Luggage Belt Models

Andrew Lim, Qiaoxue Liu, Qiubai Yu

June 18, 2019

1 Introduction

Waiting times for claiming luggage bags at airports are difficult to predict but provide crucial information in optimizing the time passengers spend at the airport. The random nature of arrival times and luggage distribution makes it particularly difficult to model waiting times at airports. We wanted a way to understand the distribution of waiting times so that we could provide better knowledge of luggage belt processes, though this required us to make many simplifying assumptions to our models. Using various simplified models, we looked at the distribution of a metric defined as T^1_{empty}, which is the first time (greater than zero) the luggage belt becomes empty. By focusing on discrete models of the luggage belt and passenger arrival times, we used combinatorial theory in identifying interesting sequences and distributions of T^1_{empty}.

2 Trivial case

- We first looked at a simple model where people were all standing in the same place closest to the luggage chute.

2.1 Assumptions

- The bags come out starting from time 1, in 1 second intervals. So, the second bag will come out on time 2, the third on time 3, and etc.
- People are all standing next to the luggage chute so they are able to pick their bag up as soon as it comes out onto the belt.
- All the people have arrived before the first bag comes out.
- Assume we have N bags and N people.

2.2 Expected Waiting Time

- We want to compute the expected waiting time of the ith person. Note that because the bags come out in a random permutation, the expected waiting time for each person will be the same. So the expected value can be computed as the following: The probability the waiting time is 1, is the probability that the ith bag comes out first which is $\frac{1}{N}$. The probability that the ith bag comes out second, is again $\frac{1}{N}$ and for any time that it comes out, its probability is $\frac{1}{N}$. Therefore, E["Waiting time of ith person"] $= 1 \cdot \frac{1}{N} + 2 \cdot \frac{1}{N} + 3 \cdot \frac{1}{N} + ... + N \cdot \frac{1}{N} = \frac{N(N+1)}{2} \cdot \frac{1}{N} = \frac{N+1}{2}$.
- Consider the case where the time intervals between baggage arrivals is C, where the first bag comes out at time C. Then we could generalize our computation of the expected waiting time of passenger i by the following: E["Waiting time of ith person"] $= C \cdot \frac{1}{N} + 2C \cdot \frac{1}{N} + 3C \cdot \frac{1}{N} + ... + NC \cdot \frac{1}{N} = \frac{N(NC+C)}{2} \cdot \frac{1}{N} = \frac{C(N+1)}{2}$.

3 Fixed Passengers, Random Luggage Order

3.1 Assumptions

- Each passenger only has one luggage
- The order of the passengers is fixed(denoted as P1, P2, ...)
- The luggage belt is infinitely long
- There are constant distance between the passengers
- It takes one unit of time for the bag to go from the first passenger to the second passenger
- All the possible orders of the luggages are with the same probability. (If we denote the total number of the luggage N, then any possible order is with probability $\frac{1}{N!}$)
- **The passenger moves very fast. Suppose Passenger 2 picks up his/her bag. S/He left instantly. In the Meantime, The next Passenger after him/her, let's say Passenger 3, will take over his/her place instantly.**
- Passengers are polite. Say bag 3 is at passenger 2. In this case, passenger 3 cannot push away passenger 2 to get his/her bag.

3.2 Definitions

- T^1_{empty}:The time at which the luggage belt firstly becomes empty

- **prefix** n: Let say there are c bags, where $c > n$. If we say the order of the bags is of prefix n, that is saying, The first n bags is a permutation of {1,2,...,n}. (e.g. The order {3,1,2,4} is of prefix 3)

3.3 Properties

- **If we say a certain order of bags is of prefix n, the T^1_{empty} of this order of bags is equals to n**

3.4 Computing Expected Value of T^1_{Empty}

Define a_k to be the total number of permutations we get for each $T^1_{empty} = k$, such that no prefix is a smaller term.
We want to find a formula for a_k in our first model for the luggage belt.

