org/10.1007/s00024-015-1105-y

[11] Takabatake, T., Shibayama, T., Esteban, M., Ishii, H., & Hamano, G. (2017). Simulated tsunami evacuation behavior of local residents and visitors in Kamakura, Japan. International Journal of Disaster Risk Reduction, 23(April), 1–14. https://doi.org/10.1016/j.ijdrr.2017.04.003

[12] Graesser, L., & Keng, W. L. (2019). Foundations of Deep Reinforcement Learning: Theory and Practice in Python. Pearson Education.