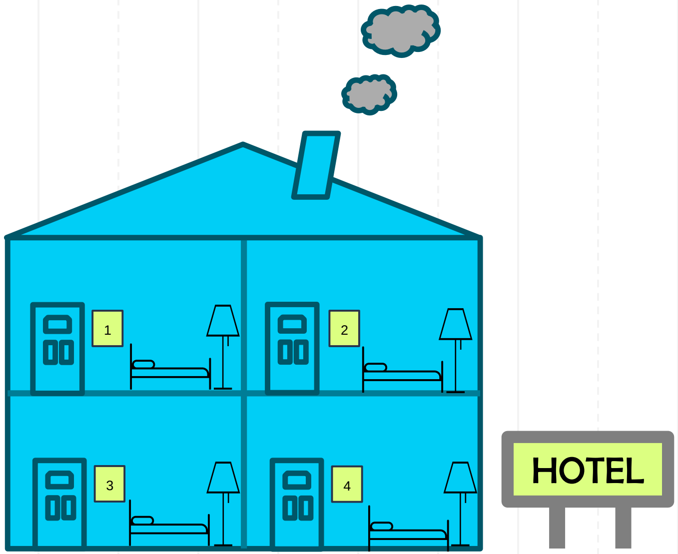
PARTIAL INFORMATION

Aleksandra Petrenko, Maya Marten

Motivation 1

Your hotel

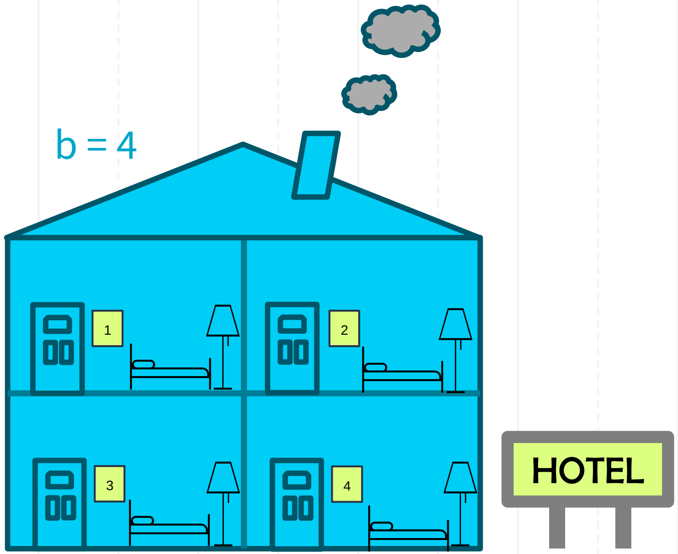


- Imagine you own this hotel
- 4 rooms to rent out
- You get requests from different types of customers

→ **Which customers do you accept?**

Partially predictable demand arrival model 2

Notations (1) - Resources



- **Single resource**
  - Hotel rooms
- **Identical units**
  - Same equipment

→ $\boldsymbol{b}$: units of the product to sell

Notations (2) - Periods

- $\boldsymbol{n}$: number of periods the product is sold

  - $n \geq b$

- $\boldsymbol{\lambda}$: time steps of the time horizon, normalized to 1:

$$\lambda = \frac{1}{n}, \frac{2}{n}, \ldots, 1$$

Notations (3) - Types of Customers

Type-1 Customer

- Pays **$1** for the room

Type-2 Customer

- Pays **$a** for the room (0<a<1)

No Customer

- Revenue of **$0** for the room

Sequence of arrival (1)

- The demand realizes sequentially
- At each time $\lambda = \frac{1}{n}, \frac{2}{n}, \ldots, 1$ we either face
    - a Type-1 Customer
    - a Type-2 Customer
    - no Customer
- At this moment we have to decide whether to accept or to reject the customer

Sequence of arrival (2)

- The sequence of arrival is denoted by $\vec{\boldsymbol{v}} = (v_1, v_2, \ldots, v_n)$, with $v_i \in \{0, a, 1\}$
    - The customers are represented by the revenue they generate if accepted

Goal

- Goal: Maximize Revenue

How to predict demand? (1)

Adversarial (unpredictable) demand

- Demand is **unpredictable**
- **Worst-case** approach
- Demand is controlled by an imaginary **adversary**

Stochastic (predictable) demand

- Demand is **predictable**
- Demand follows a known **distribution**

How to predict demand? (2) – Assessment

Adversarial (unpredictable) demand

+ Demand can't be perfectly predicted

− Worst-case approach often results in **too conservative** online policies

Stochastic (predictable) demand

+ If demand can be predicted, making online decisions incurs little loss

− Demand can't be perfectly predicted

→ Both approaches have their downsides!

Partially predictable demand model (1)

- Combine the 2 approaches:
  1. Adversary component
  2. Stochastic component

- Parameter $\boldsymbol{p}$, with $0 < p < 1$, specifies predictability of demand in the model
  - Extreme case p = 0 → pure Adversarial model
  - Extreme case p = 1 → pure Stochastic model

Partially predictable demand model (2)

- An imaginary adversary first sets an **initial sequence**

$$\overrightarrow{v_I} = (v_{I,1}, v_{I,2}, \dots v_{I,n})$$

with $v_{I,j} \in \{0, a, 1\}$, for $1 \leq j \leq n$

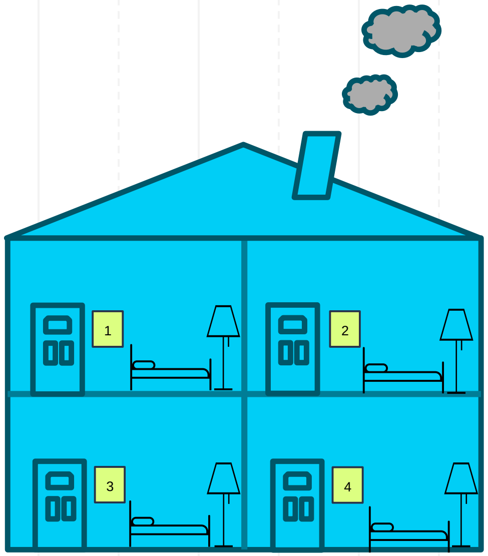
- A subset $\boldsymbol{S}$ of customers (p%) represents the stochastic group
  - won't follow the order determined by the adversary
- A subset $\boldsymbol{A}$ of customers ((1-p)%) represents the adversarial group
  - will follow the order determined by the adversary

