Bayesian Nash Equilibrium

by Alex Dolia, 11 Aug 2022

Bayesian Nash Equilibrium

We will consider an example of competition between $N+1$ sellers to sell the same item- the seller who offers the lowest price wins. Every seller has N competitors. In order to find the best bid price, we find Bayesian Nash Equilibrium by maximizing the expected utility function.

In the beginning, we consider the cases when the price or cost of competitors are uniformly distributed but in section 4 we give the example of an arbitrary distribution solution that is illustrated by Python code using a real dataset.

1. First Price Tender for Two Sellers with $C_j \sim \mathscr{U}(0, H)$.

Two sellers want to win the competition (tender) and sell the items. The seller who asks for the smallest price will win. There are other criteria for winning a bid like quality but we do not consider them in this Section.

We have the following utility function:

$$U(b_i, b_j, C_i) = \begin{cases} (b_i - C_i) & \text{if } b_i < b_j \\ 0 & \text{otherwise} \end{cases}$$

where C_i is the cost of making and delivering the product under consideration, we assume that

$C_j \sim \mathscr{U}(0, H)$, for example, *H* could be equal to 100.

The cumulative distribution function (CDF) for uniform distribution $\mathcal{U}(0, H)$ of random variable *y*can be written as $F(y) = \frac{y}{H}$.

Let us assume that $b_j = aC_j$, if the *i*th customer wins then $b_i < b_j$ *or* $b_i < aC_j$.

Therefore, $C_j > \frac{b_i}{a}$. As a result probability of winning by the *i*th customer is equal to the following (see Fig. 1):

$P(win) = 1 - F\left(\frac{b_i}{a}\right) = 1 - \frac{b_i}{aH}$.

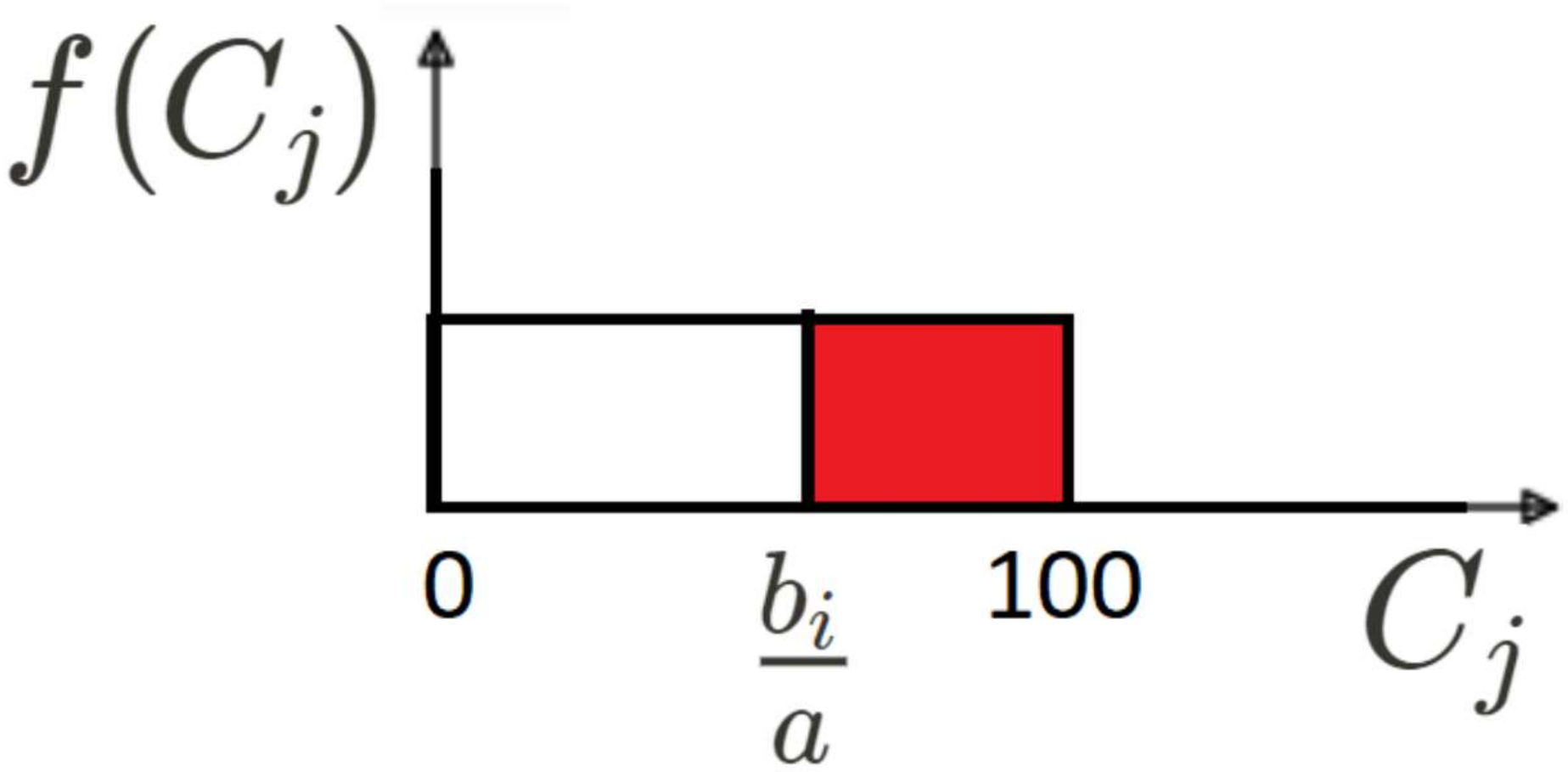


Fig. 1 Illustration of the probability of *ith* seller winning. The red area is equal to the probability of i seller winning (note that, in this example, $H = 100$).

