Plane-hotel problem: On the competitive-ratios of some deterministic and randomized online algorithms

T. Grimbergen, A. Sorbo, D. van Dongen, T. Post, J. Flikweert
Utrecht University ICS, The Netherlands

Abstract

[Abstract text here]

2012 ACM Subject Classification Theory of Computation

Keywords and phrases Online algorithms, competitiveness, competitive analysis, randomized algorithms

Digital Object Identifier 10.4230/LIPIcs...

1 Introduction

In this research, we will investigate the online ticket-hotel problem. In this problem, there are n people that have to be flown home in m days. For each day $i \in \{1, \ldots, m\}$ the algorithm receives the number of available plane tickets s_i, the price of a ticket p_i, and the hotel price h_i. Every person who is not sent home by plane must spend the night in a hotel. The algorithm must then decide irrevocably how many people it will send home f_i. In Section 2 we will elaborate on the different constraints put on the parameters above that are analyzed in this work.

It can be shown (see Appendix A) that no constant competitive online algorithms exist without certain constraints on this problem description. Specifically, it turns out that some guarantee on the number of available seats on any given day should be given and the prices p_i that the online algorithm receives should be drawn from a bounded interval.

It is crucial for the competitive analyses of online algorithms to know the optimal offline solution for any given instance. For this reason, we first propose the greedy algorithm in Section 3 and provide proof of correctness.

We will then analyze several deterministic algorithms in Section 4. A formal analysis is done on (at least) two algorithms. For the simple threshold algorithm $\texttt{ALG}_2$ we show that it is $\mathcal{O}(\sqrt{p_{\max}})$-competitive under the constraints that $h_i = 0$, and for the more sophisticated multi-threshold algorithm $\texttt{ALG}_4$ we show that it is also $\mathcal{O}(\sqrt{p_{\max}})$-competitive where the constraint $h_i = 0$ is relaxed. Then we (may) show that $\texttt{ALG}_2$ is optimal if we impose the extra restriction $n = 1$, but that it is not optimal for $n > 1$. We might suggest algorithm $\texttt{ALG}_3$ and show that this is optimal.

Furthermore, we will extend a deterministic algorithm with a random variable in Section 4, to make the process stochastic instead of deterministic, with the goal of improving its expected value. We will then compare the two algorithms and see which performs better under what circumstances, as an alternative way of analysis.

Finally, in Section 6 we will perform an empirical analysis of the algorithms to find their average performance and, if applicable, to check that the competitive ratios are indeed bounded by the results derived in Section 4.

2 Preliminaries

In this work we will work with several constraints on the general problem. In all cases we will have the $n, m \in \mathbb{N}$[1] for the number of people n and number of days m. In Table 1 we distinguish between four different constraint settings (a)-(d) on the available seats s_i, ticket price p_i and hotel price h_i. We then define an instance and the set of all instances.

Table 1 Table listing the different settings used in this work. Note that for setting (c) we use $p_{\max} \in \mathbb{N}$ for both the constraints on p_i and h_i.

	s_i	p_i	h_i
(a)	$\in \mathbb{N}_0$	$\in \mathbb{N}$	$\in \mathbb{N}_0$
(b)	$\geq n$	$\in \mathbb{N}$	$\in \mathbb{N}$
(c)	$\geq n$	$\in \{1, \ldots, p_{\max}\}\, p_{\max} \in \mathbb{N}$	$\in \{1, \ldots, p_{\max}\}$
(d)	$\geq n$	$\in \{1, \ldots, p_{\max}\}\, p_{\max} \in \mathbb{N}$	$= 0$

▶ **Definition 1.** *Let $n, m \in \mathbb{N}$, $\mathbf{s}, \mathbf{p}, \mathbf{h} \in \mathbb{N}_0^m$ and let $x \in \{a,b,c,d\}$. We define $I_x := (n, m, \mathbf{s}, \mathbf{p}, \mathbf{h})$ to be an* instance in setting (x) *if the parameters $n, m, \mathbf{s}, \mathbf{p}, \mathbf{h}$ satisfy all of the constraints of setting (x) listed in Table 1). The* set of all instances in setting (x) *are defined as $\mathcal{I}_x$.*

Throughout the paper, we use the term *average competitive ratio* of an online algorithm. For this we introduce the following definitions which are inspired by Eq. (2) of [1]).