Partially predictable demand model (3)

- Customers in S

  - join the subset independently
  - with the same propability p
  - Are permuted uniformly at random among themselves: $\sigma_S : S \to S$

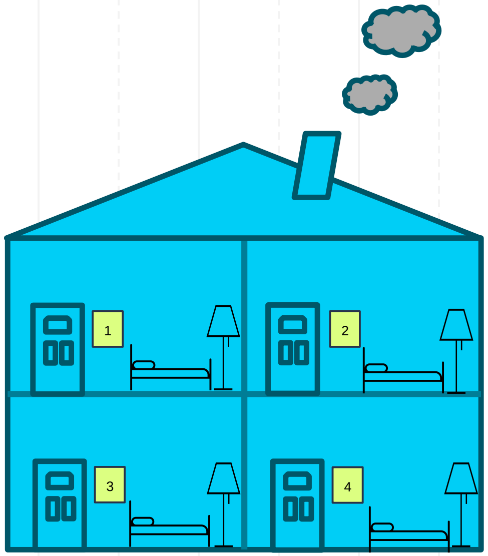
Example 1 a)



- 4 rooms to allocate: $\boldsymbol{b = 4}$
- We start allocating our hotel rooms 10 days in advance, we expect at most 1 customer per day: $\boldsymbol{n = 10}$
- We have historical data and assume, that 50% of our guests are predictable and permuted uniformly: $\boldsymbol{p = 0.5}$
- Type-2 Customers are willing to pay 0.6$ for the room: $\boldsymbol{a = 0.6}$

Type-1 Customer Type-2 Customer No Customer

Example 1 a)



Customers arrive in the following order:

λ	$\frac{1}{10}$	$\frac{2}{10}$	$\frac{3}{10}$	$\frac{4}{10}$	$\frac{5}{10}$	$\frac{6}{10}$	$\frac{7}{10}$	$\frac{8}{10}$	$\frac{9}{10}$	1
v_i	0.6	1	0	0.6	0	1	0.6	0.6	0.6	1

We assume:

- $S = \{2, 3, 6, 8, 10\}$
- $\sigma(2) = 10, \sigma(3) = 6, \sigma(6) = 8, \sigma(2) = 8, \sigma(10) = 3$

0.6	1	0	0.6	0	1	0.6	0.6	0.6	1
0.6	0.6	1	0.6	0	0	0.6	1	0.6	1

Further Proceeding

Competitive ratio → Algorithms → Computational Example

Competitive Ratio 3

How to assess the performance of online algorithms?

- **Competitive ratio**
  - **Worst-case ratio** between the revenue of the **online** scheme to that of a **clairvoyant** solution

- An online algorithm is ***c*-competitive** in the proposed partially predictable model if for any adversarial instance $\vec{v_I}$,

$$\mathbb{E}\left[ALG\left(\vec{V}\right)\right] \geq c \cdot OPT(\vec{v_I})$$

n_1:number of Type-1 Customers in the sequence
n_2:number of Type-2 Customers in the sequence
b: units of the product to sell

Optimal offline solution $\mathrm{OPT}(\vec{v})$: Strategy

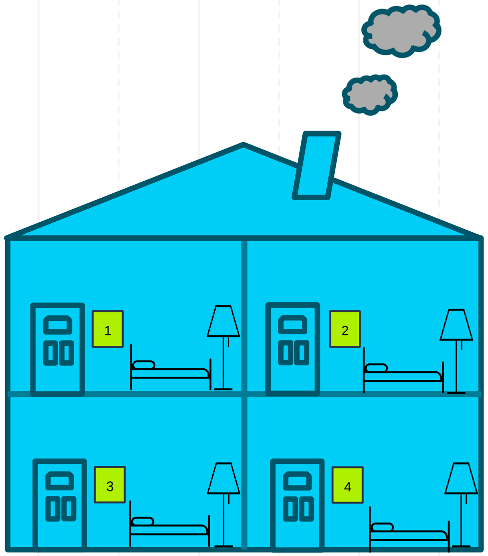
- Accept as many Type-1 Customers as possible
  - $\min\{n_1, b\}$
- If $n_1 < b$: Allocate the remaining units to as many Type-2 Customers as possible
  - $\min\{n_2, b - n_1\}$

$$\rightarrow \ \mathbf{OPT}(\vec{v}) = \mathbf{min}\{\mathbf{n_1}, \mathbf{b}\} + \mathbf{a} \ \cdot \boldsymbol{min}\{\boldsymbol{n_2}, (\boldsymbol{b} - \boldsymbol{n_1})^+\}$$

$(b - n_1)^+ \triangleq \max(b - n_1, 0)$

n_1:number of Type-1 Customers in the sequence
n_2:number of Type-2 Customers in the sequence
b: units of the product to sell

Example $\mathrm{OPT}(\vec{v})$



λ	$\frac{1}{10}$	$\frac{2}{10}$	$\frac{3}{10}$	$\frac{4}{10}$	$\frac{5}{10}$	$\frac{6}{10}$	$\frac{7}{10}$	$\frac{8}{10}$	$\frac{9}{10}$	1

v_i	0.6	0.6	1	0.6	0	0	0.6	1	0.6	1

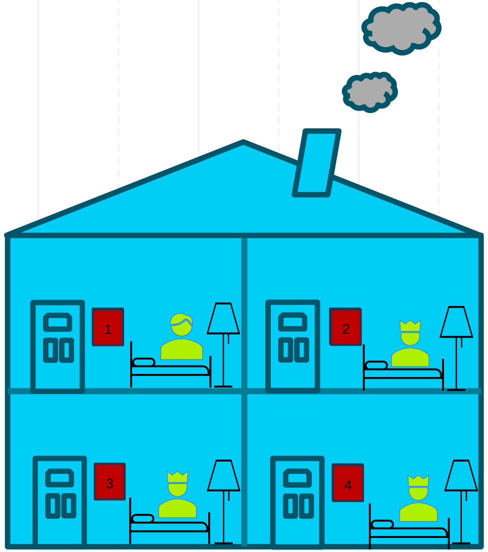
$$\mathbf{OPT}(\vec{v}) = \mathbf{min}\{\mathbf{n_1}, \mathbf{b}\} + \mathbf{a} \cdot \boldsymbol{min}\{\boldsymbol{n_2}, (\boldsymbol{b} - \boldsymbol{n_1})^+\}$$

