## 构建数学模型计算登机时间

### 常规状态

(P16)

In this model, we will calculate the total time of boarding. According to the discreteness of our model, this task can be reduced to finding a recursion formula for any passenger-based variable. Out of simplicity and authenticity concerns, we chose (velocity) as that valuable.

Before turning the spotlight on the analysis, we’ll first construct the space of cells and coordinates as shown on the slide.

(P17)

In addition, so far we are only looking into the regular cases where passengers move freely without being blocked. In this case, there doesn’t exist scenarios of contradictions such as .

(P18)

First, we’ll calculate the velocity according to the density. The density defined in the model is as shown in the slide. The visibility range is taken as because is not too big or too small and is realistic, and it also decides the time step (if taken as , the time step can be or sec), reducing the complexity of simulation.

(P19)

Next, e use Greenshields speed-density linear model (Ref. 3 in essay) to develop the relation between and .

(P20)

After that, we use dot products of vectors to calculate distribution according to density, as the slide shows.

(P21)

Additionally, by using partial summation, the distribution also has correspondent associations with the velocities.

(P22)

Therefore, we get the result.

Notice that previous calculations have shown that real-time speeds are associated under a linear bound. And we’ll also use two methods to justify our deductions: first on the next slide and then in the Sensitivity Analysis part.

(P23)

This is the recursion formula. It clearly displays the linearity. ( can be understood as the real-life speed.)

**Add diagram for gradual deduction**

### 拥挤状态

(P24)

As mentioned before, now we’ll come to the second scenario: when someone is causing a queue.

We divide the task into two parts: stowing luggage and offering seats. The first is trivial (but we’ll later add *discompliance* factors to this in the SA), and the latter can be calculated as shown mathematically – using permutation and the preservation of order.

(P25)

Here are the calculations.

(P26)

Here is the schematic diagram for this procedure.

### 状态的转换

(P27)

Now we’ll show the formula for the interconversion of states. The formulas here are further improved compared to our essay.

(P28)

Here are the ideal formula. It preserves the linearity.

(P29)

Deletion is rather trivial according to programmatic views.

### 结论部分

(P30)

Here we give the results. The weights can be calculated by accumulating all the and select the element. This can be easily be done with matrix multiplications:



(P31)

After obtaining all these indicators, we calculate the total time.

## 最优化

(P32)

In this part, we’ll focus on the modelling approach to optimizing (or minimizing) total time. Our work can be divided into two parts: inspiration from our previous calculations and the strict mathematical proof.

We’ll raise this concept named *parallelity* to describe how many of the aisle cells are occupied. Intuitively, the higher the parallelity, the more efficient the system is, and the faster the strategy is. The formulae of parallelity is shown in the slide.

(P33)

We’ll prove the intuitive idea proposed in the previous slide.

First, based on the model, we can do these analysis as shown.

(P34)

These properties are preserved by the linearity of our model.

(P35)

Secondly and mathematically, we’ll also prove this with two major claims. The first is about the optimality of all cells being occupied. The second will be useful when dealing with more complicated aircrafts.

(P36)

Here is Claim one. You can also refer to this in the essay. As the thesis has been shown before, we’ll not elaborate on this.