The expected utility of win is equal to

$$\mathbb{E}\{U_{b_i<b_j}(b_i,b_j,C_i)\} = (b_i - C_i)P(win) = (b_i - C_i)(1 - \frac{b_i}{aH})$$

or

$$\mathbb{E}\{U(b_i,b_j,C_i)\} = (b_i - C_i)(1 - \frac{b_i}{aH}) = b_i - \frac{b_i^2}{aH} - C_i + \frac{C_i \cdot b_i}{aH}$$

In order to determine the value of b_i that maximizes the utility function we find the derivative of the expected utility with respect to b_i and set it to zero:

$$\frac{\partial \mathbb{E}\{U(b_i,b_j,C_i)\}}{\partial b_i} = \left(b_i - \frac{b_i^2}{aH} - C_i + \frac{C_i \cdot b_i}{aH}\right)' = 1 - \frac{2b_i}{aH} + \frac{C_i}{aH} = \frac{aH -}{aH} = \frac{aH - 2b_i + C_i}{aH} = 0$$

In order to find the maximum of the expected utility function we set the above numerator to 0 and solve it with respect to b_i:

$$aH - 2b_i \ + \ C_i \ = \ 0$$

$$b_i^* = \frac{C_i + aH}{2} = \frac{C_i + \max(b_j)}{2},$$

where the maximum price of the competitor by b_j is equal to the following:

$$\max\left(b_j\right) \ = \ aH.$$

We can also solve the following equation:

$$\frac{\partial \mathbb{E}\{U(b_i,b_j,C_i)\}}{\partial b_i} = 1 - \frac{2b_i}{aH} + \frac{C_i}{aH} = 0$$

using the Python script:

```
import sympy as sy
sy.init_printing()
import numpy as np
```

```
b_i, C_i, a, H, L = sy.symbols('b_i C_i a H L')   #Define Symbols
sy.solve(1 - 2*b_i/(a*H) + C_i/(a*H), b_i, force=True, manual=True, set=True)
```

It will give the same solution as above but written in a slightly different form:

$([b_i], (\frac{C_i}{2} + \frac{Ha}{2}))$.

The expected utility using the optimal price is equal to the following:

$$\mathbb{E}\{U(b_i, b_j, C_i)\} = (b_i - C_i)(1 - \frac{b_i}{aH}) = (\frac{C_i + aH}{2} - C_i)(1 - \frac{\frac{C_i + aH}{2}}{aH}) = \frac{aH - C_i}{2}\frac{2aH - (C_i + aH)}{2aH}$$

$$\mathbb{E}\{U(b_i, b_j, C_i)\} = \frac{aH - C_i}{2}\frac{2aH - (C_i + aH)}{2aH} = \frac{(aH - C_i)^2}{4aH}.$$

We illustrate our theoretical result in the following example.

Example 1. First Price Tender for Two Sellers with $C_j \sim \mathscr{U}(0, H)$.

$C_i = 5$, *H* = 10 Euros, *a* = 1.5. Therefore, the maximum price of the competitor *j* is 15 Euros, $\max(b_j) = aH = 1.5 \cdot 10 = 15$. The optimal price of the *i* seller is

$b_i^* = \frac{C_i + aH}{2} = \frac{5 + 15}{2} = 10$ Euros.

The following Python script perform tender simulation 1,000,000 times (see n = 1000000). Results of simulation confirm the theoretical results - the best bid price by seller *i* is equal to 10 Euros, *bi* = 10, in both theoretical and simulation case (see Table 1).

```
import pandas as pd
from numpy.random import uniform
import numpy as np
#
np.random.seed(11)
Ci = 5   # cost of seller i
```

```
H  = 10  # H is the upper bounf for the cost of seller j
a  = 1.5
n  = 1000000 # n is the number of tenders
aCj = uniform(0, H, n)
#
abj = a * aCjabi = range(int(max(abj))+1)
#
results = []
for bi in abi:
    P_win = len(abj[abj > bi]) / len(abj) # estimated probability of win
    EU    = (bi-Ci) * P_win
    elem = {}
    elem["optimal bi"       ] = np.round((Ci+a*H)/2,2)
    elem["considered bi"    ] = bi
    elem["estimated P_win"  ] = np.round(P_win, 2)
    elem["theoretical P_win"] = np.round(1-bi/(a*H), 2)
    elem["estimated EU"     ] = np.round(EU, 2)
    elem["maximum EU"       ] = np.round((a*H-Ci)**2 /(4*a*H), 2)
    results.append(elem)
#
results_df = pd.DataFrame(results)results_df
```

Table 1 Illustration of tender with two sellers where the cost of the second seller *j* is uniformly distributed between 0 and 10 Euros. Note that, if the bid price of seller *i* is less than its cost ($b_i < C_i = 5$) then the expected utility is negative or the seller *i* is losing money.

N	optimal price, i seller	considered price, i seller	estimated P_win	theoretical P_win	estimated EU	maximum EU
0	10	0	1	1	-5	1.67
1	10	1	0.93	0.93	-3.73	1.67
2	10	2	0.87	0.87	-2.6	1.67
3	10	3	0.8	0.8	-1.6	1.67
4	10	4	0.73	0.73	-0.73	1.67
5	10	5	0.67	0.67	0	1.67
6	10	6	0.6	0.6	0.6	1.67
7	10	7	0.53	0.53	1.07	1.67
8	10	8	0.47	0.47	1.4	1.67
9	10	9	0.4	0.4	1.6	1.67
10	10	10	0.33	0.33	1.67	1.67
11	10	11	0.27	0.27	1.6	1.67
12	10	12	0.2	0.2	1.4	1.67
13	10	13	0.13	0.13	1.06	1.67
14	10	14	0.07	0.07	0.6	1.67

The main result of this section is that if the bid price of the second seller (competitor) is uniformly distributed between 0 and some maximum price then the optimal price b_i^* for seller *i* does not depend on the smallest price of seller *j* but his maximum price. Obviously, it might not be the case for an arbitrary distribution of the price of seller *j*.

