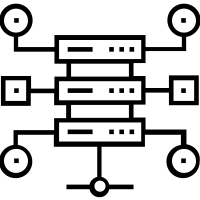
**Assignment No: 1**

**Date: 24th February, 223**

**Title: A Queuing Model of the Airport Departure Process**

(Title based on the application domain and the data structure you will be implementing)

|  |  |  |  |
| --- | --- | --- | --- |
| **Assignment Type of Submission:** |  |  |  |
| **Group** | Yes/No  Yes | **List all group members’ details:**  Student Name: Vivek Murarka  Student ID: 22200673  Student Name: Nikhitha Grace Josh  Student ID: 22200726  Student Name: Purvish Shah  Student ID: 22200112  Student Name: Ravi Raj Pedada  Student ID: 22200547 | **% Contribution**  **Assignment Workload**  20  20    20   20 |

****

1. **Problem Domain Description:**

The situation of airplanes having to wait in the runway queue for takeoff and landing has become a significant problem due to the required separation time between them. This waiting time leads to increased fuel consumption. To address this issue, an approach known as arrival/departure manager (AMAN/DMAN) has been suggested and put into operation at some of the world's major airports. However, the effectiveness of AMAN/DMAN is heavily dependent on the specific actions and algorithms it employs. Therefore, ongoing efforts to update and improve the algorithm are necessary to enhance the overall system's performance.

A queuing model for the airport departure process was suggested by Ioannis Simaiakis and Hamsa Balakrishnan at MIT. In this paper, they presented a queue-based approach through which we can observe the congestion at airport or for prediciting taxi-out time for individual flights.

The main focus of this paper is to anticipate airport performance on the assumption that the pushback schedules are already known. However, the paper does not explore how the uncertainty in the pushback schedules could affect the airport's performance.

**The Model:**

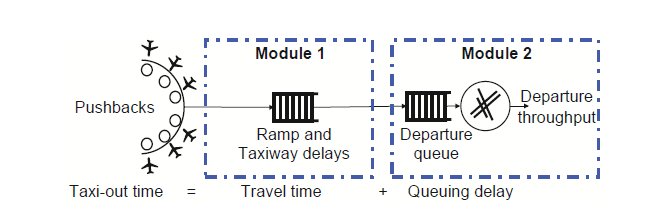


Figure 1Depature Process Model

Figure 1 illustrates the proposed model, which comprises two separate modules. The first module covers the process that occurs between the time when aircraft push back from their gates and travel to the runway. The second module focuses on the queuing process that occurs at the departure runway.

By modeling the departure process in this manner, the taxi-out time *Tl* of each departing aircraft l can be expressed as

*Tl* = *Tl*travel + *Dl* (1)

where *Tl*travel is the travel time of each departing aircraft l from its gate to the departure runway, and Dl is the queuing delay that aircraft l experiences.

Within the framework of this model, a pushback refers to the specific moment when an aircraft is physically separated from its assigned gate. Additionally, the taxi-out time is defined as the duration between the pushback and the actual takeoff of the aircraft.

Aircraft leave their gates as per the pushback schedule and move onto the ramp and taxiway system, and taxi to the departure queue which is formed at the threshold of the departure runway. While traveling, the aircraft have interactions with one another.

*Tl*travel, travel time of aircraft from their gates to their departure runway, is given by

*Tl*travel = *Tunimpeded* + *Ttaxiway* (2)

Here,

*Tunimpeded,* the unimpeded taxi-out time refers to the expected amount of time it would take for an aircraft to taxi out without any obstacles or delays. The Federal Aviation Administration (FAA) has defined this as the optimal taxi-out time that would occur when there is no congestion, inclement weather, or other factors that might cause the aircraft to be delayed during its journey from the gate to takeoff, as stated in the FAA's 2002 guidelines*.*

The Office of Aviation Policy and Plans of the Federal Aviation Administration (FAA) in 2002 employed a linear regression analysis to establish the unimpeded taxi-out time by examining the recorded taxi-out times along with the departure and arrival queues.

This average unimpeded time for an airline at given airport for a particular period can be downloaded from [FAA Operations & Performance Data](https://aspm.faa.gov/).

The second term, *Ttaxiway*, of equation(2), represents the delay that results from interactions with other aircraft that are traveling on the ramp and taxiway en route to the departure queue.

The paper suggests a linear equation to evaluate *Ttaxiway,* given by,

*Ttaxiway= αR(t)* (3)

The parameter α is determined by the airport and the layout of the runway, and it signifies the estimated frequency of times that an aircraft will come to a halt due to congestion, multiplied by the average duration of each stop. The likelihood of an aircraft stopping rises in proportion to the number of other aircraft present on the ramp and taxiway system en route to the departure queue at the time when the aircraft starts its pushback.

And *R(t)*, is given by, the quantity of aircraft that are departing and traveling on the ramps and taxiways towards the departure queue at the start of period t.