$a = 0.6 \qquad b = 4 \qquad n_1 = 3 \qquad n_2 = 5$

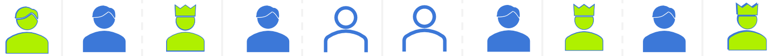
$$\begin{aligned}\mathrm{OPT}(\vec{v}) &= \min\{3,4\} + 0.6 \cdot min\{5,(4-3)^+\} \\ &= 3 + 0.6 \cdot min\{5,1\} \\ &= 3 + 0.6 \cdot 1 \\ &= 3 + 0.6 \\ &= 3.6\end{aligned}$$

n_1:number of Type-1 Customers in the sequence
n_2:number of Type-2 Customers in the sequence
b: units of the product to sell

Example $\mathbf{OPT}(\vec{v})$



λ	$\frac{1}{10}$	$\frac{2}{10}$	$\frac{3}{10}$	$\frac{4}{10}$	$\frac{5}{10}$	$\frac{6}{10}$	$\frac{7}{10}$	$\frac{8}{10}$	$\frac{9}{10}$	1
v_i	0.6	0.6	1	0.6	0	0	0.6	1	0.6	1



$$\mathbf{OPT}(\vec{v}) = \mathbf{min}\{\mathbf{n_1}, \mathbf{b}\} + \mathbf{a} \cdot \boldsymbol{min}\{\boldsymbol{n_2}, (\boldsymbol{b} - \boldsymbol{n_1})^+\}$$

$a = 0.6 \qquad b = 4 \qquad n_1 = 3 \qquad n_2 = 5$

$$\begin{aligned}
\mathrm{OPT}(\vec{v}) &= \min\{3, 4\} + 0.6 \cdot min\{5, (4-3)^+\} \\
&= 3 + 0.6 \cdot min\{5,1\} \\
&= 3 + 0.6 \cdot 1 \\
&= 3 + 0.6 \\
&= 3.6
\end{aligned}$$

4 Non-adaptive algorithm

Non-adaptive algorithm: Basics

For each time period

- Always accept a T1 customer
- **Evolving threshold rule:** Accept a T2 customer, if the number of accepted T1 and T2 customers by the evolving threshold does not exceed $\lfloor \lambda p b \rfloor$
- Otherwise, **fixed threshold rule**: accept a T2 customer, if the number of T2 customers accepted by the fixed threshold does not exceed $\lfloor \theta b \rfloor$, where $\theta = \frac{1-p}{2-a}$
- Otherwise, reject T2 customer
- Repeat until there is no more inventory or no more customers

Non-adaptive algorithm: Pseudocode

Initialize:

$$q_1 \leftarrow 0$$
$$q_{2,e} \leftarrow 0$$
$$q_{2,f} \leftarrow 0$$
$$rem.inv \leftarrow b$$
$$\theta \triangleq \frac{1-p}{2-a}$$

for $\lambda = \frac{1}{n}$ to 1 **do**

$\quad i = \lambda \cdot n$

$\quad$ **if** $rem.inv > 0$ **then**

$\quad\quad$ **if** $v_i = 1$ **then**

$\quad\quad\quad q_1 \leftarrow q_1 + 1$

$\quad\quad\quad rem.inv \leftarrow rem.inv - 1$

Non-adaptive algorithm: Pseudocode

```
        else if v_i = a and q_1 + q_{2,e} < ⌊λpb⌋ then
            q_{2,e} ← q_{2,e} + 1
            rem.inv ← rem.inv − 1
        else if v_i = a and q_{2,f} < ⌊θb⌋ then
            q_{2,f} ← q_{2,f} + 1
            rem.inv ← rem.inv − 1
        end if
    end if
end for
```

Competitive ratio

$$p + \frac{1-p}{2-a} + O\left(\frac{1}{a(1-p)p}\sqrt{\frac{\log n}{b}}\right)$$

Example ALG_1

Algorithm 1 Non-adaptive Algorithm ALG_1

```
Initialize:
    q_1 ← 0
    q_{2,e} ← 0
    q_{2,f} ← 0
    rem.inv ← b
    θ ≜ (1−p)/(2−a)
for λ = 1/n to 1 do
    i = λ · n
    if rem.inv > 0 then
        if v_i = 1 then
            q_1 ← q_1 + 1
            rem.inv ← rem.inv − 1
        else if v_i = a and q_1 + q_{2,e} < ⌊λpb⌋ then
            q_{2,e} ← q_{2,e} + 1
            rem.inv ← rem.inv − 1
        else if v_i = a and q_{2,f} < ⌊θb⌋ then
            q_{2,f} ← q_{2,f} + 1
            rem.inv ← rem.inv − 1
        end if
    end if
end for
```

Initialize:

$\lambda = \frac{1}{10}$:

$\boldsymbol{\lambda}$	$\frac{1}{10}$	$\frac{2}{10}$	$\frac{3}{10}$	$\frac{4}{10}$	$\frac{5}{10}$	$\frac{6}{10}$	$\frac{7}{10}$	$\frac{8}{10}$	$\frac{9}{10}$	1
$\boldsymbol{v_i}$	0.6	0.6	1	0.6	0	0	0.6	1	0.6	1

$\theta \triangleq \frac{1-p}{2-a} = \frac{1-0.5}{2-0.6} = \frac{5}{14}; q_1 = 0;\ q_{2,e} = 0; q_{2,f} = 0; rem.inv = 4$