2. First Price Tender for Two Sellers with $C_j \sim \mathcal{U}(L, H)$.

In this section we consider the tender between sellers i and j where the seller i knows the cost of its item, C_i, and that the price of the same item of the seller j is uniformly distributed between aL and aH. In this section, we show that the optimal bid price b_i^* for seller i depends as in Section 1 on C_i and the highest price of seller j, $\max\ (b_j)$.

2.1 Optimal Price Derivation

We have the following utility function:

$$U(b_i, b_j, C_i) = \begin{cases} (b_i - C_i) & \text{if } b_i < b_j \\ 0 & \text{otherwise} \end{cases}$$

where $C_i$is the cost of making and delivering the product under consideration, we assume that

$C_j \ \sim \ \mathscr{U}(L,\ H)$.

$$F(y) = \begin{cases} 0 & \text{for } y < L \\ \frac{y-L}{H-L} & \text{for } y \in [L, H] \\ 1 & \text{for } y > H \end{cases}$$

Let us assume that $b_j \ = \ aC_j$ if the ith customer wins then $b_i \ < \ b_j$ *or* $b_i \ < \ aC_j$.

Therefore, $C_j > \frac{b_i}{a}$. As a result probability of winning by the *i*th customer is equal to

$$P(win) = 1 - F\left(\frac{b_i}{a}\right) = \begin{cases} 1 & \text{for } \frac{b_i}{a} < L \\ 1 - \frac{\frac{b_i}{a} - L}{H-L} & \text{for } \frac{b_i}{a} \in [L, H] \\ 0 & \text{for } \frac{b_i}{a} > H \end{cases}$$

or

$$P(win) = 1 - F\left(\frac{b_i}{a}\right) = \begin{cases} 1 & \text{for } b_i < aL \\ 1 - \frac{b_i - aL}{a(H-L)} & \text{for } b_i \in a \cdot [L, H] \\ 0 & \text{for } b_i > aH \end{cases}$$

when $b_i \in [aL, aH]$ then $P(win)$ equals the following:

$1 - \frac{b_i - aL}{a(H-L)} = \frac{a(H-L)-(b_i-aL)}{a(H-L)} = \frac{aH-aL-b_i+aL}{a(H-L)} = \frac{aH-b_i}{a(H-L)}$

Therefore, $P(win)$is equal to

$$P(win) = 1 - F(\frac{b_i}{a}) = \begin{cases} 1 & \text{for } b_i < aL \\ \frac{aH-b_i}{a(H-L)} & \text{for } b_i \in [aL, aH] \\ 0 & \text{for } b_i > aH \end{cases}$$