▶ **Definition 2.** *Let $\epsilon > 0$ and let I be an instance. We then run the online algorithm $\mathtt{ALG}$ a total number of $N > 1$ times where N is determined by*

$$\left| \frac{\frac{1}{N}\sum_{i=1}^{N} \mathtt{ALG}(I)}{\mathtt{OPT}(I)} - \frac{\frac{1}{N-1}\sum_{i=1}^{N-1} \mathtt{ALG}(I)}{\mathtt{OPT}(I)} \right| < \epsilon.$$

We then define the average competitive ratio of an instance *as*

$$\hat{c}(I) = \frac{\frac{1}{N}\sum_{i=1}^{N} \mathtt{ALG}(I)}{\mathtt{OPT}(I)}.$$

In this paper we will use $\epsilon = \ldots$ (t.b.d). Note that for a deterministic online algorithm the cost $\mathtt{ALG}(I)$ is always the same, thus in this case the average competitive ratio is just $c\hat{(I)} = \frac{\mathtt{ALG}(I)}{\mathtt{OPT}(I)}$. However, for randomized algorithms the value $\mathtt{ALG}(I)$ is not necessarily constant. In this case this definition is meaningful. A drawback is that the average competitive ratio will likely not be the same each time you do this procedure.

▶ **Definition 3.** *Let $\delta > 0$ and let $J = \{I_1, \ldots, I_M\}$ be $M > 1$ uniformly drawn samples I_i from the allowed parameter space where M is determined by*

$$\left| \frac{1}{M}\sum_{i=1}^{M} \hat{c}(I_i) - \frac{1}{M-1}\sum_{i=1}^{M-1} \hat{c}(I_i) \right| < \delta.$$

We then define the average competitive ratio of an algorithm $\mathtt{ALG}$ *as*

$$\hat{c} = \frac{1}{M}\sum_{i=1}^{M} \hat{c}(I_i).$$

[1] We define $\mathbb{N} := \{1, 2, \ldots\}$ the set of strictly positive integers and $\mathbb{N}_0 := \{0\} \cup \mathbb{N}$.

In this paper we will use $\delta = ...$ (t.b.d). Note that this definition is meaningful for both deterministic and randomized algorithms. Also, the drawbacks of Definition 2 are also applicable to Definition 3.

3 Optimal offline algorithm

Algorithm 1 The greedy (optimal) offline algorithm `OPT`

```
function OPT(I)                          ▷ Returns optimal cost for instance I
    n, m, s, p, h ← I
    T, R ← ∅
    total ← 0                            ▷ total cost of solution
    for j ← 1 to m do
        if j > 1 then
            T[j] ← p[j] + Σ_{1≤i<j} h_i
        else
            T[j] ← p[j]
        end if
    end for

    sort T ascending on value            ▷ sort on cheapest days to send people back
    while n > 0 do
        (j, c) ← T.shift()               ▷ get (key,value) from, and remove first element in T
        q = min{s_j, n}                  ▷ number of people to send back
        R[j] ← q                         ▷ store number of tickets for day j
        n ← n − q                        ▷ store new n
        total ← total + (c ∗ q)          ▷ update total cost of sending everyone back
    end while
     return total
end function
```

We then prove the following theorem.

▶ **Theorem 4** (Offline greedy is optimal). *Consider the offline algorithm 1 above. The following statements are true.*

1. *The cost of this algorithm is calculated as:*
2. *The greedy algorithm always provides a feasible solution.*
3. *This solution is optimal.*

Proof. We almost have this. ◀

4 Deterministic online algorithms

We first propose the simple threshold algorithm ALG$_2$:

Algorithm 2 The $\lfloor\sqrt{p_{\max}}\rfloor$-threshold online algorithm ALG$_2$

```
Q ← ⌊√p_max⌋
for i ← 1 to m do
    if (p_i ≤ Q ∨ i = m) then
        Buy n tickets.
    end if
end for
```

We will show that the following result holds for the competitive ratios of ALG$_2$.

▶ **Lemma 5** (Competitive ratios for ALG$_2$). *Let $p_{\max} \in \mathbb{N}$. Then write $p_{\max} = k^2 + \ell$ for $k, \ell \in \mathbb{N}$ with $0 \leq \ell < 2k + 1$. Then, in setting (d), the algorithm ALG$_{\sqrt{p_{\max}}}$ is*

- $\lfloor\sqrt{p_{\max}}\rfloor$*-competitive if* $0 \leq \ell \leq k$,
- $\frac{p_{\max}}{\lfloor\sqrt{p_{\max}}\rfloor+1}$*-competitive if* $k < \ell < 2k + 1$.

Proof. We have this proof. ◀

If we consider the subset of instances of $\mathcal{I}_d$ such that $n = 1$, then we can even prove that ALG$_2$ is in fact an optimal algorithm for these instances. However, by considering the simple example instance with $(n, m, p_{\max}) = (2, 2, 4)$ we observe that if $p_1 = 2$, the optimal choice that minimizes the competitive ratio is *not* to buy 2 tickets on day 1, but to buy 1 ticket. In the former case, the worst case competitive ratio c considering all possible p_2 will be $c = 2$, while in the latter case we have $c = \frac{3}{2}$. Therefore, the algorithm ALG$_2$ fails in achieving the smallest possible competitive ratio for some instances, while clearly there exists an online algorithm that does make the optimal choice. So, ALG$_2$ is definitely not optimal for all $n \in \mathbb{N}$. We then propose the following "greedy" online algorithm ALG$_3$.

Algorithm 3 A greedy online algorithm ALG$_3$

```
CC ← ⌊√p_max⌋                                   ▷ CC := Current Cost
p_min ← p_max                                   ▷ p_min := Current smallest price
for i ← 1 to m do
    if (i = m) then
        Buy n tickets
        return
    end if
    p'_min ← min{p_i, p_min}
    n_i ← argmin_{0≤x≤n} { max_{1≤p_{i+1}≤p_max} { (CC + p_i x + p_{i+1}(n − x)) / (min{p_min, p_{i+1}} n) } }
    Buy n_i tickets.
    n ← n − n_i
end for
```

The algorithm ALG$_3$ is based on the following heuristic. When deciding how many tickets n_i to buy on any given day i, we want to choose the value that *minimizes* the competitive ratio under the assumption that the adversary will try to maximize the competitive ratio on the following day. In setting (d), the only parameter over which the adversary has control is the price p_{i+1}, hence we only have to maximize over this parameter. An important observation is that in computing the value n_i we assume that we have to buy all tickets on either day i or on day $i+1$.

We plan on at least proving a lemma along the lines of

▶ **Lemma 6** (Competitive bound). *The algorithm $\mathtt{ALG}_3$ is $\mathcal{O}(\sqrt{p_{\max}})$ competitive.*

Proof. This might be hard. ◀

Also, we would modify the calculation of n_i in the algorithm $\mathtt{ALG}_3$ to be more computationally efficient by analyzing the function

$$f : \{1, \ldots, p_{\max}\} \to \mathbb{R}, \quad p \mapsto \frac{CC + p_i x + p(n-x)}{\min\{p_{\min}, p\} n}.$$

Finally, we propose the algorithm $\mathtt{ALG}_4$ which is specifically designed for variant (c) of the constraints ($h_i \in \mathbb{Z}_{\geq 0}$).

Algorithm 4 The greedy online algorithm $\mathtt{ALG}_4$

```
P ← ⌊√p_max⌋
H ← ⌊√p_max⌋
for i ← 1 to m do
    if (i = m) then
        Buy n tickets
        return
    end if
    if (p_i ≤ h_i) then
        Buy n tickets
        return
    end if
    if (p_i + Σ_{j=1}^{i-1} h_j ≤ P) then
        Buy n tickets
        return
    end if
    if (Σ_{j=1}^{i} h_j ≥ H) then
        Buy n tickets
        return
    end if
end for
```

For this algorithm, we still expect the competitive ratio to be of order $\mathcal{O}(\sqrt{p_{\max}})$. Ideally, we want to prove in a Lemma what the competitive ratios are as a function of $p_{\max}$.