$i = \frac{1}{10} \cdot 10 = 1$

$rem.inv > 0?$ □

$v_1 = 1$? □

$v_1 = 0.6$? ✓

$q_1 + q_{2,e} < \lfloor \lambda p b \rfloor$

$0 + 0 < \left\lfloor \frac{1}{10} \cdot 0.5 \cdot 4 \right\rfloor$

$0 < \lfloor 0.2 \rfloor$

$0 < 0$ ×

Evolving threshold rule

$v_1 = 0.6$? ✓

$q_{2,f} < \lfloor \theta b \rfloor$

$0 < \left\lfloor \frac{5}{14} \cdot 4 \right\rfloor$

$0 < \left\lfloor \frac{10}{7} \right\rfloor$

$0 < 1$ ✓

Fixed threshold rule

→Accept Customer 1: $q_{2,f} = 1, rem.inv = 3$

Example ALG_1

λ	$\frac{1}{10}$	$\frac{2}{10}$	$\frac{3}{10}$	$\frac{4}{10}$	$\frac{5}{10}$	$\frac{6}{10}$	$\frac{7}{10}$	$\frac{8}{10}$	$\frac{9}{10}$	1

v_i	0.6	0.6	1	0.6	0	0	0.6	1	0.6	1

Algorithm 1 Non-adaptive Algorithm ALG_1

```
Initialize:
    q_1 ← 0
    q_{2,e} ← 0
    q_{2,f} ← 0
    rem.inv ← b
    θ ≜ (1−p)/(2−a)
for λ = 1/n to 1 do
    i = λ · n
    if rem.inv > 0 then
        if v_i = 1 then
            q_1 ← q_1 + 1
            rem.inv ← rem.inv − 1
        else if v_i = a and q_1 + q_{2,e} < ⌊λpb⌋ then
            q_{2,e} ← q_{2,e} + 1
            rem.inv ← rem.inv − 1
        else if v_i = a and q_{2,f} < ⌊θb⌋ then
            q_{2,f} ← q_{2,f} + 1
            rem.inv ← rem.inv − 1
        end if
    end if
end for
```

Initialize:

$\theta \triangleq \frac{1-p}{2-a} = \frac{1-0.5}{2-0.6} = \frac{5}{14}; q_1 = 0; q_{2,e} = 0; q_{2,f} = 0; rem.inv = 4$

$\lambda = \frac{1}{10}$:

$i = \frac{1}{10} \cdot 10 = 1$

$rem.inv > 0?$ □

$v_1 = 1$? □

$v_1 = 0.6$? ✓

$q_1 + q_{2,e} < \lfloor \lambda p b \rfloor$

$0 + 0 < \left\lfloor \frac{1}{10} \cdot 0.5 \cdot 4 \right\rfloor$

$0 < \lfloor 0.2 \rfloor$

$0 < 0$ ×

Evolving threshold rule

$v_1 = 0.6$? ✓

$q_{2,f} < \lfloor \theta b \rfloor$

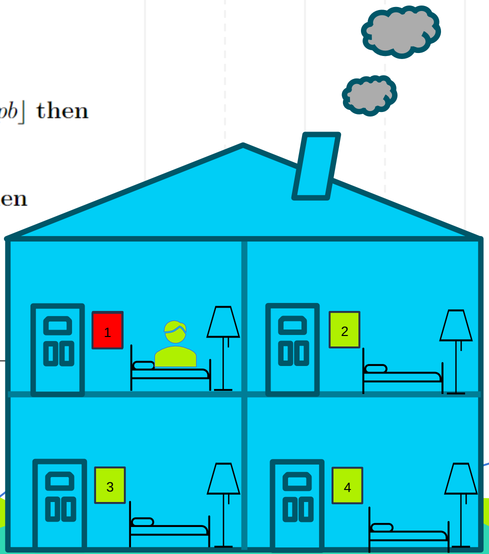
$0 < \left\lfloor \frac{5}{14} \cdot 4 \right\rfloor$

$0 < \left\lfloor \frac{10}{7} \right\rfloor$

$0 < 1$ ✓

Fixed threshold rule

→Accept Customer 1: $q_{2,f} = 1, rem.inv = 3$



Example ALG_1

λ	$\frac{1}{10}$	$\frac{2}{10}$	$\frac{3}{10}$	$\frac{4}{10}$	$\frac{5}{10}$	$\frac{6}{10}$	$\frac{7}{10}$	$\frac{8}{10}$	$\frac{9}{10}$	1
v_i	0.6	0.6	1	0.6	0	0	0.6	1	0.6	1

Algorithm 1 Non-adaptive Algorithm ALG_1

```
Initialize:
    q_1 ← 0
    q_{2,e} ← 0
    q_{2,f} ← 0
    rem.inv ← b
    θ ≜ (1−p)/(2−a)
for λ = 1/n to 1 do
    i = λ · n
    if rem.inv > 0 then
        if v_i = 1 then
            q_1 ← q_1 + 1
            rem.inv ← rem.inv − 1
        else if v_i = a and q_1 + q_{2,e} < ⌊λpb⌋ then
            q_{2,e} ← q_{2,e} + 1
            rem.inv ← rem.inv − 1
        else if v_i = a and q_{2,f} < ⌊θb⌋ then
            q_{2,f} ← q_{2,f} + 1
            rem.inv ← rem.inv − 1
        end if
    end if
end for
```

$\lambda = \frac{2}{10}$:

