SMU
SINGAPORE MANAGEMENT UNIVERSITY

School of
Computing and
Information Systems

Reinforcement Learning of Entry Checks Strategies for Large-Scale Events

Cheong Song Le

Supervisor: Dr James Koh Boon Yong

Motivation & Objectives

Entry checks are commonly set up at large-scale events and it is important for organisers to deploy the right number of resources for these checks to ensure a smooth process for guests.

This project aims to devise an optimised operation strategy for the deployment of resources for entry checks.

The objective is to use reinforcement learning to train a model to take suitable actions to minimise the business costs which include

- Waiting time of guests for entry check
- Number of gantries required
- Change in operation strategies

Model Environment

The following details the model environment the project is looking at.

There are 9 sections of gantries, each with varying number of gantries. There are 2 types of guests — guests with ticket only and guests with bag. At every time period, at least one section of gantries has to be opened for each type of guest and each section can only be used for a type of guest at any one time.

A decision has to be made on the number of sections of gantries to be used for each type of guest in each period, without information of the arrival rates of the guests.

Subscripts:
a represents guests with ticket only
b represents guests with bag

<u>Formalisation of the model environment</u>

Action space: $[a_a, a_b]$

- Number of sections of gantries to open or close for each gantry type

State Observation: $[C_a, C_b, U_a, U_b, T]$

- C: Number of additional gantries opened
- U: Average queue level of guests
  * M_1 and M_2 are queue marker of a certain length
- T: The time period, 30-minutes each period

Average Queue Level (U)	Average Number of Guests in Queue
0	<=M_1
1	(M_1, $M_1+\frac{M_2-M_1}{2}$]
2	($M_1+\frac{M_2-M_1}{2}$,M_2]
3	>M_2

State Transition:

- Based on number of sections of gantries opened after taking an action
- Based on number of incoming guests, guests already in queue, guests cleared. The time taken for guests to be cleared is based on a distribution, unique for each type of guests.

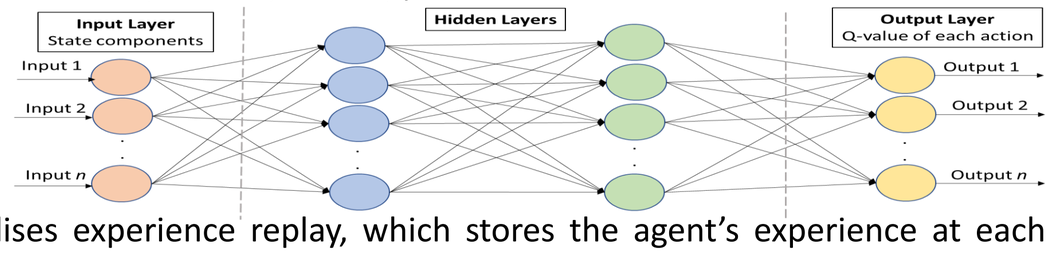
Reward Function: $r = -\beta(\Delta_a+\Delta_b)-\mu(N_a+N_b)-(H_a+H_b)$, where

$$H_x = c[\min(P_x, M_1 * N_x)] - d[\min(M_1 * N_x, \max(P_x - M_1 * N_x, 0))] - e[\max(0, P_x - M_2 * N_x)] - f(P_x > M_1 * N_x)] - g(P_x > M_2 * N_x), \quad x\epsilon(a,b)$$

- Δ: 1 if there is a change in number of sections of gantries opened, else 0
- N: Number of gantries opened
- P: Number of guests in queue
- $\beta, \mu, c, d, e, f, g$ are the penalty weightages associated relative to the cost of each guest in the queue

Reinforcement Learning Model

Deep Q-Learning (DQN) is an off-policy method that uses neural network to map input states to (Q-value, Action) pairs.



It utilises experience replay, which stores the agent's experience at each time step $e_t=(S_t, A_t, R_t, S_{t+1})$, where e is the experience of one step, S is the state, A is the action taken, R is the reward received.

The agent chooses an action based on an appropriate action-selection strategy. The states are then replayed for the reinforcement learning algorithm to learn from. DQN uses experience replay to learn in minibatches by sampling from the pool of stored experiences.

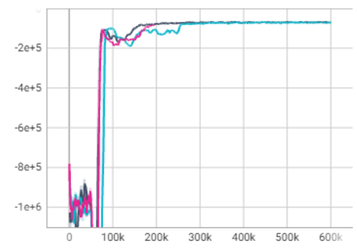
The agent will decide on an action to take, observe the transition of the environment, and receive an associated reward. Thus, the agent's objective is to take suitable actions to maximise the long-term cumulative reward.

Results

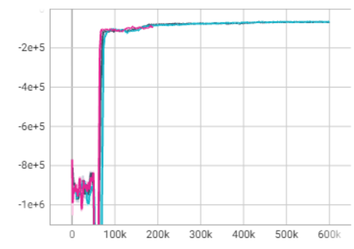
Stable Baselines3 (SB3) library was used for the implementation of the DQN algorithm.

Two environments with different action spaces were ran. Each experiment was repeated thrice to obtain a mean which has a lower variance as compared to running just once.

All the models were based on SB3's DQN algorithm with learning rate of 0.001 and gamma of 0.9. It is found that the total reward has converged after training for over 600,000 steps.



<u>Training total reward for Environment 1</u>



<u>Training total reward for Environment 2</u>

The models were evaluated on 48-period episodes. The results of the average total reward over 100 episodes is shown below.

Environment	Action Space	48-period average total reward
Environment 1	$a_x \epsilon (-1, 0, 1)$	-70,559
Environment 2	$a_x \epsilon (-2, -1, 0, 1, 2)$	-65,971

The results show that with a larger action space for the agent to learn from, it performed better, and saw an increase in the average total rewards. This result is reasonable as with a larger action space, the agent should be able take actions that are better or at least the same as an environment with a smaller action space.

Validation

To understand how well the trained model is performing, we evaluated the trained model with a current business operation strategy. The current strategy used to determine the number of sections of gantries to open for each period is very reactive, where a standard number of sections of gantries will be open during fixed peak periods of guest's arrival. During non-peak period, the operation strategy is to open one more section of gantry if the average queue length goes beyond M_1, and to close one section of gantry if the average queue length is within M_1.

Total reward from current operation strategy	Total reward from the trained model
-100,919	-65,971

Comparing the results, it is shown that the trained model is able to make intelligent decisions on the gantry action strategy and improve the total reward compared to the current operation strategy, effectively reducing the operation costs.

Furthermore, the decisions made for the current business operation strategy comes with experience which is not always guaranteed. With this model, event organisers without prior experience can utilise the model to decide on actions to be taken while achieving cost reduction to the business.

Conclusion

We developed a model that is able to achieve better total rewards than the current operation strategy. This met the objective of using reinforcement learning to train a model to learn the best policy for making decisions to minimise business costs and achieve an acceptable waiting time for guests.

This model is able to be adopted for different event scenarios by retraining the model based on the specific environment of the event.

Further work can be done to finetune the model and incorporate learning and retraining of agent in a non-stationary environment.