A Tale of Two Cities

Prof. Yishay Mansour, Prof. Amos Fiat, Mr. Lior Shultz

November 6, 2016

Abstract

We consider the following problem to achieve a basic understanding of taxi driver on-line decisions: We have two cities and jobs arrive one after another and the taxi driver can decide for each job if he wants to pick up the passenger or forfeit the job. Accepting a job yields the driver a reward of 1. However, accepting a job in a different city means that the driver will have to drive to the other city and pay 1 for the relocation - effectively earning nothing for the fare.

1 Notation

We will denote the following:

1. T - The number of total jobs received.
2. $1, 2$ - The two cities.
3. $p_{t,1}$ - The probability of the taxi being at city 1 at time t of our algorithm.
4. $p_{t,2}$ - The probability of the taxi being at city 2 at time t of our algorithm.

We will notice that always $p_{t,1} + p_{t,2} = 1$.

5. A turn is defined by first receiving a job request and then the decision as to how to act upon that request. We will denote the city the driver decided to go to in time t as d_t.
6. We will denote the job received at time t as $J_t \in \{1, 2\}$

2 Deterministic Algorithm

Theorem 1. *For any deterministic algorithm there exists a sequence for which the algorithm receives a reward of* 0.

Proof. If we consider ALG at time t we will notice that we can compute d_t since it only depends on $J_1, J_2 \cdots J_{t-1}$ thus we can simply create a new job J_t so that $J_t \neq d_t$. This way at time t the reward will be 0 and this can be done for every t gaining a total of 0 for the entire algorithm. □

Theorem 2. *For every sequence* $J_1, J_2 \cdots J_t$ *we get* $OPT > \frac{T}{2}$.

Proof. Since $|\{t|J_t = 1\}| + |\{t|J_t = 2\}| = T$ then either $|\{t|J_t = 1\}| \geq \frac{1}{2}$ or $|\{t|J_t = 2\}| \geq \frac{1}{2}$ and so OPT can just pick the one that is greater and remain there receiving a reward of at least $\frac{T}{2}$ □

These two theorems prove together that no finite competitive ratio can be achieved for any deterministic algorithm.

3 Online simple strategies

We will discuss the following two simple strategies:

1. $STAY$ - At the beginning we will use a coin to decide on a city and simply stay there for the rest of the algorithm. It is easy to see that this algorithm is expected to earn $\frac{T}{2}$.

2. $RAND\frac{1}{2}$ - Every turn of the algorithm if the job is in a different city we will flip a coin and move to that city with probability $\frac{1}{2}$. If the job is in the city we are already in we will of course accept it along with our reward.

4 Rand Analysis

Let's look at a possible way of analyzing the profit of the $RAND\frac{1}{2}$ algorithm. At time t we have a probability of $p_{t,1}$ of being in 1 at the beginning of the turn and probability of $1 - p_{t,2}$ of being in 2 at the beginning of the turn. This means that we can expect to earn $p_{t,1}$ and update the probability of being at 1 at time $t+1$ to $p_{t+1,1} = p_{t,1} + \frac{1}{2}(1 - p_{t,1}) = \frac{1}{2} + \frac{1}{2}p_{t,1}$ and the probability of being at 2 to $\frac{1}{2}(1 - p_{t,1})$. In the case of consistently alternating jobs (the job sequence consisting of $1, 2, 1, 2, 1, \ldots$) we can see that a balance is achieved when $p_{t,1} = \frac{1}{3}$. This is because we expect the tables to turn after the turn is finished due to symmetry meaning we expect $p_{t+1,2} = p_{t,1}$ and from this we get $p_{t,1} = p_{t+1,2} = \frac{1}{2}(1 - p_{t,1})$ and hence $p_{t,1} = \frac{1}{3}$. This means that in every step we will earn exactly $\frac{1}{3}$ achieving a total expected revenue of $\frac{T}{3}$ compared to the $\frac{T}{2}$ of OPT.

5 Rand Worse Case

The following sequence can get a worse competitive ratio for the $RAND\frac{1}{2}$ algorithm: $J_0 = 0, J_1 = 0, J_2 = 1$. This sequence is repeated many time in order to create the real sequence. It is easy to see that OPT can get $\frac{2}{3}T$. However, in an analysis similar to the alternating case we can see that $p_t, 0 = \frac{3}{7}$ at the beginning of the sequence. Then $p_(t+1).0 = \frac{5}{7}$ and then $p_(t+2), 1 = \frac{1}{7}$ yielding a total of $\frac{9}{7}$ per 3 turns giving us a competitive ratio of $\frac{14}{9}$.