$i = \frac{2}{10} \cdot 10 = 2$

$rem.inv > 0?$ ✓

$v_1 = 1$? ×

$v_1 = 0.6$? ✓

$q_1 + q_{2,e} < \lfloor \lambda p b \rfloor$

$0 + 0 < \left\lfloor \frac{2}{10} \cdot 0.5 \cdot 4 \right\rfloor$

$0 < \lfloor 0.4 \rfloor$

$0 < 0$ ×

$v_1 = 0.6$? ✓

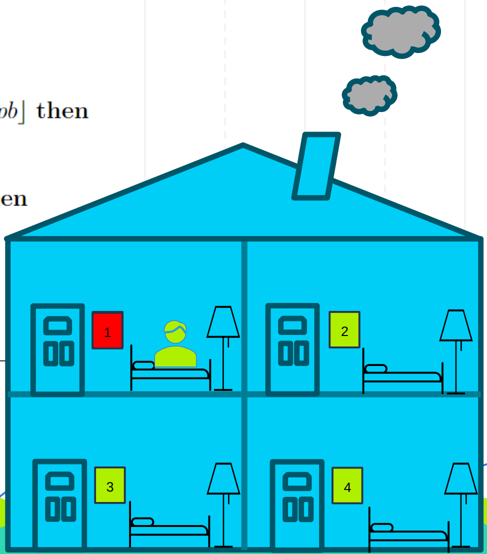
$q_{2,f} < \lfloor \theta b \rfloor$

$1 < \left\lfloor \frac{5}{14} \cdot 4 \right\rfloor$

$1 < \left\lfloor \frac{10}{7} \right\rfloor$

$1 < 1$ ×

→Reject Customer 2



Example ALG_1

Algorithm 1 Non-adaptive Algorithm ALG_1

Initialize:
- $q_1 \leftarrow 0$
- $q_{2,e} \leftarrow 0$
- $q_{2,f} \leftarrow 0$
- $rem.inv \leftarrow b$
- $\theta \triangleq \frac{1-p}{2-a}$

for $\lambda = \frac{1}{n}$ to 1 **do**
- $i = \lambda \cdot n$
- **if** $rem.inv > 0$ **then**
  - **if** $v_i = 1$ **then**
    - $q_1 \leftarrow q_1 + 1$
    - $rem.inv \leftarrow rem.inv - 1$
  - **else if** $v_i = a$ and $q_1 + q_{2,e} < \lfloor \lambda p b \rfloor$ **then**
    - $q_{2,e} \leftarrow q_{2,e} + 1$
    - $rem.inv \leftarrow rem.inv - 1$
  - **else if** $v_i = a$ and $q_{2,f} < \lfloor \theta b \rfloor$ **then**
    - $q_{2,f} \leftarrow q_{2,f} + 1$
    - $rem.inv \leftarrow rem.inv - 1$
  - **end if**
- **end if**

end for

λ	$\frac{1}{10}$	$\frac{2}{10}$	$\frac{3}{10}$	$\frac{4}{10}$	$\frac{5}{10}$	$\frac{6}{10}$	$\frac{7}{10}$	$\frac{8}{10}$	$\frac{9}{10}$	1
v_i	0.6	0.6	1	0.6	0	0	0.6	1	0.6	1

$\lambda = \frac{3}{10}$:

$i = \frac{3}{10} \cdot 10 = 3$

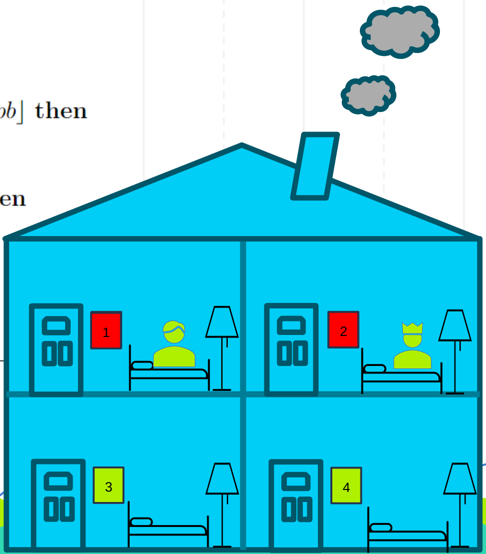
$rem.inv > 0?$ ✓

$v_1 = 1$ ✓

$q_1 = 1$

$rem.inv = 2$

→Accept Customer 3



Example ALG_1

Algorithm 1 Non-adaptive Algorithm ALG_1

Initialize:
- $q_1 \leftarrow 0$
- $q_{2,e} \leftarrow 0$
- $q_{2,f} \leftarrow 0$
- $rem.inv \leftarrow b$
- $\theta \triangleq \frac{1-p}{2-a}$

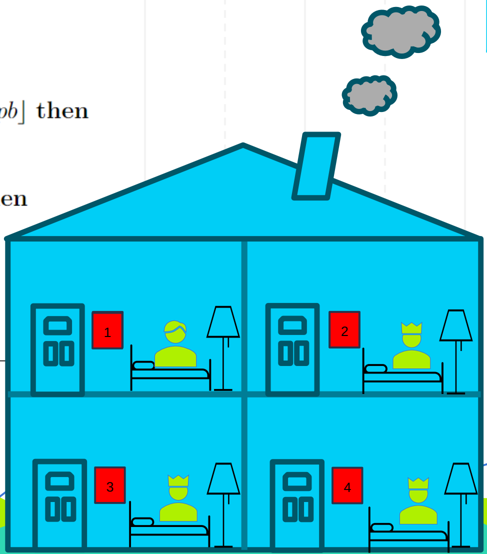
for $\lambda = \frac{1}{n}$ to 1 **do**
- $i = \lambda \cdot n$
- **if** $rem.inv > 0$ **then**
  - **if** $v_i = 1$ **then**
    - $q_1 \leftarrow q_1 + 1$
    - $rem.inv \leftarrow rem.inv - 1$
  - **else if** $v_i = a$ and $q_1 + q_{2,e} < \lfloor \lambda p b \rfloor$ **then**
    - $q_{2,e} \leftarrow q_{2,e} + 1$
    - $rem.inv \leftarrow rem.inv - 1$
  - **else if** $v_i = a$ and $q_{2,f} < \lfloor \theta b \rfloor$ **then**
    - $q_{2,f} \leftarrow q_{2,f} + 1$
    - $rem.inv \leftarrow rem.inv - 1$
  - **end if**
- **end if**

end for

Solution

λ	$\frac{1}{10}$	$\frac{2}{10}$	$\frac{3}{10}$	$\frac{4}{10}$	$\frac{5}{10}$	$\frac{6}{10}$	$\frac{7}{10}$	$\frac{8}{10}$	$\frac{9}{10}$	1

v_i	0.6	0.6	1	0.6	0	0	0.6	1	0.6	1



Adaptive algorithm 5

Adaptive algorithm: Basics

For each time period

- Introduce $c \in [0,1]$
- Always accept a T1 customer
- **First condition:** Accept a T2 customer, if $u_{1,2}(\lambda) < b$
- Otherwise, **second condition**: accept a T2 customer, if

$q_2 > \lfloor \Phi b + c(b - u_1(\lambda))^+ \rfloor$, where $\Phi = \frac{1-c}{1-a}$

- Otherwise, reject a T2 customer
- Repeat until there is no more inventory or no more customers

Adaptive algorithm: Pseudocode

Initialize:

$q_1 \leftarrow 0$

$q_2 \leftarrow 0$

$\Phi \triangleq \frac{1-c}{1-a}$

$\delta \triangleq \frac{\Phi b}{n}$

$rem.inv = b$

for $\lambda = \frac{1}{n}$ to 1 **do**

  if $\lambda < \delta$ **then**

    $u_1(\lambda) \triangleq b$

    $u_{1,2}(\lambda) \triangleq b$

  else if $\lambda \geq \delta$ **then**

    $u_1(\lambda) \triangleq \min\{\frac{o_1(\lambda)}{\lambda p}, \frac{o_1(\lambda)+(1-\lambda)(1-p)n}{1-p+\lambda p}\}$

    $u_{1,2}(\lambda) \triangleq \min\{\frac{o_1(\lambda)+o_2(\lambda)}{\lambda p}, \frac{o_1(\lambda)+o_2(\lambda)+(1-\lambda)(1-p)n}{1-p+\lambda p}\}$

  end if

Adaptive algorithm: Pseudocode

$i = \lambda \cdot n$
if $rem.inv > 0$ **then**
    if $v_i = 1$ **then**
        $q_1 \leftarrow q_1 + 1$
        $rem.inv \leftarrow rem.inv - 1$
    else if $v_i = a$ and $u_{1,2}(\lambda) < b$ **then**
        $q_2 \leftarrow q_2 + 1$
        $rem.inv \leftarrow rem.inv - 1$
    else if $v_i = a$ and $q_2 \leq \lfloor \Phi b + c(b - u_1(\lambda))^+ \rfloor$ **then**
        $q_2 \leftarrow q_2 + 1$
        $rem.inv \leftarrow rem.inv - 1$
    end if
end if
end for

Competitive ratio

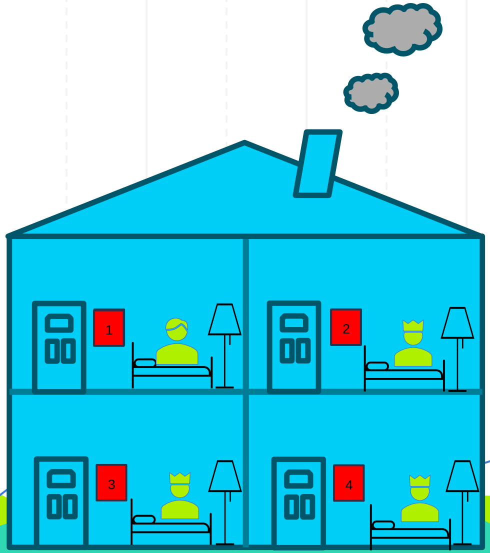
$$\mathrm{c} - O\left(\frac{1}{(1-c)^2 a p^{\frac{3}{2}}}\sqrt{\frac{n^2 \log n}{b^3}}\right)$$

$- \, competitive \; for \; \forall \, c \leq c^*, c \; < 1$

Example ALG_1

Solution

λ	$\frac{1}{10}$	$\frac{2}{10}$	$\frac{3}{10}$	$\frac{4}{10}$	$\frac{5}{10}$	$\frac{6}{10}$	$\frac{7}{10}$	$\frac{8}{10}$	$\frac{9}{10}$	1
v_i	0.6	0	1	0.6	1	0	0.6	0	0	0.6



6 Comparison of both algorithms

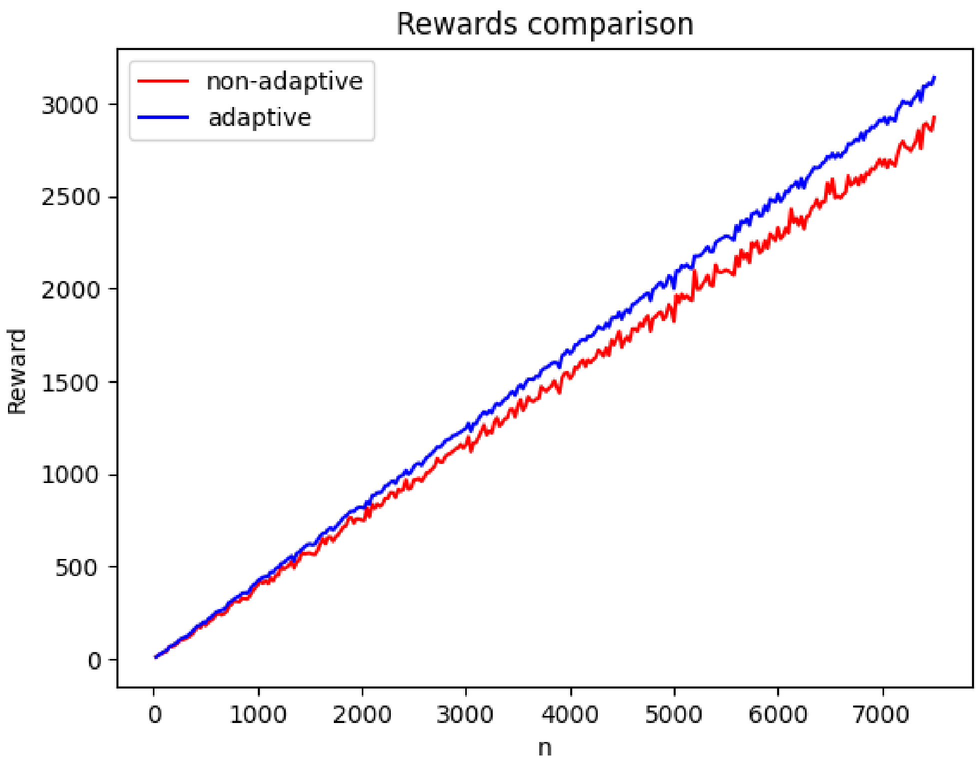
Methodology

With the help of Python and Google Colab we create:

