Simulation

This situation required a simulation to be created of a simplified airport security system at a busy airport. Passengers arrive according to a Poisson distribution with λ1 = 5 per minute to the ID/boarding-pass check queue, where there are several servers who each have exponential service time with mean rate of 0.75 minutes. After that, the passengers are assigned to the shortest of the several personal-check queues, where they go through the personal scanner (where time is uniformly distributed between 0.5 minutes and 1 minute).

The goal of this simulation was to find an optimal number of ID/Boarding Pass Checkers as well as Personal-Check Queues to keep average wait times below 15 minutes. The model with the lowest number of ID and Personal-Check Queues utilized while keeping average wait times below 15 minutes was retained, to balance airport resources with an efficient security system. The code for my approach can be seen below:

```
# Load Libraries
import simpy
import random as rd
import statistics as st
import numpy as np
import seaborn as sns

wait_time = []

# Environment Process
class airport(object):
    def __init__(self, env, id_checks, personal_checks):
        self.env = env
        self.ID_Queue = simpy.Resource(env, id_checks)
        self.Personal_Queue = simpy.Resource(env, personal_checks)

    # ID Check Distribution
    def check_ID_process(self, passenger):
        ID_time_rand = rd.expovariate(0.75)
        yield self.env.timeout(ID_time_rand)

    # Personal Check Distribution
    def personal_check_process(self, passenger):
        personal_time_rand = rd.uniform(0.5,1.0)
        yield self.env.timeout(personal_time_rand)

# Airport Process
def airport_process(env, passenger, airport):
    # passenger arrival
    arrival = env.now
```

```
    # id check
    with airport.ID_Queue.request() as request:
        yield request
        yield env.process(airport.check_ID_process(passenger))

    # personal check
    with airport.Personal_Queue.request() as request:
        yield request
        yield env.process(airport.personal_check_process(passenger))

    # wait time update
    wait_time.append(env.now - arrival)

# Run Airport
def run_airport(env, id_checks, personal_checks):
    Airport = airport(env, id_checks, personal_checks)

    for passenger in range(50):
        env.process(airport_process(env, passenger, Airport))

    while True:
        arrivals_int = rd.expovariate(2)
        yield env.timeout(arrivals_int)

        passenger += 1
        env.process(airport_process(env, passenger, Airport))

# Calculate Wait Time
def average_wait_time(wait_time):
    average_wait = st.mean(wait_time)
    return average_wait

# Run Simulation
def main(id_checks, personal_checks):
    # for reproducibility
    rd.seed(123)

    # Run the simulation
    env = simpy.Environment()
    env.process(run_airport(env, id_checks, personal_checks))
    env.run(until=600)

    return average_wait_time(wait_time)

# Test Cases
num_id_list = list(range(1,26))*25
num_personal_list = list(np.repeat(list(range(1,26)),25))

# Run on Test Cases, Return when Wait Time < 15
```

```
total_util = []
waits = []
for i in range(len(num_personal_list)):
    total_util.append(num_id_list[i] + num_personal_list [i])
    waits.append(main(num_id_list[i], num_personal_list [i]))

best_param_index = [i for i,x in enumerate(waits) if x < 15.0][0]

print('Lowest Combination: %d ID Queues & %d Personal-Check Scanners %(num_id
_list[best_param_index], num_personal_list [best_param_index]))
print('Wait Time: %d'%(waits[best_param_index]))

line = sns.lineplot(x=total_util,
             y=waits,
            ci=None)

line.set(xlabel = 'Total # of Queues Utilized', ylabel = "Average Wait Time (
in min)")

line.axhline(15, color = 'red', linestyle = ':')
```

```
Lowest Combination: 15 ID Queues & 14 Personal-Check Scanners
Wait Time: 14
```

As it can be seen, when 15 ID Queues and 14 Personal-Check Scanners were utilized, average passenger wait time was below 15 minutes. A plot of average wait time and total number of ID Queues and Scanners utilized can be seen below: