# 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

We consider the plane-hotel problem, where the hotel and plane seat prices arrive online and the algorithm must irrevocably decide the number of tickets to buy on each day. The objective is to minimize the sum of hotel and plane ticket costs. First we show prove that the optimal algorithm in the offline variant of the problem is given by a greedy algorithm. Then, we prove that simple deterministic threshold algorithms attain competitive ratios of order $\mathcal{O}(\sqrt{p_{\max}})$ under different sets of constraints, namely with or without hotel prices. Furthermore, we prove optimality for two algorithms in the settings with only one person or a maximum of two days respectively. Finally, we introduce some random algorithms and carry out an empirical study on the competitive ratios of these randomized algorithms compared to the deterministic algorithms of preceding sections by uniformly sampling instances. We conclude that for average case performance our randomized algorithm slightly outperforms the deterministic algorithms, but for worst-case competitive ratios our deterministic algorithms remain superior.

GitHub: `https://github.com/TimGrimbergen/ADS_assignment.git`

**2012 ACM Subject Classification** Theory of computation → Online algorithms

**Keywords and phrases** Online algorithms, competitiveness, competitive analysis, randomized online algorithms, plane-hotel problem

## 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 on each day decide irrevocably how many tickets it will buy $f_i$. In Section 2, we will elaborate further on the different constraints put on the parameters above that are analyzed in this work.

It can be shown (see Appendices A and B) 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 also be drawn from a bounded interval.

For the competitive analyses of online algorithms it is very useful to know the optimal offline solution for any given instance. For this reason, we first propose an optimal greedy algorithm $\texttt{ALG}_1$ in Section 3 and provide a proof of correctness.

We will then analyze several deterministic algorithms in Section 4. A formal analysis is done on three algorithms, $\texttt{ALG}_2$, $\texttt{ALG}_3$ and $\texttt{ALG}_4$. For the simple threshold algorithm $\texttt{ALG}_2$ we show that it is $\mathcal{O}(\sqrt{p_{\max}})$-competitive under the constraints that $h_i = 0$. Also, we prove $\texttt{ALG}_2$ is an optimal algorithm if we further impose the restriction that $n = 1$. For the more sophisticated double-threshold algorithm $\texttt{ALG}_4$ we show that it is also $\mathcal{O}(\sqrt{p_{\max}})$-competitive if the constraint $h_i = 0$ is relaxed. Lastly, we show that the "greedy online" algorithm $\texttt{ALG}_3$ is optimal if $m = 2$ and $h_i = 0$.

Furthermore, we introduce two randomized algorithms in Section 4. Specifically, the fully "naive" randomized algorithm $\texttt{ALG}_5$ and a stochastic generalization of the threshold algorithm $\texttt{ALG}_2$.

Finally, in Section 6 we perform an empirical analysis on the algorithms $\texttt{ALG}_2$, $\texttt{ALG}_3$, $\texttt{ALG}_5$ and $\texttt{ALG}_6$ to find their average competitive ratios. Upon comparing the results with this new average metric, we conclude that our randomized algorithm $\texttt{ALG}_6$ indeed slightly outperforms the deterministic algorithms $\texttt{ALG}_2$ and $\texttt{ALG}_3$. As a bonus we "verify" that the theoretical competitive ratio of $\texttt{ALG}_2$ is correct.

## 2 Preliminaries

In this work, we will set several constraints on the general problem. In Table 1 we further distinguish six different constraint settings (a)-(f) on the number of people $n$, number of days $m$, available seats $s_i$, ticket price $p_i$ and hotel price $h_i$. The constraint that $\sum_{i=1}^{m} s_i \geq n$ is always required. Generally speaking, the settings become increasingly restrictive. Settings (a) and (b) are only relevant for the appendix and setting (c) is only relevant for $\texttt{ALG}_4$. The main focus of this work is setting (d), but the "sub-settings" (e) and (f) are further investigated for $\texttt{ALG}_2$ and $\texttt{ALG}_3$ respectively.

We also point out that we only consider "feasible" algorithms in this work. That is to say: $\texttt{ALG}$ is an element of space of all algorithms of setting (x) if and only if for all feasible instances $I \in \mathcal{I}_x$ (see Definition 1) the algorithm $\texttt{ALG}$ provides an actual solution, i.e., it buys exactly $n$ tickets. Some information of an instance is always known to the online algorithm, while

**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 $h_i$. Note that we define $\mathbb{N} := \{1, 2, \dots\}$ the set of strictly positive integers and $\mathbb{N}_0 := \{0\} \cup \mathbb{N}$.

| | $n$ | $m$ | $s_i$ | $p_i$ | $h_i$ |
|---|---|---|---|---|---|
| (a) | $\in \mathbb{N}$ | $\in \mathbb{N}$ | $\in \mathbb{N}_0$ | $\in \mathbb{N}$ | $\in \mathbb{N}_0$ |
| (b) | $\in \mathbb{N}$ | $\in \mathbb{N}$ | $\geq n$ | $\in \mathbb{N}$ | $\in \mathbb{N}_0$ |
| (c) | $\in \mathbb{N}$ | $\in \mathbb{N}$ | $\geq n$ | $\in \{1, \dots, p_{\max}\}$, $p_{\max} \in \mathbb{N}$ | $\in \{0, \dots, p_{\max}\}$ |
| (d) | $\in \mathbb{N}$ | $\in \mathbb{N}$ | $\geq n$ | $\in \{1, \dots, p_{\max}\}$, $p_{\max} \in \mathbb{N}$ | $= 0$ |
| (e) | $= 1$ | $\in \mathbb{N}$ | $\geq n$ | $\in \{1, \dots, p_{\max}\}$, $p_{\max} \in \mathbb{N}$ | $= 0$ |
| (f) | $\in \mathbb{N}$ | $\leq 2$ | $\geq n$ | $\in \{1, \dots, p_{\max}\}$, $p_{\max} \in \mathbb{N}$ | $= 0$ |

some information is revealed day by day. The information that is always known consists of $n$, $m$ and the restrictions on the *bounds* of $\mathbf{s}$, $\mathbf{p}$ and $\mathbf{h}$. The information that is only revealed day by day consists of the *values* of $\mathbf{s}, \mathbf{p}$ and $\mathbf{h}$. We then define (sub)instances and the set of all (sub)instances.

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

*The set of all* sub-instances of $\mathcal{I}_x$ belonging to $(n, m, p_{\max})$ *are denoted by $\mathcal{I}_x(n, m, p_{\max})$ and consists of only those instances $I = (n', m', p'_{\max}, \mathbf{s}, \mathbf{p}, \mathbf{h}) \in \mathcal{I}_x$ such that $(n', m', p'_{\max}) = (n, m, p_{\max})$ which are defined as the* sub-instances belonging to $(n, m, p_{\max})$.

We will also use the following definitions for the competitive ratios.

▶ **Definition 2.** *Let ALG be an online algorithm and let OPT be an optimal offline algorithm. For an instance $I$ we define the* competitive ratio of the algorithm for an instance *as*

$$c_{ALG}(I) = \frac{ALG(I)}{OPT(I)},$$

*where $ALG(I)$ is the algorithms cost and $OPT(I)$ is the optimal offline cost upon executing the instance $I$. Now let $x \in \{a,b,c,d,e,f\}$ and let $n, m, p_{\max} \in \mathbb{N}$ be such that the constraints of setting (x) are satisfied. We then define the* competitive ratio of the algorithm for a set of sub-instances $\mathcal{I}(n, m, p_{\max})$ *as*

$$c_{ALG} = \sup_{I \in \mathcal{I}} \{c(I)\} \in [1, \infty].$$

In general, if a parameter $n$, $m$ or $p_{\max}$ is present in the expression of $c_{\mathtt{ALG}}$, then this implies that the ALG is $c_{\mathtt{ALG}}$-competitive for a set of sub-instances $\mathcal{I}_x(n, m, p_{\max})$ which should be clear from the context.

Regarding the optimality of an online algorithm, we employ the definition below.

▶ **Definition 3.** *Let ALG be an online algorithm. We say ALG is a* **strongly** optimal deterministic algorithm in setting (x) *if for all tuples $(n, m, p_{\max})$ satisfying the constraints of setting (x) and deterministic online algorithms ALG′ we have that*

$$\sup_{I \in \mathcal{I}_x(n,m,p_{\max})} \{c_{ALG}(I)\} \leq \sup_{I \in \mathcal{I}_x(n,m,p_{\max})} \{c_{ALG'}(I)\}.$$

This definition of strong optimality ensures that the online algorithm should behave optimally *with all of the knowledge it knows beforehand.*

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

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

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

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

We will report the value of $\epsilon$. Note that for a deterministic online algorithm, the cost $\mathtt{ALG}(I)$ is always the same, thus in this case the average competitive ratio of an instance is just $\hat{c}_{\mathtt{ALG}}(I) = \frac{\mathtt{ALG}(I)}{\mathtt{OPT}(I)}$. 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 5.** *Let $J = \{I_1, \ldots, I_M\}$ with $M \gg 1$, where the instances $I_i$ are uniformly drawn samples from the allowed parameter space $\mathcal{I}$. We then define the* average competitive ratio of an algorithm *ALG as*

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

The specific value of $M$ used is reported. Note that this definition is meaningful for both deterministic and randomized algorithms.

## 3 Optimal offline algorithm

**Algorithm 1** An optimal offline greedy algorithm $\texttt{ALG}_1$

---

**for** all days $i$ **do**
  -Assign an effective price (see Eq. 1) for day $i$ depending on ticket price of day $d$ and cumulative hotel prices from day 1 to $i-1$
  -Add a tuple with these effective costs and preliminary information to an array $A$
**end for**
Sort those days ascending on cheapest cost, equals maintain original order.
**while** there are still people who need to be sent back **do**
  -Get the first (key, value)-pair from $A$, and remove that first element
  -Retrieve the maximum amount of people that can be sent home that day
  -Buy that number of tickets and store the associated cost
  -Update the amount of people who don't have tickets yet
**end while**
**return** The total cost of tickets and hotel prices

---

We then prove the following theorem.

▶ **Theorem 6** (Offline greedy is optimal). *Consider the offline algorithm 1, $\texttt{ALG}_1$. The following statements are true.*

1. *The greedy algorithm always provides a feasible solution.*
2. *This solution is optimal.*

**Proof.** Define $G = \{g_1, \ldots, g_m\}$ where $g_i$ is the number of tickets bought by $\texttt{ALG}_1$ on day $i$ and define $OPT = \{o_1, \ldots, o_m\}$ where $o_1$ is the number of tickets bought by an optimal solution $OPT$ on day $i$. Define $Q = (q_1, \ldots, q_m)$, with

$$q_i := p_i + \sum_{j=1}^{i-1} h_j \tag{1}$$

the "effective price" of a ticket on day $i$. Define $q^*$ to be the largest effective price for which $\texttt{ALG}_1$ has bought at least one ticket. Then define

$$J^+ := \{i \,|\, q_i > q^*\}$$
$$J^- := \{i \,|\, q_i < q^*\}$$
$$J^= := \{i \,|\, q_i = q^*\}.$$

By design of $\texttt{ALG}_1$, we know that $g_j = s_j$ for all $j \in J^-$, and by definition of $q^*$ we have that $g_j = 0$ for all $j \in J^+$. If $\sum_{j \in J^-} o_j < \sum_{j \in J^-} g_j$, then for any two days $j_1 \in J^-$ and $j_2 \in J^= \cup J^+$ we have that the optimal cost $C_{OPT}$ would decrease if the optimal solution $OPT$ were to buy a ticket on day $j_1$ instead of on $j_2$. As this is an obvious contradiction to the optimal solution being optimal, we must have $\sum_{j \in J^-} o_j = \sum_{j \in J^-} g_j$.
If $\sum_{j \in J^=} o_j < \sum_{j \in J^=} g_j$, then for any two days $j_1 \in J^=$ and $j_2 \in J^+$ we have that the optimal cost $C_{OPT}$ would decrease if the optimal solution $OPT$ were to buy a ticket on day $j_1$ instead of on $j_2$. As this is an obvious contradiction to the optimal solution being optimal, we must have $\sum_{j \in J^=} o_j = \sum_{j \in J^=} g_j$.
These statements also hold in special cases. If $J^- = \emptyset \wedge J^+ = \emptyset$, by definition of $J^=$, all

tickets are bought in $J^{=}$ for the same price $q^*$, so the greedy algorithm's cost is the same as the optimal cost. Using this same reasoning, it's trivial that for $J^- = \emptyset \wedge J^+ \neq \emptyset$ the greedy algorithm's cost and an optimal algorithm. The statements also apply to $J^- \neq \emptyset \wedge J^+ = \emptyset$, as there are simply no tickets available more expensive than $q^*$. ◀

## 4 Deterministic online algorithms

We first propose the simple threshold algorithm $\texttt{ALG}_2$:

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

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

The following result holds for the competitive ratio of $\texttt{ALG}_2$.

▶ **Lemma 7** (Competitive ratios for $\texttt{ALG}_2$). *Let $p_{\max}, m, n \in \mathbb{N}$ and write $p_{\max} = k^2 + \ell$ for $k \in \mathbb{N}$ with $0 \leq \ell < 2k+1$. Then, in setting (d), the algorithm $\texttt{ALG}_2$ is*

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

**Proof.** Let $n, m, p_{\max} \in \mathbb{N}$ so that the constraints of setting (d) are satisfied. For $m = 1$ the competitive ratio of 1 trivially follows from the fact that both $\texttt{ALG}_2$ and $\texttt{OPT}$ buy $n$ tickets on the first day. So from now on we may assume $m > 1$. Also, we notice that the number of people $n$ of an instance does not affect the competitive ratio $c_{\texttt{ALG}_2}(I)$ because both $\texttt{ALG}_2$ and $\texttt{OPT}$ buy all of their tickets on a single day, so

$$c_{\texttt{ALG}_2}(I) = \frac{\texttt{ALG}_2(I)}{\texttt{OPT}(I)} = \frac{p_{\texttt{ALG}_2}(I)n}{p_{\texttt{OPT}}(I)n} = \frac{p_{\texttt{ALG}_2}(I)}{p_{\texttt{OPT}}(I)},$$

where $p_{\texttt{ALG}_2}(I)$ and $p_{\texttt{OPT}}(I)$ are the prices at which $\texttt{ALG}_2$ and $\texttt{OPT}$ respectively buy $n$ tickets for instance $I$. Since we are in setting (d), we may define an instance $I$ by just its prices, so $I = \mathbf{p} = (p_1, \ldots, p_m)$. We then consider the following two collectively exhaustive and mutually exclusive cases.

1. $\forall i \in \{1, \ldots, m\}, p_i > k$ : In this case we specifically have the bounds $p_{\texttt{OPT}}(I) \geq k+1$ and $p_{\texttt{ALG}_2}(I) \leq p_{\max}$. Thus, for all instances $I$ in this case the competitive ratio $c_{\texttt{ALG}_2}(I)$ is bounded by

$$c_{\texttt{ALG}_2}(I) = \frac{p_{\texttt{ALG}_2}(I)}{p_{\texttt{OPT}}(I)} \leq \frac{p_{\max}}{k+1}.$$

2. $\exists i \in \{1, \ldots, m\}, p_i \leq k$ : In this case we specifically have the bounds $p_{\texttt{OPT}}(I) \geq 1$ and $p_{\texttt{ALG}_2}(I) \leq k$. Thus, for all instances $I$ in this case the competitive ratio $c_{\texttt{ALG}_2}(I)$ is bounded by

$$c_{\texttt{ALG}_2}(I) = \frac{p_{\texttt{ALG}_2}(I)}{p_{\texttt{OPT}}(I)} \leq \frac{k}{1} = k.$$

From the above two cases, we conclude that the competitive ratio $c_{\texttt{ALG}_2}(I)$ for an arbitrary instance $I \in \mathcal{I}_d$ is bounded by

$$c_{\texttt{ALG}_2}(I) \leq \max\{k, \frac{p_{\max}}{k+1}\} = \max\{k, \frac{k^2+\ell}{k+1}\},$$

and therefore that $c_{\texttt{ALG}_2} \le \max\{k, \frac{k^2+\ell}{k+1}\}$.
Now remember that we write $p_{\max} = k^2+\ell$ with $0 \le \ell < 2k+1$ and notice that $k = \lfloor\sqrt{p_{\max}}\rfloor$.
If $0 \le \ell \le k$, then we have

$$k \ge \ell \iff k^2+k \ge k^2+\ell \iff k(k+1) \ge k^2+\ell$$
$$\iff k \ge \frac{k^2+\ell}{k+1} \iff \lfloor\sqrt{p_{\max}}\rfloor \ge \frac{p_{\max}}{\lfloor\sqrt{p_{\max}}\rfloor+1}. \tag{2}$$

Thus in this case the competitive ratio $c_{\texttt{ALG}_2}$ is determined by $k$. Indeed the analysis is tight if we consider the instance $I = (k, 1, 1, \ldots, 1)$.
If $\ell > k$, then we can follow the inequalities in Eq. 2 to find that the competitive ratio is determined by $\frac{k^2+\ell}{k+1}$. The analysis here is again tight if we consider the instance $I = (k+1, \ldots, k+1, k^2+\ell)$.
Substituting $p_{\max}$ and $\lfloor\sqrt{p_{\max}}\rfloor$ for $k$ and $\ell$ we conclude that we indeed find the competitive ratios as stated in the Lemma description. ◀

Using Lemma **??**, we can even prove that $\texttt{ALG}_2$ is an optimal algorithm in setting (e), where the number of people is further restricted to $n = 1$.

▶ **Theorem 8.** *The algorithm $\texttt{ALG}_2$ is a strongly optimal deterministic online algorithm in setting (e).*

**Proof.** Let $p_{\max}, m \in \mathbb{N}$ so that the constraints of setting (e) are satisfied.
If $m = 1$ then $\texttt{ALG}_2$ is definitely optimal, since buying the the ticket on day 1 is trivially the optimal decision. So from now on we assume $m > 1$. Furthermore, we write $p_{\max} = k^2+\ell$ with $k \in \mathbb{N}$ and $0 \le \ell < 2k+1$. We then distinguish two cases in which we show that there exists no algorithm $\texttt{ALG}$ with a smaller competitive ratio than $\texttt{ALG}_2$ for sub-instances $\mathcal{I}_d(n = 1, m, p_{\max})$.

1. $\ell \le k$ : In this case we know by Lemma **??** that the competitive ratio of $\texttt{ALG}_2$ is $c_{\texttt{ALG}_2} = \lfloor\sqrt{p_{\max}}\rfloor = k$. For the sake of contradiction, assume there exists an algorithm $\texttt{ALG}$ such that $c_{\texttt{ALG}} < k$. Then, consider the two instances, which are defined only by the prices $\mathbf{p}$ on each of the $m$ days

$$I_1 = (k, k, \ldots, k, 1), \quad I_2 = (k, k, \ldots, k, p_{\max}).$$

   If $\texttt{ALG}$ waits for the last day to buy the ticket upon seeing the price $k$ for the first $m-1$ days, then the competitive ratio for instance $I_2$ would be $c_{\texttt{ALG}}(I_2) = \frac{p_{\max}}{k} \ge k$. However, if $\texttt{ALG}$ does not wait for the last day to buy the ticket upon seeing the price $k$ for some number of days, then the competitive ratio, for instance $I_1$, would be $c_{\texttt{ALG}}(I_1) = \frac{k}{1} \ge k$. Thus, no matter the decision $\texttt{ALG}$ makes, we find that there exists an instance $I$ such that $c_{\texttt{ALG}}(I) \ge k$ that contradicts the assumption that $c_{\texttt{ALG}} < k$.
2. $\ell > k$ : In this case we know by Lemma **??** that the competitive ratio of $\texttt{ALG}_2$ is $c_{\texttt{ALG}_2} = \frac{p_{\max}}{\lfloor\sqrt{p_{\max}}\rfloor+1} = \frac{k^2+\ell}{k+1}$. For the sake of contradiction, assume there exists an algorithm $\texttt{ALG}$ such that $c_{\texttt{ALG}} < \frac{k^2+\ell}{k+1}$. Then, consider the two instances

$$I_1 = (k+1, k+1, \ldots, k+1, 1), \quad I_2 = (k+1, k+1, \ldots, k+1, p_{\max}).$$

   If $\texttt{ALG}$ waits for the last day to buy the ticket upon seeing the price $k+1$ for the first $m-1$ days, then the competitive ratio for instance $I_2$ would be $\frac{k^2+\ell}{k+1}$. However, if $\texttt{ALG}$

does not wait for the last day to buy the ticket upon seeing the price $k+1$ for some number of days, then the competitive ratio for instance $I_1$ would be

$$c_{\texttt{ALG}}(I_1) = \frac{k+1}{1} = \frac{k^2+2k+1}{k+1} > \frac{k^2+\ell}{k+1},$$

since $\ell < 2k+1$. Thus, no matter the decision ALG makes, we find that there exists an instance $I$ such that $c_{\texttt{ALG}}(I) \geq \frac{k^2+\ell}{k+1}$ contradicting the assumption that $c_{\texttt{ALG}} < \frac{k^2+\ell}{k+1}$.

◀

As we have just seen, in setting (e) $\texttt{ALG}_2$ is in fact an optimal algorithm. However, in setting (d) $\texttt{ALG}_2$ is definitely not optimal. 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 find for the worst case $c = \frac{3}{2}$. Therefore, the algorithm $\texttt{ALG}_2$ fails in achieving the smallest possible competitive ratio for the set of sub-instances $\mathcal{I}(n = 2, m = 2, p_{\max} = 4)$, while clearly there exists an online algorithm that does make the optimal choice. So, $\texttt{ALG}_2$ is definitely not optimal for setting (d).

The observations above inspire the following "greedy" online algorithm $\texttt{ALG}_3$.

**Algorithm 3** A greedy online algorithm $\texttt{ALG}_3$

```
CC ← 0                                          ▷ 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 { (CC + p_i x + p_max (n − x)) / (p_min n), (CC + p_i x + 1·(n − x)) / n } }
    Buy n_i tickets.
    CC ← CC + p_i n_i
    n ← n − n_i
end for
```

The algorithm $\texttt{ALG}_3$ 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*. An important observation is that in computing the value $n_i$ we assume that we have to buy all remaining tickets on either day $i$ or on day $i+1$. Also, we only consider an adversary that either chooses to make $p_{i+1}$ as high as possible or as low as possible, since these are (very likely) the cases in which the competitive ratio attains a high value. In fact, Lemma 9 implies that $p_{\max}$ and 1 are the only candidates for $p_{i+1}$ that can affect the worst-case competitive ratio under some assumptions.

▶ **Lemma 9.** *Let $n, x, p_{\max}, p_1 \in \mathbb{N}$ with $p_1 \in \{1, \dots, p_{\max}\}$ and $x \in \{0, \dots, n\}$. Then,*

$$\max_{1 \leq p_2 \leq p_{\max}} \left\{ \frac{p_1 x + p_2(n-x)}{\min\{p_1, p_2\} n} \right\} = \max \left\{ \frac{p_1 x + p_{\max}(n-x)}{p_1 n}, \frac{p_1 x + (n-x)}{n} \right\}.$$

**Proof.** Assume for the sake of contradiction that the maximum is assumed for $1 < p_2 < p_{\max}$. Then distinguish between the following cases. If $p_1 \le p_2$ then we have

$$\frac{p_1x + p_2(n-x)}{\min\{p_1,p_2\}n} = \frac{p_1x + p_2(n-x)}{p_1n} < \frac{p_1x + p_{\max}(n-x)}{p_1n},$$

so contradiction. If $p_1 > p_2$ then we have

$$p_1x < p_1p_2x \implies p_1x+p_2(n-x) < p_1p_2x+p_2(n-x) \implies \frac{p_1x + p_2(n-x)}{p_2n} < \frac{p_1x + n - x}{n},$$

so again we reach a contradiction. ◀

▶ **Theorem 10.** *The algorithm $\mathtt{ALG}_3$ is a strongly optimal deterministic online algorithm in setting (f).*

**Proof.** Let $m, n, p_{\max} \in \mathbb{N}$ such that they satisfy the constraints of setting (f). If $m = 1$ then $\mathtt{ALG}_3$ is trivially optimal because both $\mathtt{ALG}_3$ and $\mathtt{OPT}$ buy $n$ tickets on the first day. Therefore, assume $m = 2$.
Since $m = 2$, we can make the following observation.
*If an algorithm $\mathtt{ALG}$ decides to buy $x$ tickets on day 1, then it must buy $n-x$ tickets on day 2.*
This means that in this case, we may define an algorithm $\mathtt{ALG}$ as the function

$$\mathtt{ALG} : \{1, \dots, p_{\max}\} \ni p_1 \mapsto x \in \{0, \dots, n\}. \tag{3}$$

As can be inferred from the description of $\mathtt{ALG}_3$, the function to which $\mathtt{ALG}_3$ reduces if $m = 2$ is given by

$$\mathtt{ALG}_3 : p_1 \mapsto \operatorname*{argmin}_{0 \le x \le n} \left\{ \max \left\{ \frac{p_1x + p_{\max}(n-x)}{p_1n}, \frac{p_1x + (n-x)}{n} \right\} \right\}.$$

For the sake of contradiction, assume there exists an algorithm $\mathtt{ALG}$ such that $c_{\mathtt{ALG}} < c_{\mathtt{ALG}_3}$ for the sub-instances $\mathcal{I}(n, m, p_{\max})$ of setting (f). If the price on the first day of an instance $I$ is given by $p_1$, then the worst case competitive ratio of an algorithm $\mathtt{ALG}$ for $I$ is given by

$$c_{\mathtt{ALG}}(I) = \max_{1 \le p_2 \le p_{\max}} \left\{ \frac{p_1x + p_2(n-x)}{\min\{p_1,p_2\}n} \right\} = \max \left\{ \frac{p_1x + p_{\max}(n-x)}{p_1n}, \frac{p_1x + (n-x)}{n} \right\},$$

where the second equality holds because of Lemma 9. So, we have that the maximum is attained for $p_2 \in \{1, p_{\max}\}$. Now it is clear that $\mathtt{ALG}_3$ corresponds to the optimal function, i.e., buy the number of tickets such that the worst-case competitive ratio is minimized. More formally, for all instances $I \in \mathcal{I}_f(n, m, p_{\max})$ that start with $p_1$ such that $\mathtt{ALG}(p_1) = \mathtt{ALG}_3(p_1)$, the competitive ratios are also the same so $c_{\mathtt{ALG}_3}(I) = c_{\mathtt{ALG}}(I)$. If however $\mathtt{ALG}(p_1) \ne \mathtt{ALG}_3(p_1)$, then by the definition of argmin we have for all instances $I$ starting with $p_1$ that

$$\begin{aligned} c_{\mathtt{ALG}_3}(I) &= \max \left\{ \frac{p_1\mathtt{ALG}_3(p_1) + p_{\max}(n - \mathtt{ALG}_3(p_1))}{p_1n}, \frac{p_1\mathtt{ALG}_3(p_1) + (n - \mathtt{ALG}_3(p_1))}{n} \right\} \\ &\le \max \left\{ \frac{p_1\mathtt{ALG}(p_1) + p_{\max}(n - \mathtt{ALG}(p_1))}{p_1n}, \frac{p_1\mathtt{ALG}(p_1) + (n - \mathtt{ALG}(p_1))}{n} \right\} = c_{\mathtt{ALG}}(I). \end{aligned}$$

So, for all $p_1$ we have

$$\max_{p_2}\{c_{\mathtt{ALG}_3}(p_1,p_2)\} \le \max_{p_2}\{c_{\mathtt{ALG}}(p_1,p_2)\}.$$

Since $n, m$ and $p_{\max}$ were chosen arbitrarily at the beginning, we conclude that for the set of sub-instances $\mathcal{I}_f(n, m, p_{\max})$ we have that $c_{\mathtt{ALG}_3} \le c_{\mathtt{ALG}}$ for any online algorithm $\mathtt{ALG}$, i.e., $\mathtt{ALG}_3$ is strongly optimal. ◀

Finally, to show that the hotel prices in this problem are not very impactful on the worst-case competitive ratios, we propose the algorithm $\texttt{ALG}_4$ which is specifically designed for setting (c) (so $h_i \in \{1, \ldots, p_{\max}\}$). The algorithm can be viewed as a natural extension of $\texttt{ALG}_2$, but now operating with two thresholds.

**Algorithm 4** The double threshold algorithm $\texttt{ALG}_4$

$P \leftarrow \lfloor \sqrt{p_{\max}} \rfloor$
$H \leftarrow \lfloor \sqrt{p_{\max}} \rfloor$
**for** $i \leftarrow 1$ *to* $m$ **do**
  **if** $(i = m \vee p_i + \sum_{j=1}^{i-1} h_j \leq P \vee \sum_{j=1}^{i} h_j \geq H)$ **then**
    Buy $n$ tickets
    **return**
  **else**
    Wait for the next day
  **end if**
**end for**

To show that the competitive ratios of $\texttt{ALG}_4$ are still $\mathcal{O}(\sqrt{p_{\max}})$, we prove the Lemma below.

▶ **Lemma 11** (Competitive ratios for $\texttt{ALG}_4$). *Let $n, m, p_{\max} \in \mathbb{N}$ such that the constraints of setting (c) are satisfied. 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 $\texttt{ALG}_4$ is*

- *1-competitive if $m = 1$*
- *$k$-competitive if $l = 0$ (and $m > 1$),*
- *$\left(k + \frac{l-1}{k+1}\right)$-competitive if $l > 0$ (and $m > 1$).*

**Proof.** Let $n, m, p_{\max} \in \mathbb{N}$ such that the constraints of setting (c) are satisfied. For generality and conciseness let the thresholds $P, H \in \mathbb{N}$ of $\texttt{ALG}_4$ be undefined.
If $m = 1$, then $\texttt{ALG}_4$ is trivially optimal again, so assume $m > 1$.
We then distinguish three collectively exhaustive classes for any sub-instance $I \in \mathcal{I}_c(n, m, p_{\max})$ corresponding to possible termination events for $\texttt{ALG}_4$.

1. $\forall k \in \{1, \ldots, m-1\} : p_k + \sum_{j=1}^{k-1} h_j > P \wedge \sum_{j=1}^{k} h_j < H$.
2. $\exists k \in \{1, \ldots, m-1\} : p_k + \sum_{j=1}^{k-1} h_j \leq P$ while $\forall \ell \in \{1, \ldots, k-1\} : \left(p_\ell + \sum_{j=1}^{\ell-1} h_j > P\right) \wedge \left(\sum_{j=1}^{\ell} h_j < H\right)$.
3. $\exists k \in \{1, \ldots, m-1\} : \sum_{j=1}^{\ell} h_j \geq H$ while $\forall \ell \in \{1, \ldots, k-1\} : \left(p_\ell + \sum_{j=1}^{\ell-1} h_j > P\right) \wedge \left(\sum_{j=1}^{\ell} h_j < H\right)$.

It is not difficult to see that the classes are collectively exhaustive, because either an instance is an element of class 1 or it is not, in which case it *has* to be an element of class 2 or 3. We then bound the competitive ratio for instances that are elements of certain classes, yielding four competitive ratios $c_1, \ldots, c_3$. The final competitive ratio $c$ will thus be the maximum over these three competitive ratios. At the end, we just have to provide an instance to show that the analysis is tight. Without loss of generality, we will omit the parameters $n$ because both $\texttt{ALG}_4$ and $\texttt{OPT}$ buy all tickets on the same day. Now, consider an instance $I \in \mathcal{I}_c(n, m, p_{\max})$.

1. **$I$ is an element of class 1:** In this case we buy all tickets on the last day, since none of the other statements were triggered for any day $k < m$. Furthermore, we specifically have

$p_k + \sum_{j=1}^{k-1} h_j > P$ and $\sum_{j=1}^{k} h_j < H$ for all $k \in \{1, \ldots, m-1\}$. We split the instance up into two more cases

**a.** $p_m + \sum_{j=1}^{m-1} h_j \leq P$. In this case, the optimal cost $\texttt{OPT}(I)$ is clearly achieved by buying on the last day. This results in a competitive ratio of 1.

**b.** $p_m + \sum_{j=1}^{m-1} h_j > P$. In this case, the optimal cost is bounded from below by $\texttt{OPT}(I) \geq P+1$ while the cost of the algorithm is bounded from above by $\texttt{ALG}_4(I) \leq p_{\max} + H - 1$. This results in a competitive ratio of

$$c_1 \leq \frac{p_{\max} + H - 1}{P+1}.$$

So, we conclude that for this class of instances, we obtain the bound

$$c_1 \leq \frac{p_{\max} + H - 1}{P+1}.$$

2. **$I$ is an element of class 2:** In this case the algorithm terminates because on some day $k$ we have $p_k + \sum_{j=1}^{k-1} h_j \leq P$. Clearly, $\texttt{ALG}_4(I) \leq P$ and $\texttt{OPT}(I) \geq 1$, so this results in a competitive ratio

$$c_2 \leq \frac{P}{1} = P$$

3. **$I$ is an element of class 3:** In this case the algorithm terminates because on some day $k$ we have $\sum_{j=1}^{k} h_j \geq H$. So, on day $k-1$, the cumulative hotel cost must have been bounded by $H-1$ hence the cost of the algorithm is bounded by $\texttt{ALG}_4(I) \leq p_{\max} + H - 1$. We distinguish three more cases

**a.** The optimal cost is achieved by buying after day $k$. In this case, we have that $\texttt{OPT}(I) \geq H+1$, because if the tickets are bought after day $k$, then all hotel prices up until that point also have to be paid. This results in a competitive ratio bounded by

$$c_3 \leq \frac{p_{\max} + H - 1}{H+1}$$

**b.** The optimal cost is achieved by buying before day $k$. In this case, we have that $\texttt{OPT}(I) \geq P+1$, because the $P$-threshold was never hit before day $k$. This results in a competitive ratio bounded by

$$c_3 \leq \frac{p_{\max} + H - 1}{P+1}$$

**c.** The optimal cost is achieved by buying on day $k$. In this case, the algorithm does the same as the optimal algorithm, so this results in a competitive ratio of 1.

So, the competitive ratio is bounded by

$$c_3 \leq \max\left\{\frac{p_{\max} + H - 1}{H+1}, \frac{p_{\max} + H - 1}{P+1}\right\}.$$

In the case of $\texttt{ALG}_4$ where we have $P = H = \lfloor\sqrt{p_{\max}}\rfloor$, we find that the actual competitive ratio $c$ over all sub-instances is bounded by

$$c \leq \max\{c_1, c_2, c_3\} = \max\left\{\frac{p_{\max} + \lfloor\sqrt{p_{\max}}\rfloor - 1}{\lfloor\sqrt{p_{\max}}\rfloor + 1}, \lfloor\sqrt{p_{\max}}\rfloor\right\},$$

and writing $p_{\max} = k^2 + \ell$ with $0 \leq \ell < 2k+1$ we find

$$c \leq \max\left\{\frac{k^2 + \ell + k - 1}{k+1}, k\right\} = \max\left\{k + \frac{\ell - 1}{k+1}, k\right\}.$$

If $\ell = 0$, so $p_{\max}$ is a square, then we choose the instance $\mathbf{p} = (k, 1, 1, \dots, 1)$ and $\mathbf{h} = (0, \dots, 0)$ such that indeed the competitive ratio indeed assumes $k$ in this case. If $\ell > 0$, then we choose the instance $\mathbf{p} = (k+1, p_{\max}, p_{\max}, \dots, p_{\max})$ and $\mathbf{h} = (k-1, 0, 0, \dots, 0)$ such that the competitive ratio indeed assumes $\frac{p_{\max}+k-1}{k+1}$. This proves the statement of the Lemma. ◀

This result implies that the simple threshold algorithm 2 can be extended to be applicable to the much less restrictive setting with constraints (c) while retaining the same order of 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 the naive randomized algorithm $\texttt{ALG}_5$ and the more sophisticated randomized algorithm $\texttt{ALG}_6$, which can be viewed as a generalization of $\texttt{ALG}_2$. The naive random algorithm $\texttt{ALG}_5$ randomly decides at initialization on what number of tickets to buy on each of the $m$ days, such that the total number of tickets sum up to $n$, see the definition below.

**Algorithm 5** The naive randomized algorithm $\texttt{ALG}_5$

Initialize an array $R$ of length $m$ with random integers on every position such that $\sum_{i=1}^{m} R_i = n$.
**for** $i \leftarrow 1$ *to* $m$ **do**
  Buy $R[i]$
**end for**

Furthermore, the more sophisticated randomized algorithm $\texttt{ALG}_6$ selects the number of tickets to buy according to some probability. This probability is calculated on the basis of the ticket price (see Fig. 1). We set a price lower bound $a$ and upper bound $b$. The probability of buying a ticket is 1 when the price $p_i$ is less than $a$, it then decreases linearly until the $b$ bound. Then, the probability of buying is 0 when $p_i$ is above the upper bound $b$.
Finally, the number of tickets to buy is determined by sampling from a binomial distribution

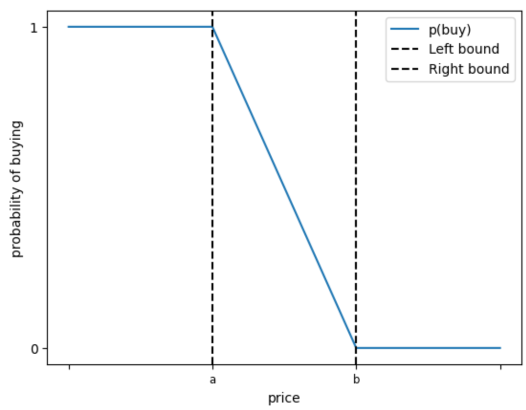



**Figure 1** Probability (y-axis) to buy a ticket for a person on day $i$ as a function of the price $p_i$ (x-axis). In this case $a = 1$, $b = 2$ and $p_{\max} > 2$.

with $n_i$ trials and the determined probability. The values for the parameters $\alpha$ and $\beta$ are taken to be respectively 0.9 and 0.1. These parameters were chosen arbitrarily in an attempt to minimize the average competitive ratio of the algorithm on a set of generated instances. In a real setting, these parameters would need to be optimized according to a set of past instances. The algorithm $\texttt{ALG}_6$ is described in more detail below.

**Algorithm 6** The randomized $\frac{p_i}{p_{\max}}$-ratio online algorithm

Initialize the left bound $a = 0.9 * \sqrt{p_{\max}}$ and right bound $b = \sqrt{p_{\max}} + 0.1 * (p_{\max} - \sqrt{p_{\max}})$.
**for** $i \leftarrow 1$ *to* $m$ **do**
  **if** $(i = m)$ **then**
    Buy all remaining tickets.
  **end if**
  **if** $p_i < a$ **then**
    Probability of buying a ticket is 1
  **else if** $a <= p_i < b$ **then**
    Probability of buying a ticket is determined by linearly interpolating between $a$ and $b$ (see Fig. 1).
  **else**
    Probability of buying a ticket is 0
  **end if**
  Draw from a binomial distribution with $n$ trials and the probability from above to determine the number of tickets to buy.
**end for**

## 6 Average performance

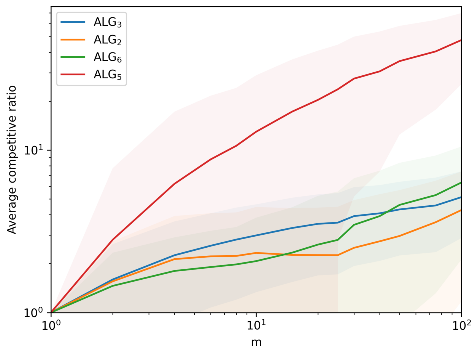
In this section, we will compare the average performances of the algorithms deterministic algorithms $\texttt{ALG}_2$ and $\texttt{ALG}_3$ with the randomized algorithms $\texttt{ALG}_5$ and $\texttt{ALG}_6$ seen in the preceding sections. The setting will be (d), so $h = 0$. First, we compare all algorithms in the violin plot the *average competitive ratios* of the algorithms, where we refer to Section 2 for the necessary definitions.

First we analyze the case instance $(n, m, p_{\max}) = (10, 10, 128)$ to generate the distributions of competitive ratios, see Fig. 2d. Note that the worst-case competitive ratio of $\texttt{ALG}_2$ is $c_{\texttt{ALG}_2} = 10$, which is indeed the theoretical result derived in Lemma 7. The worst-case performance of $\texttt{ALG}_3$ appears to be even better than $\texttt{ALG}_2$. Also, we see that indeed the randomized algorithm $\texttt{ALG}_6$ slightly outperforms the other algorithms on average competitive ratio. We also note that the performance of the naive randomized algorithm $\texttt{ALG}_5$ is very poor in all cases.

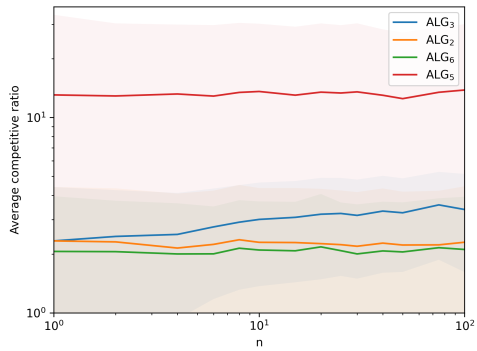
In Fig. 2a, we see that all average competitive ratios increase as a function of $m$. For smaller $m < 10$ the randomized algorithm $\texttt{ALG}_5$ outperforms the deterministic algorithms on average, however for larger $m > 10$ the simple threshold algorithm $\texttt{ALG}_2$ has minimal average competitive ratio.

In Fig. 2a, we see that the average competitive ratios are constant as a function of $n$. We observe that the algorithm $\texttt{ALG}_6$ slightly outperforms the deterministic algorithms on average competitive ratio.

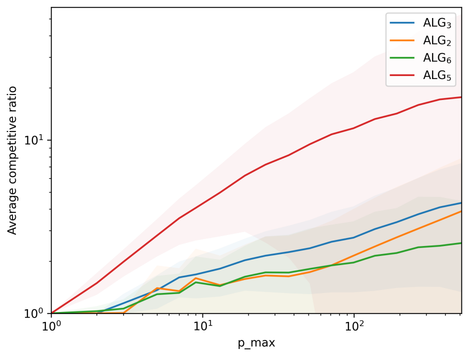
In Fig. 2a, we see that the average competitive ratios increase as a function of $p_{\max}$. For smaller $p_{\max} < 100$, the average competitive ratios of $\texttt{ALG}_2$ and $\texttt{ALG}_6$ are comparable, but for larger $p_{\max} > 100$, we observe that $\texttt{ALG}_6$ outperforms the deterministic algorithms.



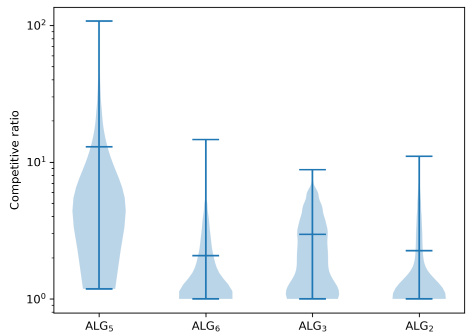

**(a)** Average competitive ratios with varying number of days $m \in [1, 100]$ and fixed $(n, p_{\max}) = (10, 128)$.




**(b)** Average competitive ratios with varying number of people $n \in [1, 100]$ and fixed $(n, p_{\max}) = (10, 128)$.




**(c)** Average competitive ratios with varying maximal price $p_{\max} \in [1, 512]$ and fixed $(n, m) = (10, 10)$.




**(d)** Observed competitive ratios with fixed $(n, m, p_{\max}) = (10, 10, 128)$.

**Figure 2** For figures 2a, 2b and 2c we sampled $M = 1000$ instances for every instance $(n, m, p_{\max})$. To determine the convergence of the randomized algorithms we use $\epsilon = 10^{-5}$. The solid lines describes the average competitive ratio of the 1000 instances, and the light shaded area's indicate the standard deviations corresponding to the algorithms. For figure 2d we sampled $M = 10000$ instances.