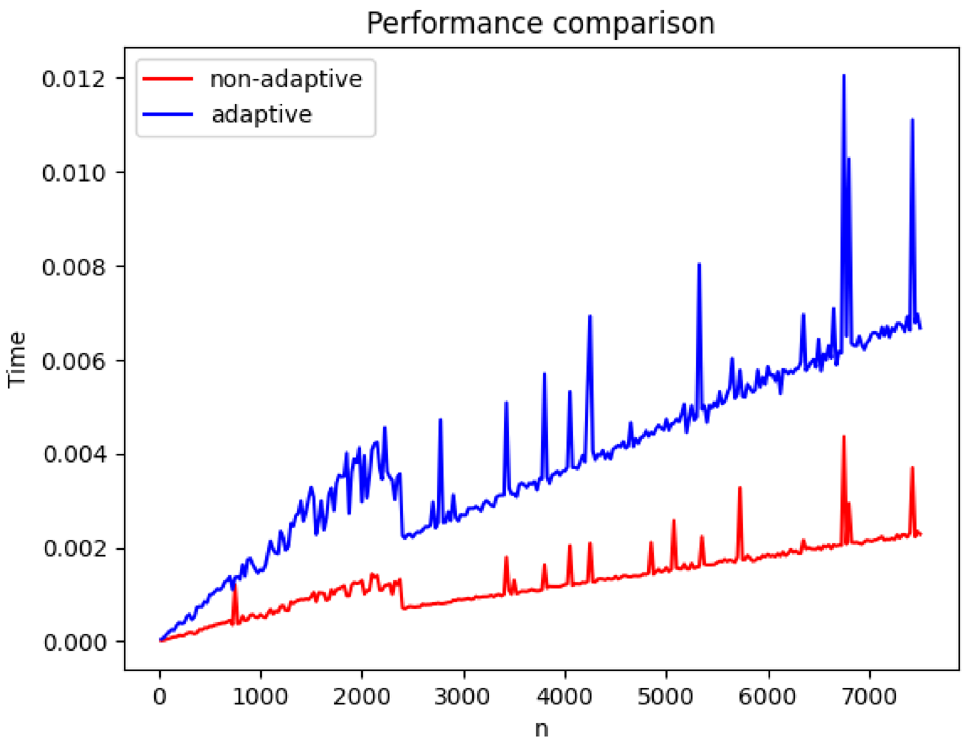
- Customer sequence generator according to demand arrival model
- Two functions for the non-adaptive algorithm (detailed and not detailed)
- Two functions for the adaptive algorithm (detailed and not detailed)
- Simulation:

- We begin with $n = 25$ and increase it by 25
- For each n we create a customer sequence and run both algorithms and repeat it $k = 300$ times
- We run the simulation for 25 different combinations of parameters $a, d = \frac{b}{n}, p, c$

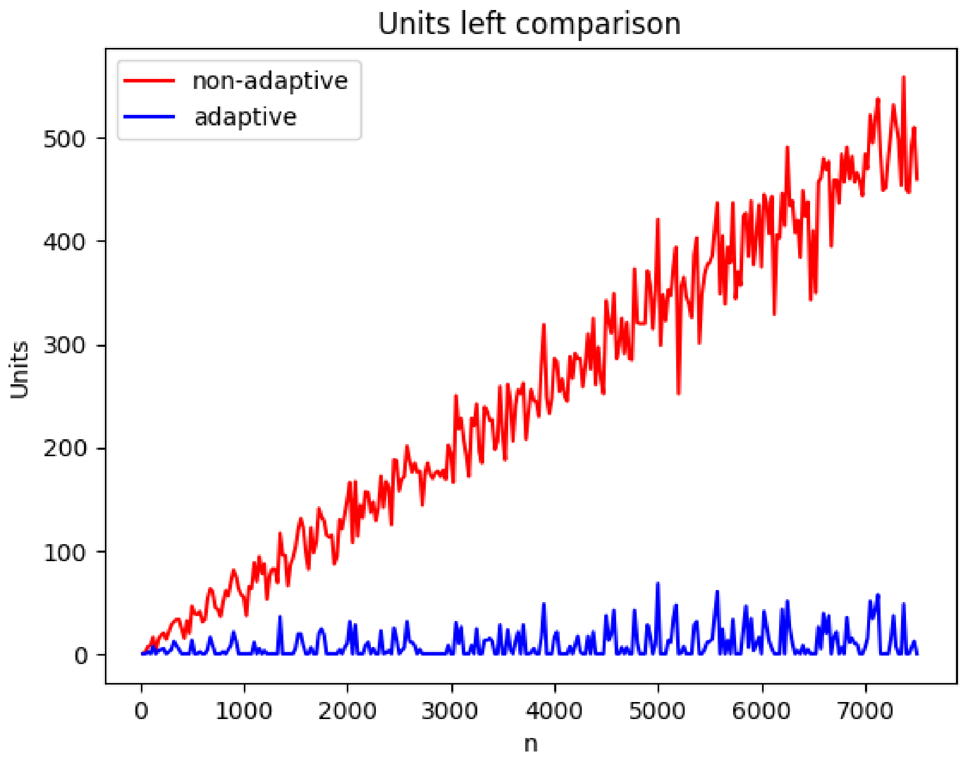
Simulation results for a = 0.5, d = 0.5, p = 0.5, c = 0.5