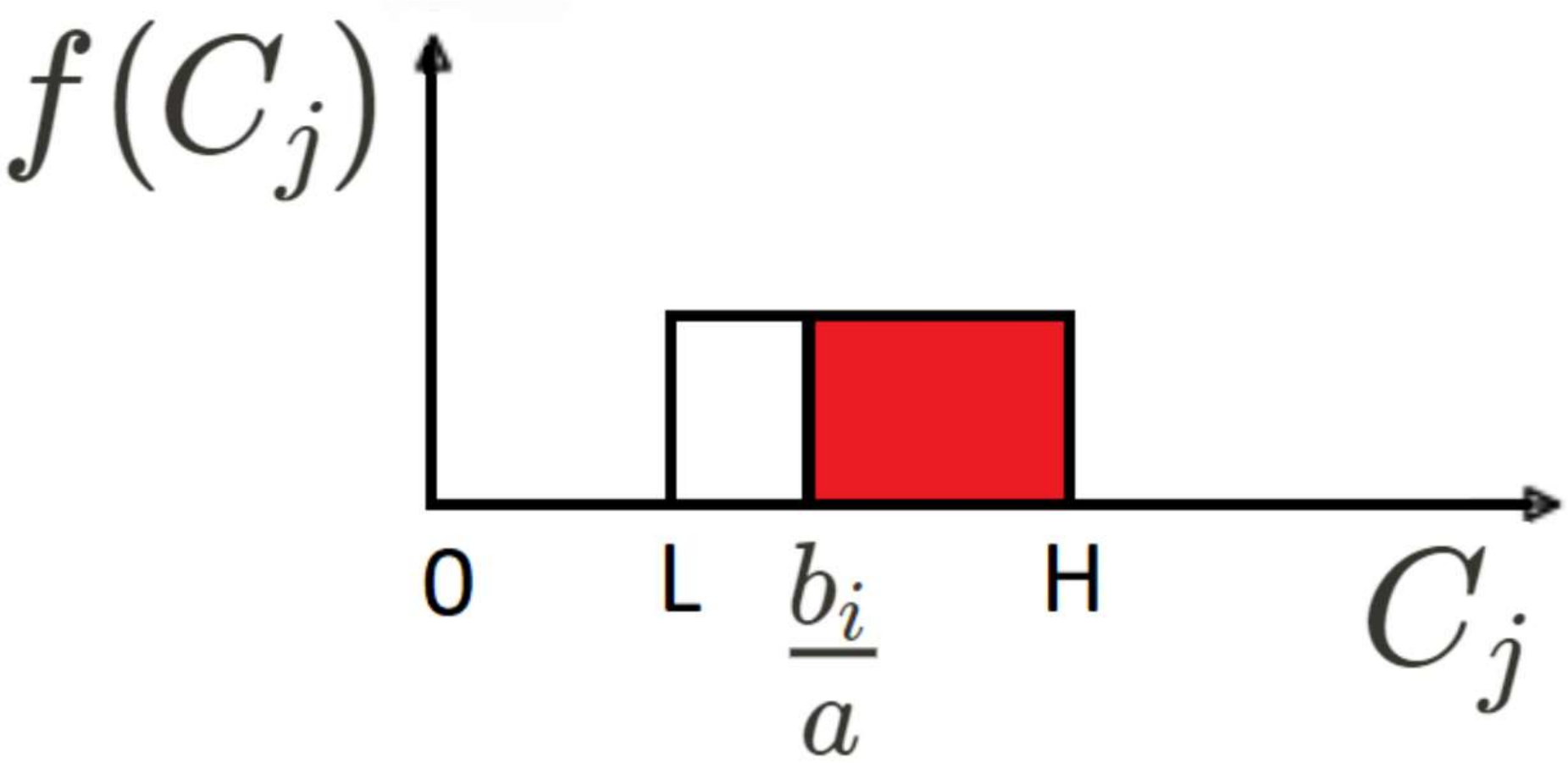


Fig. 2 Illustration of the probability of the ith seller winning when j seller has the item cost drawn from the uniform distribution defined by L and H. The red area is equal to the probability of the ith seller winning.

If $b_i \leq aL$ then the expected utility is maximized when $b_i^* = aL$ and, therefore, the expected utility is equal to the following:

$$\mathbb{E}Ub_i \leq aL(b_i, b_j, C_i) = (b_i - C_i)P(win) = (aL - C_i)P(win) = aL - C_i.$$

But when $b_i \in [aL, aH]$ then the expected utility can be computed as follows:

$$\mathbb{E}\{U_{b_i \in [aL,aH]}(b_i, b_j, C_i)\} = (b_i - C_i)\frac{aH - b_i}{a(H-L)}$$

In order to determine the value of *bi* that maximizes the expected utility function when $b_i \in [aL, aH]$we find the derivative of the expected utility with respect to b_i and set it to zero:

$$\frac{\partial \mathbb{E}\{U_{b_i \in [aL,aH]}(b_i,b_j,C_i)\}}{\partial b_i} = ((b_i - C_i)\frac{aH - b_i}{a(H-L)})' = \frac{aH - b_i}{a(H-L)} - \frac{b_i - C_i}{a(H-L)}$$

where we use the following Leibniz product rule to find the derivative:

$$\frac{\partial(u \cdot \nu)}{\partial x} = \frac{\partial u}{\partial x} \cdot \nu + u \cdot \frac{\partial \nu}{\partial x},$$

$$u = (b_i - C_i);$$

$$\nu = \frac{aH - b_i}{a(H-L)}.$$

After some manipulation we got the following:

$$\frac{\partial \mathbb{E}\{U_{b_i \in [aL,aH]}(b_i,b_j,C_i)\}}{\partial b_i} = \frac{aH - b_i - b_i + C_i}{a(H-L)} = \frac{-2b_i + aH + C_i}{a(H-L)}.$$

In order to find the extreme points (minimum or maximum) we set the above derivative to zero:

$$\frac{\partial \mathbb{E}\{U_{b_i \in [aL,aH]}(b_i,b_j,C_i)\}}{\partial b_i} = \frac{-2b_i + aH + C_i}{a(H-L)} = 0.$$

Therefore the derivative is equal to zero when the above numerator is equal to zero:

$$-2b_i + aH + C_i = 0.$$

Then the optimal price for the seller i can be calculated as the following:

$b_i^* = \frac{C_i + aH}{2} = \frac{C_i + \max(b_j)}{2}$.

We can solve the following equation:

$\frac{\partial \mathbb{E}\{U_{b_i \in [aL, aH]}(b_i, b_j, C_i)\}}{\partial b_i} = \frac{-2b_i + aH + C_i}{a(H - L)} = 0$

also using sympy Python library as following:

```
import sympy as sy
sy.init_printing()
import numpy as np

b_i, C_i, a, H, L = sy.symbols('b_i C_i a H L')   #Define Symbols

sy.solve((-2*b_i + a * H + C_i) / (a * (H-L)), b_i,
           force=True, manual=True, set=True)
```

It will give the same solution but written in a slightly different form:

$([b_i], (\frac{C_i}{2} + \frac{Ha}{2}))$.

In order to find the optimal price b_i^* in the interval $b_i \in [aL, aH]$ we need to sum the cost of the item for the *ith seller* with the highest possible price for the seller j and divide the sum of these two values by two.

2.2 Comparison of Optimal Prices for $b_i \leq aL$ and $b_i \in [aL, aH]$

In Section 2.1 we found two optimal prices for seller i: for $b_i \leq aL$ and $b_i \in [aL, aH]$

$$b_i^* = \begin{cases} aL & \text{for } b_i \leq aL \\ \frac{C_i + aH}{2} & \text{for } b_i \in [aL, aH] \end{cases}.$$

Obviously, the higher the price the higher is maximum possible profit. Based on this point of view we need to select $b_i \in [aL, aH]$. But we might have a low probability of winning. Therefore, we need to select an optimal price that has the maximum expected utility.

If $b_i \leq aL$ then the expected utility is maximized when $b_i = aL$ and, therefore, the expected utility is equal to

$$\mathbb{E}\{U_{b_i \leq aL}(b_i, b_j, C_i)\} = (b_i - C_i)P(win) = (aL - C_i)P(win) = aL - C_i.$$

But when $b_i \in [aL, aH]$ then the expected utility can be computed as follows:

$$\mathbb{E}\{U_{b_i \in [aL,aH]}(b_i, b_j, C_i)\} = (b_i - C_i)\frac{aH - b_i}{a(H-L)} = (\frac{C_i + aH}{2} - C_i)\frac{aH - \frac{C_i + aH}{2}}{a(H-L)}$$

where on the right-hand side of the above expression we substitute $\frac{C_i + aH}{2}$ instead of b_i. After some massage of the above expression we get the following expected utility:

$$\mathbb{E}\{U_{b_i \in [aL,aH]}(b_i, b_j, C_i)\} = \frac{aH - C_i}{2}\frac{aH - C_i}{2a(H-L)} = \frac{1}{a(H-L)}(\frac{aH - C_i}{2})^2.$$

Therefore the question arises whether should we choose $b_i \leq aL$ or $b_i \in [aL, aH]$.