1. The original formula we derived for a_k:
   $a_k = k! - \sum_{i=1}^{k-1} a_i * (k-1)!$

2. Why OGF
   Note that in our first model, the formula we derived for the number of permutation of the bags when $T^1_{empty} = k$ has the recurrent term $\sum_{i=0}^{\infty} a_i * (k-1)!$ inside. We want to use the ordinary generating function to derive a formula for a_k.

3. OGF: A(x)
   Using OGF, we have $A(x) = \sum_{k=0}^{\infty} a_k * x^k$

4. Why we use $F(x) = \sum_{k\geq 0} k! * (x^k)$
   We first managed to express $k!$ in terms of a_k: $k! = \sum_{i=1}^{k} a_k * (k-1)!$
   Inspired by $(\sum_{i\geq 0} a_i * x^i) * (\sum_{j\geq 0} b_j * x^j) = \sum_{k\geq 0} c_k * x^k$.
   We then get $c_k = \sum_{i=0}^{k} a_i * b_{k-i}$
   Let $b_{k-1} = (k-i)!$,
   we have $\sum_{k\geq 0}(\sum_{i=1}^{k} a_i * (k-i)!)x^k = \sum_{k\geq 0} k! * (x^k)$.
   Define $F(x) = \sum_{k\geq 0} k! * (x^k)$.
   Manipulate algebra of the previous equation, we found a first order differential equation for F(x).
   The solution is $F(x) = \int \frac{e^{1/x}}{x} dx / (x * e^{1/x})$.

5. Use A(x) and F(x) to find the formula for a_k
   $A(x) = \sum_{k=0}^{\infty} a_k * x^k$
   $F(x) = \sum_{k\geq 0} k! * (x^k)$
   We found that $A(x) = 1 - \frac{1}{F(x)}$.

$\frac{1}{1-y} = \sum_{k\geq 0} y^k$, and $y = F(x) + 1$
Thus, $\frac{1}{F(x)} = \sum_{k\geq 0}(1 - F(x))^k$
Thus, $A(x) = 1 - \sum_{k\geq 0}(1 - F(x))^k$
$= 1 - \sum_{k\geq 0}(-\sum_{j\geq 1} j! * x^j)^k$
$= 1 - \sum_{k\geq 0}(-\sum_{j_1\geq 1} j_1! * x^{j_1}) * (-\sum_{j_2\geq 1} j_2! * x^{j_2}) * ...$
$= 1 - \sum_{k\geq 0}(\sum_{j_i\geq 1} j_1! * j_2! * ... * j_k! * (-1)^k * x^{j_1+j_2+...+j_k})$, with $\sum j_i = n$
Thus, $a_n = \sum_{j_i\geq 1} j_1! * j_2! * ... * j_k! * (-1)^k$

4 Nonlazy Model

4.1 Assumptions

The spacing between the bags and the spacing between people is equal.

1. Assumptions for bags:
   - The bags are single-spaced.
   - The bags are equally spaced.
   - The bags travels at one unit length(length of spacing) per second.
2. Assumptions for people:
   - People are single-spaced.
   - People are equally spaced.
   - People moves infinitely fast.
   - If a person has his bag, all the people behind him can move forward by one spacing immediately(no travel time).
3. Assumptions for luggage belt:
   - The belt is infinitely long.

4.2 Permutation Table

$T^1_{empty}\backslash n$	1	2	3	4	5	6	7	8	9
1	1	1	2	6	24	120	720	5040	40320
2		1	2	4	12	48	240	1440	10080
3			2	6	18	54	216	1080	6480
4				8	29	100	384	1536	7680
5					37	173	686	2940	13992
6						225	1217	5578	26412
7							1577	9896	50802
8								12810	90302
9									116812

4.3 Observations

1. $max\{T^1_{empty}\} = n$ (Upper Bound on T^1_{empty})
   - The permutation $n, n-1, n-2, ..., 2, 1$ must have the largest T^1_{empty}. Thus, $max\{T^1_{empty}\} = n$.
2. Defince $b_{k,2k-1}$ as $N(T^1_{empty} = k, n = 2k-1)$. Then, for any $n \geq 2k-1$, we have $b_{n,2k-1} = b_{k,2k-1} * \frac{(n-k)!}{(k-1)!}$
3. Define $I_{2k-1,k}$ as $I_{2k-1,k} = \frac{b_{2k-1,k}}{(k-1)!}$. Then we have $I_{2k-1,k} = 1, 2, 9, 64, 583...$, which happens to be $1^0, 2^1, 3^2, 4^3$. It seems that $I_{2k-1,k}$ has a nice pattern. However, $583 \neq 5^4$ and we still don't know why.