▶ **Lemma 7** (Competitive ratio's for $\mathtt{ALG}_4$). *Let $p_{\max} \in \mathbb{N}$. Then write $p_{\max} = k^2 + \ell$ for $k, \ell \in \mathbb{N}$ with $0 \leq \ell < 2k+1$. Then, in setting (c), the algorithm $\mathtt{ALG}_{\sqrt{p_{\max}}}$ is*

- *…-competitive if …,*
- *…-competitive if ….*

Proof. We almost have this proof. ◀

This result would mean that the simple threshold algorithm 2 can be modified slightly to be applicable to the much less restrictive setting with constraints (c) while still yielding essentially the same competitive ratio as before.

5 Randomized online algorithms

In the previous chapter, we proposed certain deterministic algorithms to try and solve the given problem. In this chapter, however, we will explore some randomized variants of these algorithms, with the goal of reaching a better *average competitive ratio* (see Definitions 3 and 2) of the algorithm, in comparison to its deterministic counterparts.

As an example, recall algorithm 2. One way to introduce randomness in the deterministic algorithms is to randomly select how many people we want to send home once the threshold-value has been reached. In this case $\lambda_i \in \mathbb{N}$ will be randomly sampled from $\{0, 1, \ldots, n\}$ and the algorithm looks like this:

Algorithm 5 The randomized variant of the $\lfloor\sqrt{p_{\max}}\rfloor$-threshold online algorithm $\texttt{ALG}_2$

```
Q ← ⌊√p_max⌋
r ← n                                        ▷ Variable for remaining people
for i ← 1 to m do
    λ_i ← randint(0, λ)                      ▷ Get a random integer number in a range
    if (i = m) then
        Buy r tickets.
        break
    end if
    if (p_i ≤ Q) then
        k ← n − λ_i                          ▷ Get the number of tickets to buy
        t ← min{k, r}                        ▷ In case of k > r
        Buy t tickets.
        r ← r − t                            ▷ Update remaining people
    end if
end for
```

6 Average performance

In this section, we will compare the average performances of the different algorithms seen in the preceding sections. The setting will be constraint (d) so, $h = 0$. Specifically, we compare the *average competitive ratios* of the algorithms, where we refer to Section 2 for the necessary definitions.

We expect the random algorithm $\texttt{ALG}_5$ to have a better average performance in comparison to the deterministic algorithm $\texttt{ALG}_2$, since the deterministic algorithm only tries to have a minimum competitive ratio and does not care about its average performance.

We will compare the average performances for several parameter settings with parameters $n, m, p_{\max} \in \mathbb{N}$, uniformly taken from the following distributions:

- $n \in \mathcal{U}(1, 100)$
- $m \in \mathcal{U}(2, 20)$
- $p_{\max} \in \mathcal{U}(20, 30)$

These values are taken arbitrarily, and can be altered in case new evidence suggests a better distribution.

We plan on displaying the results in the following way:
For each algorithm, we display a violin plot to show the distribution of the ratios for each algorithm. For the deterministic algorithms, we will underlay a barplot representing the theoretical competitive ratios. This is done to verify that the whole distribution of the experimental ratios lies within the theoretical competitive ratios similarly to what was done in [1]. Finally, we will overlay the mean ratios on the violin plot to enable comparisons.

7 Conclusions

A No constant competitive online algorithms for settings (a) and (b)

In this Appendix we prove that under certain conditions there does not exist a competitive deterministic online algorithm. The results can be summarized in the following two propositions.

▶ **Proposition 8.** *There does not exist an algorithm* $\mathtt{ALG}$ *and a real number* K *such that*

$$\mathtt{ALG}(I_a) \leq K \cdot \mathtt{OPT}(I_a)$$

for all $I_a \in \mathcal{I}_a$.

Proof. We have this ◀

▶ **Proposition 9.** *There does not exist an algorithm* $\mathtt{ALG}$ *and a real number* K *such that*

$$\mathtt{ALG}(I_b) \leq K \cdot \mathtt{OPT}(I_b)$$

for all $I_b \in \mathcal{I}_b$.

Proof. We have this ◀

References

1 Binghan Wu, Wei Bao, and Dong Yuan. Competitive analysis for two-level ski-rental problem. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(13):12034–12041, May 2021. URL: https://ojs.aaai.org/index.php/AAAI/article/view/17429, doi:10.1609/aaai.v35i13.17429.