Or in other words, we need to determine whether if $\mathbb{E}U_{b_i \leq aL}(b_i, b_j, C_i)$ is greater than$\mathbb{E}Ub_i \in [aL, aH](b_i, b_j, C_i)$ or vice versa, $\mathbb{E}U_{b_i \in [aL, aH]}(b_i, b_j, C_i) \geq \mathbb{E}U_{b_i \leq aL}(b_i, b_j, C_i)$.

In order to answer this question, we need from the expected utility obtained for *bi* ∈ [*aL*, *aH*] subtract the expected utility that corresponds to *bi* ≤ *aL*:

$\mathbb{E}\{U_{b_i \in [aL, aH]}(b_i,b_j, C_i)\} - \mathbb{E}\{U_{b_i \leq a L}(b_i,b_j, C_i)\}=\frac{1}{a(H-L)}(\frac{aH-C_i}{2})^2-(aL-C_i)$,

$\frac{1}{a(H-L)}(\frac{aH-C_i}{2})^2-(aL-C_i)=\frac{a^2H^2-2aHC_i+C^2_i}{4a(H-L)}-(aL-C_i)$

We want to find the smallest value of the expected utility difference with respect to cost C_i and check if it is positive:

$\min_{Ci \geq 0}[\mathbb{E}U_{b_i \in [aL, aH]}(b_i, b_j, C_i) - \mathbb{E}U_{bi \leq aL}(b_i, b_j, C_i)] \geq 0$

If it is indeed positive then

$\mathbb{E}U_{b_i \in [aL, aH]}(b_i, b_j, C_i) \geq \mathbb{E}U_{b_i \leq aL}(b_i, b_j, C_i)$ and we need to select the bid price in $b_i \in [aL, aH]$:

$b_i = \frac{C_i+aH}{2}$.

We need to find the following derivative, set it to zero and solve the equation with respect to *Ci*:

$\frac{\partial[\mathbb{E}\{U_{b_i\in[aL,aH]}(b_i,b_j,C_i)\}-\mathbb{E}\{U_{b_i\leq aL}(b_i,b_j,C_i)\}]}{\partial C_i} = (\frac{a^2H^2-2aHC_i+C_i^2}{4a(H-L)} - (aL - C_i))'$

$\frac{\partial[\mathbb{E}\{U_{b_i\in[aL,aH]}(b_i,b_j,C_i)\}-\mathbb{E}\{U_{b_i\leq aL}(b_i,b_j,C_i)\}]}{\partial C_i} = \frac{-2aH+2C_i}{4a(H-L)} + 1 = \frac{-aH+C_i}{2a(H-L)} + 1$

Set derivative to zero and find an extreme point of C_i:

$\frac{-aH+C_i}{2a(H-L)} + 1 = 0$

$C_i^* = -2aH + 2aL + aH = a(2L - H).$

When $C_i = a(2L - H)$ then we get the minimum of the expected utility difference:

$C_i^* = \arg\min_{C_i\geq 0}[\mathbb{E}\{U_{b_i\in[aL,aH]}(b_i,b_j,C_i)\} - \mathbb{E}\{U_{b_i\leq aL}(b_i,b_j,C_i)\}] = a(2L - H).$

In order to prove it we need to find the second derivative with respect to C_i and if it is positive then the extreme point C_i^* is location of the minimum:

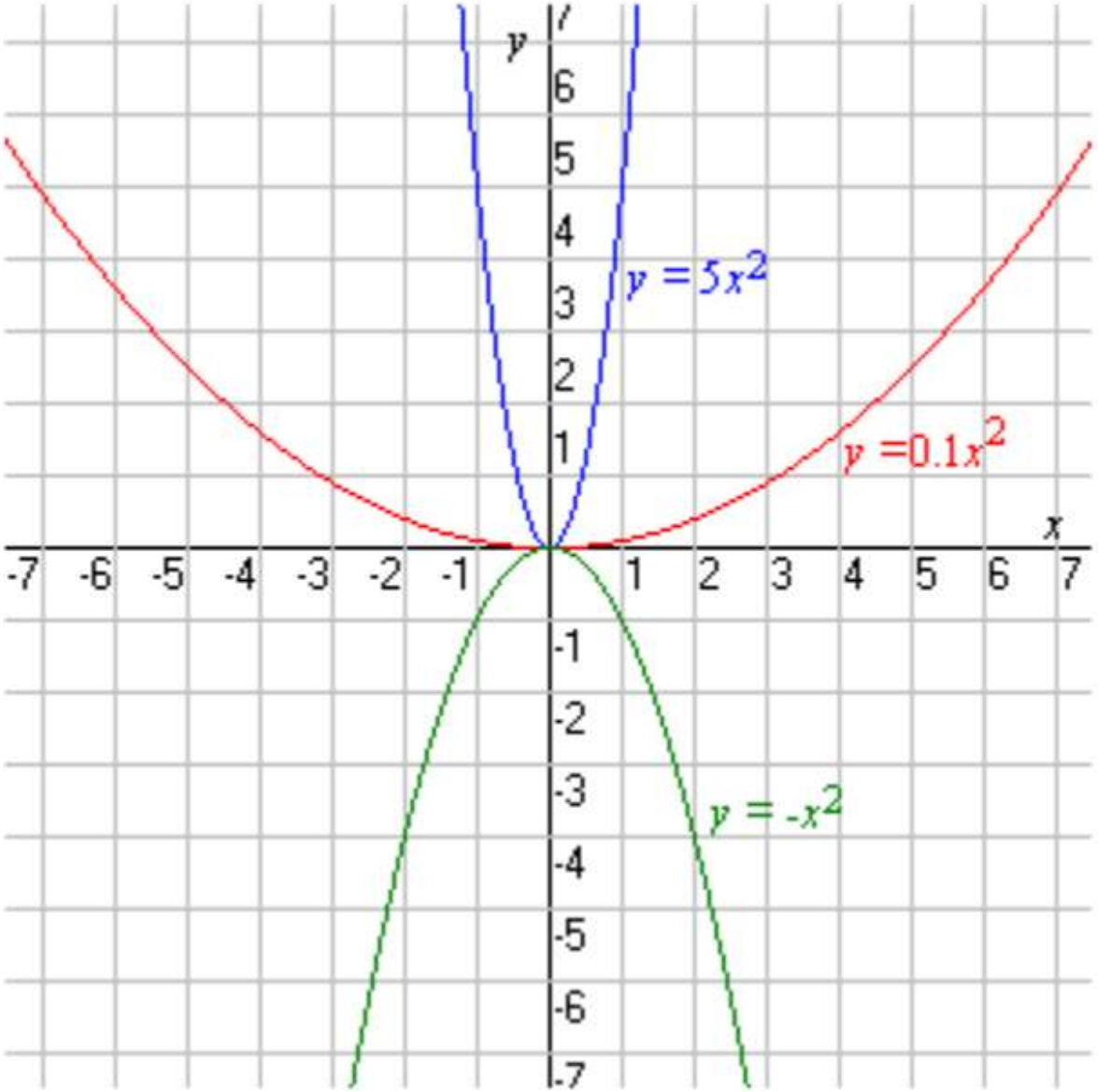
$\frac{\partial^2[\mathbb{E}\{U_{b_i\in[aL,aH]}(b_i,b_j,C_i)\}-\mathbb{E}\{U_{b_i\leq aL}(b_i,b_j,C_i)\}]}{\partial C_i^2} = \frac{1}{2a(H-L)} \geq 0,$

where $a \geq 0$ and $H - L \geq 0$ because $H \geq L$.

You can show it in a different way. The following expression is an analytical expression of a parabola with respect to C_i:

$\frac{a^2H^2 - 2aHC_i + C_i^2}{4a(H-L)} - (aL - C_i) = \frac{1}{4a(H-L)}C_i^2 + \frac{0.5H-L}{H-L}C_i + (\frac{a^2H^2}{4a(H-L)} - aL)$

If the coefficient of C_i^2 is positive, the parabola opens up; otherwise, it opens down. In our case, the coefficient of C_i^2 is positive, $\frac{1}{4a(H-L)} > 0$ (see $\frac{1}{4a(H-L)}C_i^2$ in the above expression), therefore the parabola opens up or has a minimum. See the following picture for illustration:


y
x
$y = 5x^2$
$y = 0.1x^2$
$y = -x^2$
-7 -6 -5 -4 -3 -2 -1 1 2 3 4 5 6 7
6 5 4 3 2 1 -1 -2 -3 -4 -5 -6 -7

Now in order to prove that the minimum of the expected utility difference is positive we substitute the value of C_i^* instead of C_i in the expected utility function:

$$\mathbb{E}\{U_{b_i \in [aL,aH]}(b_i, b_j, C_i^*)\} - \mathbb{E}\{U_{b_i \leq aL}(b_i, b_j, C_i^*)\} = \frac{(aH - C_i^*)^2}{4a(H-L)} - (aL - C_i^*) =$$

$$\frac{(aH - a(2L-H))^2}{4a(H-L)} - (aL - a(2L-H)) = \frac{(2aH-2aL)^2}{4a(H-L)} - \frac{4a(H-L)a(H-L)}{4a(H-L)} =$$

$$\frac{(2aH-2aL)^2}{4a(H-L)} - \frac{4a(H-L)a(H-L)}{4a(H-L)} = \frac{(2a(H-L))^2 - (2a(H-L))^2}{4a(H-L)} = 0$$

$$\mathbb{E}\{U_{b_i \in [aL, aH]}(b_i, b_j, C_i^*)\} - \mathbb{E}\{U_{b_i \leq aL}(b_i, b_j, C_i^*)\} = 0.$$

What we have shown is that the expected utility for the optimal price in the range $b_i \leq aL$, $\mathbb{E}\{U_{b_i \leq aL}(b_i, b_j, C_i^*)\}$, cannot be larger than the expected utility for the price in the range $b_i \in [aL, aH]$, $U_{b_i \in [aL, aH]}(b_i, b_j, C_i^*)$. Therefore, we should choose the price in the range $b_i \in [aL, aH]$ or in other words:

$b_i^* = \frac{C_i + aH}{2}$.

This result is supported by simulation (see Example 2 and Table 2).

Example 2. First Price Tender for Two Sellers with $C_j \sim \mathcal{U}(L, H)$.