5 Double-spaced Nonlazy Model

5.1 Revised assumptions from Single-spaced Nonlazy Model

- The bags are double-spaced.

5.2 Variables of interest

- T^1_{empty}: The first time when the luggage belt becomes empty.
- n: The number of total people/luggages.
- N: Number of Permutation for each value of T^1_{empty}

5.3 Permutation Table

$T^1_{empty} \backslash n$	1	2	3	4	5	6
1	1	2	6	24	120	720
2	1	2	6	24	120	720
3		1	4	12	48	240
4		1	4	24	96	480
5			2	12	60	336
6			2	12	102	576
7				6	56	348
8				6	56	636
9					28	328
10					28	328
11						164
12						164

5.4 Some discoveries

- $max\{T^1_{empty}\} = 2 \cdot (n-1)$
- $N(T^1_{empty} = 1) = N(T^1_{empty} = 2) = (n-1)!$
- $N(T^1_{empty} = 2n-3) = N(T^1_{empty} = 2n-2)$
- $N(T^1_{empty} = 2n-5) = N(T^1_{empty} = 2n-4) = 2 \cdot N(T^1_{empty} = 2n-3) = 2 \cdot N(T^1_{empty} = 2n-2)$

5.5 Unsolved questions

- We still don't know how to explain the last two discoveries.

6 Appendix

Shown is the code that was written in CoCalc used to compute T^1_{empty} given any permutation that represents the order in which bags come onto the luggage belt.

```
#Counts the number of times 'number' appears in the list 'numList'.
def count_numberOf(numList, number):
    count = 0
    for num in numList:
        if num == number:
            count += 1
    return count

#Compares a list representing the people remaining around
#the luggage belt and another list representing the bags on
#the luggage belt and sets elements of the lists to zero if
#a person takes their bag.
def compareList(iterList, modList):
    for i in range(len(iterList)):
        num_zeros = count_numberOf(modList[0:iterList[i]],0)
        if iterList[i] != 0:
            if iterList[i] - 1 - num_zeros <= i:
                modList[iterList[i]-1] = 0
                iterList[i] = 0

#Simulates luggage model and computes T1_Empty
def t1Empty(bagOrder, personList, bagSpace):
    luggageBelt = []
    time = 0
    for bag in bagOrder:
        luggageBelt.insert(0,bag)
```

```
        time = time + 1
        compareList(luggageBelt, personList)
        if all(bags == 0 for bags in luggageBelt) == True:
            return time
        for iter in range(bagSpace - 1):
            luggageBelt.insert(0,0)
            time = time + 1
            compareList(luggageBelt, personList)
            if all(bags == 0 for bags in luggageBelt) == True:
                return time

#Inputs a list of permutations.
#Outputs the distribution of T1_Empty for all inputed
#permutations as a dictionary.
def t1Empty_dist(allbagOrder, bagSpace):
    distDict = {}
    for bagOrder in allbagOrder:
        personList = [1] * len(bagOrder)
        t1Emp = t1Empty(bagOrder, personList, bagSpace)
        if distDict.has_key(t1Emp):
            distDict[t1Emp] += 1
        else:
            distDict[t1Emp] = 1
    return distDict

#Outputs all permutations that are associated with
#a particular value of T1_Empty as a dictionary.
def t1Empty_permList(allbagOrder, bagSpace):
    permListDict = {}
    for bagOrder in allbagOrder:
        personList = [1] * len(bagOrder)
        t1Emp = t1Empty(bagOrder, personList, bagSpace)
        if permListDict.has_key(t1Emp):
            permListDict[t1Emp] = permListDict[t1Emp] + [bagOrder]
        else:
            permListDict[t1Emp] = [bagOrder]
    return permListDict
```