6 Random Walk Analysis

It is easy to see that if we generate jobs with equal probability for each city any ON will get exactly $\frac{T}{2}$ in expectancy. If we wish to analyze OPT we have to assume that with probability $\frac{1}{2}$ the next job will be at the city we're already at and with probability $\frac{1}{2}$ the job will be at the other city. If the job is in the other city it means that we will not get any reward from it however we will get the reward for the job after that in OPT since if the job after is in the other city we will move there and get it (TO DO: prove that there is an OPT that does this and is equal to any other OPT) and if the job is in the city we're at we can just stay there and pass on the job. This means that with probability $\frac{1}{2}$ we will gain 1 in the next 2 turns. Now, to calculate the average we realize that with probability $\frac{2}{3}$ we are in one of the 2-step cycles (since they last for 2 steps instead of 1) and with probability $\frac{1}{3}$ we are in the 1-step cycle. This nets us a total of $\frac{2}{3} \cdot \frac{1}{2} + \frac{1}{3} \cdot 1 = \frac{2}{3}$. A total of $\frac{2}{3} \cdot T$.

7 The Optimal Algorithm

We will present and algorithm for the two cities problem assuming we have knowledge of all J_t and show that it is optimal:

ALG - The algorithm starts at J_1 and for every step if $J_{t+1} = d_t$ then we will of course stay at d_t and if $J_{t+1} \neq d_t$ then we will set $d_{t+1} = J_{t+2}$. In other words, if there is a job in a different city we will move there only if the job after that is in the other city as well.

Optimality Proof. We will show that for any optimal algorithm OPT at any point the relation to ALG will be one of the following states:

1. They have earned the same amount and are in the same city.
2. ALG is a point ahead of OPT.
3. They earned the same amount, are in different cities but the next job is in the same city as ALG.

Note that at no point in time could ALG have earned two points more than OPT since then in the next turn ALG can move to the same city as OPT and remain in a lead of at least 1. At this point if ALG takes exactly the same steps as OPT from that point on ALG will remain ahead of OPT contradicting its optimality.

It is easy to see that in the beginning we are in state 1 if OPT and ALG chose the same city and in state 3 otherwise (since ALG starts off in the city with the first job).

We will now proceed to show that any of the 3 states will lead to a different state as long as we follow the aforementioned algorithm.

state nr. 1: If both algorithms make the same decision then we will stay in state 1. If they make different decisions then if ALG decided to move then that means that the next job will also be in the other city meaning we will be in state 3 and if ALG stayed and OPT moved then the next job will also be where ALG is because otherwise ALG wouldn't have moved, thus we will also be in state 3.

state nr. 2: We will assume the worst case scenario where OPT earns 1 and ALG doesn't. This means that if they are in the same city afterwards then we will be in state 1. Otherwise, if they are in different cities then we are necessarily in state 3 because if the next job is in the city where OPT is then ALG would have moved there in the first turn in contradiction to them being in different cities.

state nr. 3: It is easy to see that next turn ALG earns 1 while OPT earns 0 meaning we will necessarily be in state 2.

Since in all states ALG has earned at least as much as OPT this means that at no point OPT will be able to surpass ALG meaning that ALG is also optimal. □

8 The K-City Problem

In this section we will examine the same problem except now we will assume k cities instead of 2. The cities will still be numbered 1 through k.

9 The Optimal Algorithm

This is the general case of the optimal algorithm for 2 cities. What remains to be decided is when we move from d_t to J_{t+1}. If they are equal then there is no decision to be made. If they are not the same then we move if $\min(i|i > t+1 \wedge J_i = J_{t+1}) < \min(i|i > t+1 \wedge J_i = J_t)$, meaning we will choose the city where we will have to wait the least time to get another job.

TODO: optimality proof in this case is very similar to the 2 cities case except state nr. 3 needs to be changed to the closer city in time.

10 Rand Walk Analysis

It is easy to see that if we randomly generate a job sequence any ON will get $\frac{T}{k}$ in expectancy. In order to analyze the earning expectancy of OPT we will look at the amount of time needed for OPT to earn 1. This means that in the interval we expect to see some city twice and since the way OPT works is that the first time this occurs we will manage to earn 1 and we have k cities according to birthday problem related calculations this is $O(\sqrt{k})$. Meaning that the algorithm will have earned $O(\frac{T}{\sqrt{k}})$ in expectancy.

TODO: write it properly. Get a more accurate estimation for birthday problem...