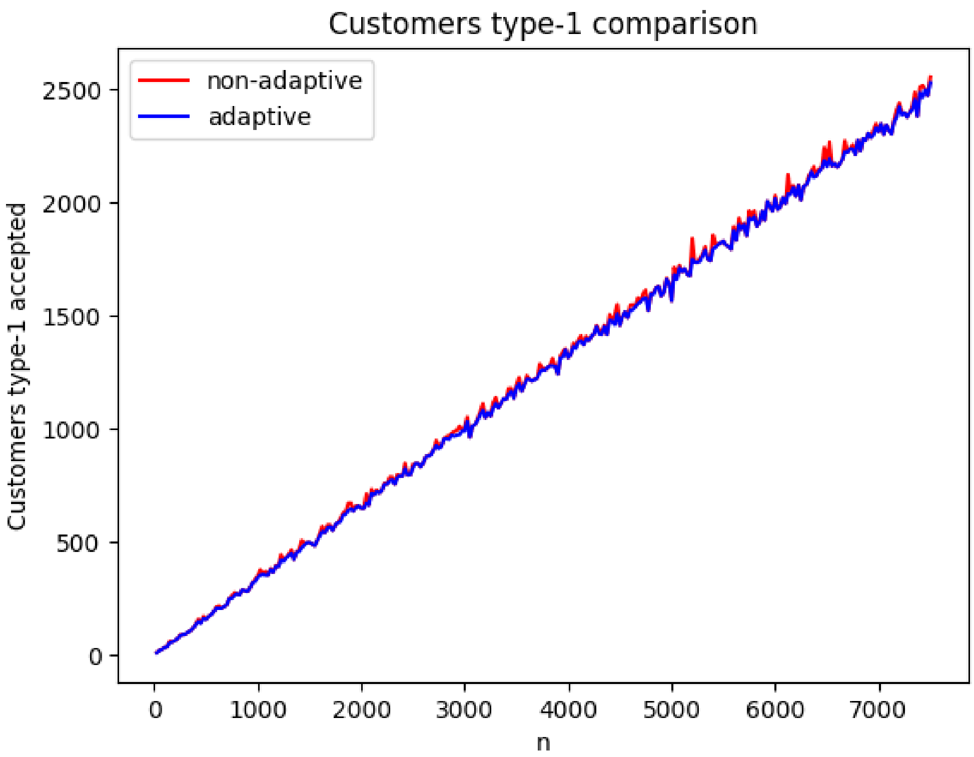
Simulation results for a = 0.5, d = 0.5, p = 0.5, c = 0.5



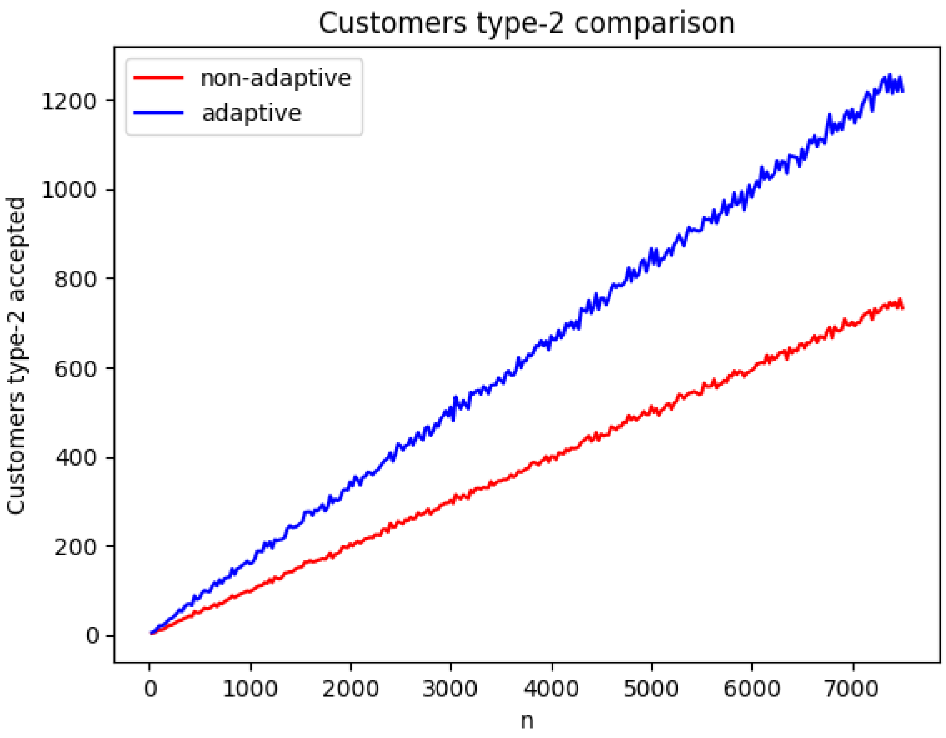
Simulation results for a = 0.5, d = 0.5, p = 0.5, c = 0.5



Simulation results for a = 0.5, d = 0.5, p = 0.5, c = 0.5



Simulation results for a = 0.5, d = 0.5, p = 0.5, c = 0.5



Simulation table (fragment)

a	d	p	c	Larger reward	Less computational time	Units left	Customers T1 and T2
0,2	0,8	0,2	0,2	A	NA	Units left increase with n, NA leaves more units	T1 - equally, A accepts more T2
0,5	0,2	0,2	0,5	NA	NA	No units left	NA accepts more T1 and less T2
0,2	0,2	0,2	0,8	A	NA	No units left	A accepts more T1 and less T2
0,5	0,5	0,2	0,5	A	NA	Units left increase with for NA n, NA leaves more units	T1 - equally, NA a bit more and less T2

Observations

- With larger d, the number of units left increases with n. The non-adaptive algorithm leaves more units, accepts less type-2 customers and, consequently, gains less reward (on average). Type-1 customers are accepted equally.
- However, with small d, the non-adaptive algorithm gains more reward as it accepts more type-1 customers and less type-2. In this case, none of the algorithms leaves unsold units.
- For medium d, the share of units left for the adaptive algorithm becomes smaller with increasing n. The adaptive algorithm gets more reward by accepting a bit less customers of type-1 and much more customers of type-2.

Observations

- The non-adaptive algorithm is much faster than the adaptive algorithm.
- The adaptive algorithm gains larger reward more often than the non-adaptive one.
- The non-adaptive algorithm tends to leave more units unsold, so it rejects too many type-2 customers.
- The adaptive algorithm gets larger reward for $a = 0.5$.
- With growing c, the adaptive algorithm rejects more type-2 customers.

References

- Hwang, D., Jaillet, P., and Manshadi, V. (2021). Online resource allocation under partially predictable demand. *Operations Research*, 69(3):895–915.
- Link to the Python code: https://colab.research.google.com/drive/1VozvH_0Yh9msRQJQxYwDIafiGUHvC9jN?hl=de#scrollTo=McmQsgaF_s4H

THANKS!

Any questions?