## 7 Discussion

In conclusion, we have shown that the simple deterministic threshold algorithms $\texttt{ALG}_2$ is optimal in setting (e) and attains a $\mathcal{O}(\sqrt{p_{\max}})$ competitive ratio in setting (d). The algorithm $\texttt{ALG}_3$ is shown to be optimal in setting (f), and $\texttt{ALG}_4$, is also shown to be $\mathcal{O}(\sqrt{p_{\max}})$-competitive for the most general setting (c).

On the deterministic side, since $\texttt{ALG}_3$ seems to performs very good for worst-case competitive ratio (see Fig 2d), future work could focus on proving more results on $\texttt{ALG}_3$ in the more interesting and general setting (d). Furthermore, $\texttt{ALG}_3$ could even be modified to incorporate hotel prices to also accommodate for setting (c).

On the randomized side, we have shown that our randomized algorithm $\texttt{ALG}_6$ can outperform the deterministic algorithms on average competitive ratio. A more sophisticated analysis would further optimize the parameters $a$ and $b$, and perhaps even change the probability curve between $a$ and $b$ to make it non-linear.

# A Single online algorithms for the unrestricted plane-hotel problem

The least restricted variant of the plane-hotel problem has no more than one feasible algorithm. Recall from Section 2 the set $\mathcal{I}_a$, containing all valid instances of the plane-hotel problem. For the purpose of this proof, we now define the set $\mathcal{A}_a$ containing all feasible algorithms creating valid solutions to all instances of problem (x), that is:

$$\mathcal{A}_a = \{\text{ALG} \mid \forall I \in \mathcal{I}_a : \text{ALG produces a valid solution for } I\} \tag{4}$$

A valid solution to an instance $I$ is any solution that sends all people home: $\sum_{i=1}^{m} f_i = n$. We can now prove that the following holds:

▶ **Lemma 12.** $|\mathcal{A}_a| = 1$ *with the only feasible algorithm always taking the first $n$ available seats.*

**Proof.** Assume that there exists a feasible algorithm `ALG` that does not by the first $n$ tickets for some instance $I \in \mathcal{I}_a$. We then have that there exists some day $i^* < m$ on which we have that $\sum_{i=1}^{i^*} f_i < n$. So, since $\sum_{i=1}^{m} f_i = n$, `ALG` must buy the remaining tickets on some days $> i^*$. Now consider the valid instance $I' \in \mathcal{A}$ which is identical to $I$ up until day $i^*$, but then has $s_i = 0$ for all $i > i^*$. Now, `ALG` does not produce a feasible solution on $I^*$, hence `ALG` is not a feasible algorithm. This contradiction implies that the only element of $\mathcal{A}_a$ is the algorithm which always buys the first $n$ tickets for every instance $I \in \mathcal{I}_a$. Note this algorithm *does* always produce a valid solution for every feasible instance since $\sum_{i=1}^{m} s_i \geq n$. So we conclude that indeed $|\mathcal{A}_a| = 1$.

◀

For the only algorithm in $\mathcal{A}_a$, we then prove the following theorem.

▶ **Theorem 13.** *Let $K \in \mathbb{N}$ and $n, m$ with $m \leq 2$. Then, the 'take-every-seat' algorithm is at least $K$-competitive.*

**Proof.** Choose the instance $I \in \mathcal{I}_a$ with $\mathbf{s} = (n, n, \ldots, n)$, $\mathbf{p} = (K+1, 1, \ldots, 1)$ and $\mathbf{h} = 0$. Then the competitive ratio of the algorithm is $c(I) = \frac{(K+1)n}{n} = K + 1 > K$. Therefore, the algorithm is at least $K$-competitive. ◀

From Theorem 13, we conclude that no constant-competitive algorithm exists in setting (a), because the only feasible algorithm is not constant-competitive. From these results, it is evident that some other guarantee on the number of seats is required for a feasible constant-competitive algorithm to exist.

# B No constant-competitive algorithm for the plane-hotel problem with a guarantee on seats

As outlined in Section 2, we opted, for simplicity, to constrain the number of seats in the subsequent settings (b)-(f) such that on each day $i$ there are $n$ seats available. Before doing analyses with stricter guarantees, we first do an analysis with only this very loose extra restriction of setting (b). Define $\mathcal{A}_b$ as the set of all feasible algorithms in setting (b) analogous to $\mathcal{A}_a$ in Eq. 4.

▶ **Theorem 14.** *Let `ALG` $\in \mathcal{A}_b$, $m, n \in \mathbb{N}$ with $m > 2$ and $K \in \mathbb{N}$. Then, `ALG` is at least $K$-competitive.*

**Proof.** Consider the adversarial instance $I_{\text{adv}} = (1, 2, \mathbf{s}, \mathbf{p}, \mathbf{h})$, where:

$\mathbf{s} = (n, n, \dots, n)$

$\mathbf{h} = 0$

$p_1 = K$

$$\forall i > 1 : p_i = \begin{cases} 1 & \text{if } f_1 = n \\ \frac{nK^2 - Kf_1}{n - f_1} & \text{if } f_1 < n \end{cases}$$

On this instance all algorithms have 2 options, either fly every person home on day 1 or not. We now evaluate the competitive ratio in both of these cases to obtain a lower bound on the competitive ratio.

1. If $f_1 = n$ people are flown home on the first day, then consider the instance with with $p_2 = 1$. The algorithms cost is $Kn$ while the optimal cost is $n$, thus the competitive ratio is $\frac{Kn}{n} = K$.
2. If $f_1 < n$ people are flown home on the first day, then consider the instance with $p_2 = \frac{nK^2 - Kf_1}{n - f_1}$. This time the algorithm incurs the cost

$$f_1 K + (n - f_1)\frac{nK^2 - Kf_1}{n - f_1} = f_1 K + nK^2 - Kf_1 = nK^2,$$

while the optimal incurs a cost of $Kn$. The resulting competitive ratio is $\frac{K^2 n}{Kn} = K$, which is the same as in the first case.

In either case, the competitive ratio for `ALG` is $K$. So, `ALG` is at least $K$-competitive. ◀

From Theorem 14, we conclude that there does not exist a feasible algorithm $\texttt{ALG} \in \mathcal{A}_b$ with a bounded competitive ratio, since $K$ can be chosen arbitrarily large. This result implies that some restriction on the maximal price of a ticket on every day is required.

## 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`.