Combining equation(2) and (3) with (1), we get,

*Tl*travel= *Tunimpeded* + *Ttaxiway*

*Tl*travel = *Tunimpeded* + *αR(t)*

*Tl* = *Tl*travel + *Dl*

*Tl* = *Tunimpeded* + *αR(t)* + *Dl*

Another factor that we must consider is the time taken by the aircraft to travel from the departure gates to the runway queue. Further the gate from the runway more time it will take to travel, hence our final equation to generate the expected wheel off time is given by

*Tl* = *Tunimpeded* + *αR(t)* + *Dl* + *T*gate

**Queueing Model:**

The aircraft to depart from the airport, it needs to travel through two queues.

First is taxiway queue*.* Once a aircraft is pushed back from the gates, it enters to ramp and taxiway queue. This is a dynamic queue and doesn’t have a fixed length. As more aircraft enters this queue the size of the queue increases. There could be more than one plane which can be pushed back at same time. This creates a confusion that which aircraft enter this queue first and how long it needs to stay in this queue before it enters the second queue.

Once the aircraft enters the taxiway queue it goes through several stages of test which make it flight ready and then only starts heading to the depature queue. Some aircraft might take lesser time than others. There are other factors also which affect the time taken by the different airline carriers. Hence, some airlines might take lesser time than others. The mean time taken by different airlines at a given airport in different seasons is recorded by FAA and data is publicly available.

We consider this time along with the average time taken by aircraft to cover the from its gate to queue, to prioritize which aircraft enters the queue first.

There could be some aircraft already in this first queue which might have more expected wheel-off time than the incoming aircraft queue. Hence, we need to prioritize this new incoming aircraft in the queue depending on its expected wheel-off time. And push all the existing aircraft in the queue which has more expected wheel-off time and then recalculates their new expected wheel-off time, depending on the size of the queue.

The second queue is the departure queue, which is of finite length. If there is space available in the departure queue, then the aircraft with the highest priority in the taxiway queue leaves the taxiway queue and enters the departure queue. Aircraft in the departure queue are served on a First-Come-First-Served (FCFS) basis. If any aircraft ahead in this queue, could not depart from the runway at its expected time, it will delay the aircraft behind this. This is considered a queueing delay.

We record this queuing delay of each aircraft by subtracting the actual wheel-off time from the initially expected wheel-off time. The mean queueing delay gives the efficiency of the airport in handling the departure traffic. And this observation can be used to further reduce the congestion at the airport.

1. **Theoretical Foundations of the Data Structure(s) utilised**

To process the queueing model discussed earlier, we need two types of Queue data structures, first is the priority queue for taxiway queue and then is a normal queue for the departure queue, and then two list to store actual and expected wheel of time.

1. Queue

A queue is a linear data structure in which elements are inserted at one end (rear) and removed from the other end (front). It follows the FIFO (First In First Out) principle, which means that the first element inserted in the queue is the first one to be removed.

We use this queue to represent and process the departure queue in our queuing model. This queue is of finite length, as there is limited waiting are for a aircraft at the runway queue. The aircraft which entered first in this queue departs first, thus adhering to the FIFO principle.

A linked list can be used to implement a queue data structure, which is called a linked list queue or simply a queue using linked list. In a linked list queue, each element of the queue is represented as a node in the linked list.

The front of the queue is represented by the head of the linked list, and the rear of the queue is represented by the tail of the linked list. When an element is enqueued, a new node is added at the tail of the linked list, and when an element is dequeued, the node at the head of the linked list is removed.

The advantage of using a linked list as a queue is that it allows dynamic resizing of the queue as elements are added or removed. However, linked list queue operations such as enqueue and dequeue have a time complexity of O(1) only when the head and tail pointers are properly maintained. Otherwise, they can take O(n) time in the worst case.

1. Priority Queue:

A priority queue is a way to organize data so that the item with the highest priority can be accessed quickly and efficiently. It works similarly to a regular queue, but with the added feature of each element having a priority value assigned to it. In this type of queue, elements are arranged in order based on their priority value, with the highest priority element always placed at the front of the queue. As elements are removed from the queue, the next highest-priority element moves to the front.

Different types of data structures can be used to implement priority queues, depending on the specific needs of the application, including arrays, linked lists, binary heaps, and balanced binary search trees. Priority queues are especially useful in situations where items need to be processed in order of priority, such as in job scheduling, network routing, and simulation systems.

We used a linked list to create a priority queue, because we need to have a queue of variable length to store the aircraft data which are currently in the taxiway queue. Also, we need to frequently prioritize and re-order the queue as the data is inserted in the queue.

The queueing operation in priority queue.

1. List
2. **Analysis/Design (UML Diagram(s))**
3. **Code Implementation (please add your TA -** [**jason.walsh3@ucdconnect.ie**](mailto:jason.walsh3@ucdconnect.ie) **– as a collaborator)**

GitHub (link):

Set of Experiments run and results

Comments:

1. **Video of the Implementation running**

Zoom (link & password):

Comments:

**Please save as pdf and submit on Brightspace**

**Students belonging to the same group** please **submit the same file .**