$C_i = 5$, $L = 4$ Euros, $H = 10$ Euros and $a = 1.5$. Therefore, the maximum price of the competitor *j* is 15 Euros, $\max(b_j) = aH = 1.5 \cdot 10 = 15$. The optimal price of the *i* seller is

$b_i^* = \frac{C_i + aH}{2} = \frac{5+15}{2} = 10$ Euros. Note that, the optimal bid price b_i^* does not depend on L or the lowest price of the seller j, $\min(b_j) = aL$, and it is the same for $C_j \sim \mathcal{U}(0, H)$ and $C_j \sim \mathcal{U}(L, H)$. We confirm our theoretical results by simulation (see the following Python script and Table 2).

```
import pandas as pd
from numpy.random import uniform
import numpy as np
#
np.random.seed(11)
Ci = 5   # cost of seller i
L  = 4H  = 10
# H is the upper bound for the cost of seller j
a  = 1.5n  = 1000000 # n is the number of tenders
aCj = uniform(L, H, n)
#
abj = a * aCj
abi = range(int(max(abj))+1)
#
results = []
for bi in abi:
    P_win = len(abj[abj > bi]) / len(abj)
    EU    = (bi-Ci) * P_win
    elem = {}
    elem["optimal bi"       ] = np.round((Ci+a*H)/2,2)
    elem["considered bi"    ] = bi
    elem["estimated P_win"  ] = np.round(P_win, 2)
    if   (bi <= a*L):
         elem["theoretical P_win"] = 1
         elem["maximum EU"       ] = np.round(a*L - Ci, 2)
    elif (bi > a*L) and (bi <= a*H):
          elem["theoretical P_win"] = np.round((a*H-bi)/(a*(H-L)), 2)
          elem["maximum EU"       ] = np.round((a*H-Ci)**2 /(4*a*(H - L)), 2)
    elem["estimated EU"     ] = np.round(EU, 2)
    results.append(elem)
#
results_df = pd.DataFrame(results)
```

```
results_df[['optimal bi', 'considered bi', 'estimated P_win', 'theoretical P_win',
            'estimated EU', 'maximum EU']]
```

Table 2 Illustration of tender with two sellers where the cost of the second seller *j* is uniformly distributed between 4 and 10 Euros. Note that, unlike Example 1 in Example 2 the maximum expected utility has different values for different price ranges: $b_i <= aL$ and $b_i \in [aL, aH]$. Besides, the total maximum of expected utility is higher in Example 2 than in Example 1.

N	optimal price, i seller	considered price, i seller	estimated P_win	theoretical P_win	estimated EU	maximum EU
0	10	0	1	1	-5	1
1	10	1	1	1	-4	1
2	10	2	1	1	-3	1
3	10	3	1	1	-2	1
4	10	4	1	1	-1	1
5	10	5	1	1	0	1
6	10	6	1	1	1	1
7	10	7	0.89	0.89	1.78	2.78
8	10	8	0.78	0.78	2.33	2.78
9	10	9	0.67	0.67	2.66	2.78
10	10	10	0.55	0.56	2.77	2.78
11	10	11	0.44	0.44	2.66	2.78
12	10	12	0.33	0.33	2.33	2.78
13	10	13	0.22	0.22	1.78	2.78
14	10	14	0.11	0.11	0.99	2.78

As in Section 1 the optimal bid price b_i^* does not depends on the lowest price of seller j but on the cost $C_i$and the maximum price of seller j.

3. First Price Tender for $N\ +\ 1$ Sellers with $C_{j\ \neq\ i}\ \sim\ \mathscr{U}\left(L_j,\ H_j\right)$ and Given Constants $a_{j\ \neq\ i}$.

In this scenario the seller i compete against N other sellers $j\ \neq\ i$ where the competitors of seller i have the cost of their item C_j distributed as uniform random variable between L_j and H_j, $j\ =\ 1,\ \ldots,\ N$, and the price b_j is a linear function of cost, $b_j\ =\ a_jC_j$ where a_j some constant. The task is to find the optimum price b_i for seller i. The seller who offers the lowest price win.

3.1 Derivative of Expected Utility for *N* + 1 Sellers.

The maximum of the expected utility for seller *i* and ∀*j* ≠ *i* with respect to *bi* is equal to the following:

$b_i^* = \arg\max_{b_i} \mathbb{E}\{U_i(b_i, b_{j=1,..,N}, C_i)\} = \arg\max_{b_i}(b_i - C_i)\prod_{j=1}^{N} P_j(win).$

or

$b_i^* = \arg\max_{b_i} \mathbb{E}\{U_i(b_i, b_{j=1,..,N}, C_i)\} = \arg\max_{b_i} \log[\mathbb{E}\{U_i(b_i, b_{j=1,..,N}, C_i)\}]$

$b_i^* = \arg\max_{b_i} \log[\mathbb{E}\{U_i(b_i, b_{j=1,..,N}, C_i)\}] = \arg\max_{b_i} \log[(b_i - C_i)\prod_{j=1}^{N} P_j(win)]$

$b_i^* = \arg\max_{b_i} \log[(b_i - C_i)\prod_{\forall j\neq i} P_j(win)] = \arg\max_{b_i} \log(b_i - C_i) + \sum_{j=1}^{N} \log P_j(win)$

$$P_j(win) = 1 - F_1(\tfrac{b_i}{a_j}) = \begin{cases} 1 & \text{for } b_i < a_jL_j \\ 1 - \frac{b_i - a_jL_j}{a_j(H_j - L_j)} & \text{for } b_i \in a_j \cdot [L_j, H_j] \\ 0 & \text{for } b_i > a_jH_j \end{cases}$$

or

$$P_j(win) = 1 - F_1(\tfrac{b_i}{a_j}) = \begin{cases} 1 & \text{for } b_i < a_j L_j \\ \frac{aH_j - b_i}{a_j(H_j - L_j)} & \text{for } b_i \in a_j \cdot [L_j, H_j] \text{ - } \\ 0 & \text{for } b_i > a_j H_j \end{cases}$$

In order to find the maximum of the expected utility we need to find its derivative with respect to b_i, set it to zero and find *bi* that solves this equation:

$\frac{\partial \log[\mathbb{E}\{U_i(b_i, b_{j=1,..,N}, C_i)\}]}{\partial b_i} = \frac{1}{b_i - C_i} + \sum_{j=1}^{N} \frac{\partial \log P_j(win)}{\partial b_i} = 0$

$$\frac{\partial \log P_j(win)}{\partial b_i} = \begin{cases} 0 & \text{for } b_i < a_j L_j \\ -\frac{1}{a_j H_j - b_i} & \text{for } b_i \in [a_j L_j, a_j H_j] \end{cases}$$

where we obtain the derivative of log *Pj*(*win*) with respect to b_i as follows:

$\frac{\partial \log \frac{a_j H_j - b_i}{a_j(H_j - L_j)}}{\partial b_i} = \frac{\partial[\log(aH_j - b_i) - \log(a_j(H_j - L_j))]}{\partial b_i} = \frac{\partial \log(aH_j - b_i)}{\partial b_i} - \frac{\partial \log(a_j(H_j - L_j))}{\partial b_i} = \frac{\partial \log(aH_j - b_i)}{\partial b_i}.$

$\frac{\partial \log \frac{a_j H_j - b_i}{a_j(H_j - L_j)}}{\partial b_i} = \frac{\partial \log(aH_j - b_i)}{\partial b_i} = -\frac{1}{a_j H_j - b_i}$

This result will be used to derive the optimal price *bi* for seller i in the following subsection.

3.2 First Price Tender for $N + 1$ Sellers with $C_{j \neq i} \sim \mathscr{U}(L, H)$ and Given Constant *a*.

Now we consider a particular case when $L_1 = L_2 ... = L_N$, $H_1 = H_2 ... = H_N$ and $a_1 = a_2 ... = a_N$. If all sellers $\forall j \neq i$ have the same L, H, and a then when the price of seller i is less or equal to aL, $b_i \leq aL$, the expected utility function

is maximized when $b_i^* = aL$. Therefore, in this case, we need to estimate the lowest possible price equal to aL and set it to the bid price.

The expected utility is equal to the following:

$$\mathbb{E}U_i(b_i, b_{j=1,..,N}, C_i) = (b_i - C_i)P(win) = (aL - C_i)$$

where $P(win) = 1$ for $b_i < aL$.

If seller i selects the optimal price in the interval between aL and aH, $b_i \in [aL, aH]$, and for all sellers $\forall j \neq i$ we have the same L, H , and a then we can obtain the optimal price b_i^* for the seller *i* using the following manipulations:

$$\frac{1}{b_i - C_i} + N\frac{\partial \log P_j(win)}{\partial b_i} = 0$$

When $b_i \in [aL, aH]$ The derivative of $P_j(win)$ with respect to b_i is equal to

$\frac{\partial \log P_j(win)}{\partial b_i} = -\frac{1}{aH - b_i}$, therefore, after substitution this derivative in the above equation we get the following:

$$\frac{1}{b_i - C_i} - N\frac{1}{aH - b_i} = 0.$$

We need to find the value of *bi* from the above equation:

$$aH - b_i - N(b_i - C_i) = 0$$

$$-b_i(1 + N) = -NC_i - aH$$

$$b_i(1 + N) = NC_i + aH$$

$$b_i^* = \frac{NC_i + aH}{1 + N}.$$

If the number of competitors goes to infinity then the bid price b_i should be equal to the cost:

$$\lim_{N\to\infty} \frac{NC_i + aH}{1 + N} = \lim_{N\to\infty}[\frac{NC_i}{1 + N} + \frac{aH}{1 + N}] = \lim_{N\to\infty}[\frac{C_i}{1/N + 1} + \frac{aH}{1 + N}] = C_i.$$

If $N = 1$ then we get the same results as before when there are only two sellers in the tender:

$b_i^* = \frac{NC_i+aH}{1+N} = \frac{1 \cdot C_i+aH}{1+1} = \frac{C_i+aH}{2}$.

Therefore, in this scenario, the more competitors the seller i has the less is optimal price.

We can find the expected utility function for the optimal price *bi** using the following manipulations:

$$\mathbb{E}\{U_i(b_i, b_{j=1,..,N}, C_i)\} = (b_i - C_i)P(win) = (\frac{NC_i+aH}{1+N} - C_i)\frac{aH-\frac{NC_i+aH}{1+N}}{a(H-L)}N$$

$$\frac{NC_i+aH-C_i-NC_i}{(1+N)^2}\frac{aH+aNH-NC_i-aH}{a(H-L)}N = \left(\frac{N}{1+N}\right)^2\frac{(aH-C_i)^2}{a(H-L)}$$

Therefore, the expected utility function for the optimal price *bi** when *bi* ∈ [*aL*, *aH*] is equal to:

$$\mathbb{E}\{U_i(b_i, b_{j=1,..,N}, C_i)\} = \left(\frac{N}{1+N}\right)^2\frac{(aH-C_i)^2}{a(H-L)}$$

If the seller i has only one competitor, *N* = 1, then we get the expected utility function equal to the following:

$$\mathbb{E}\{U_i(b_i, b_j, C_i)\} = \left(\frac{N}{1+N}\right)^2\frac{(aH-C_i)^2}{a(H-L)} = \left(\frac{1}{1+1}\right)^2\frac{(aH-C_i)^2}{a(H-L)} = \frac{(aH-C_i)^2}{4a(H-L)}$$

That is, we get the same results for two sellers as in Section 2.2 (see $\mathbb{E}\{Ubi \in [aL, aH](bi, bj, Ci)\}$).

3.3 First Price Tender for 3 Sellers with $C_{j \neq i} \sim \mathscr{U}(L_j, H_j)$and Given Constants $a_{j \neq i}$.

Different examples of (L_1, H_1) and(L_2, H_2) (see Fig. 3). the example that is illustrated by Fig. 3.1 is a particular case considered in Section 3.2 when *N* = 2.

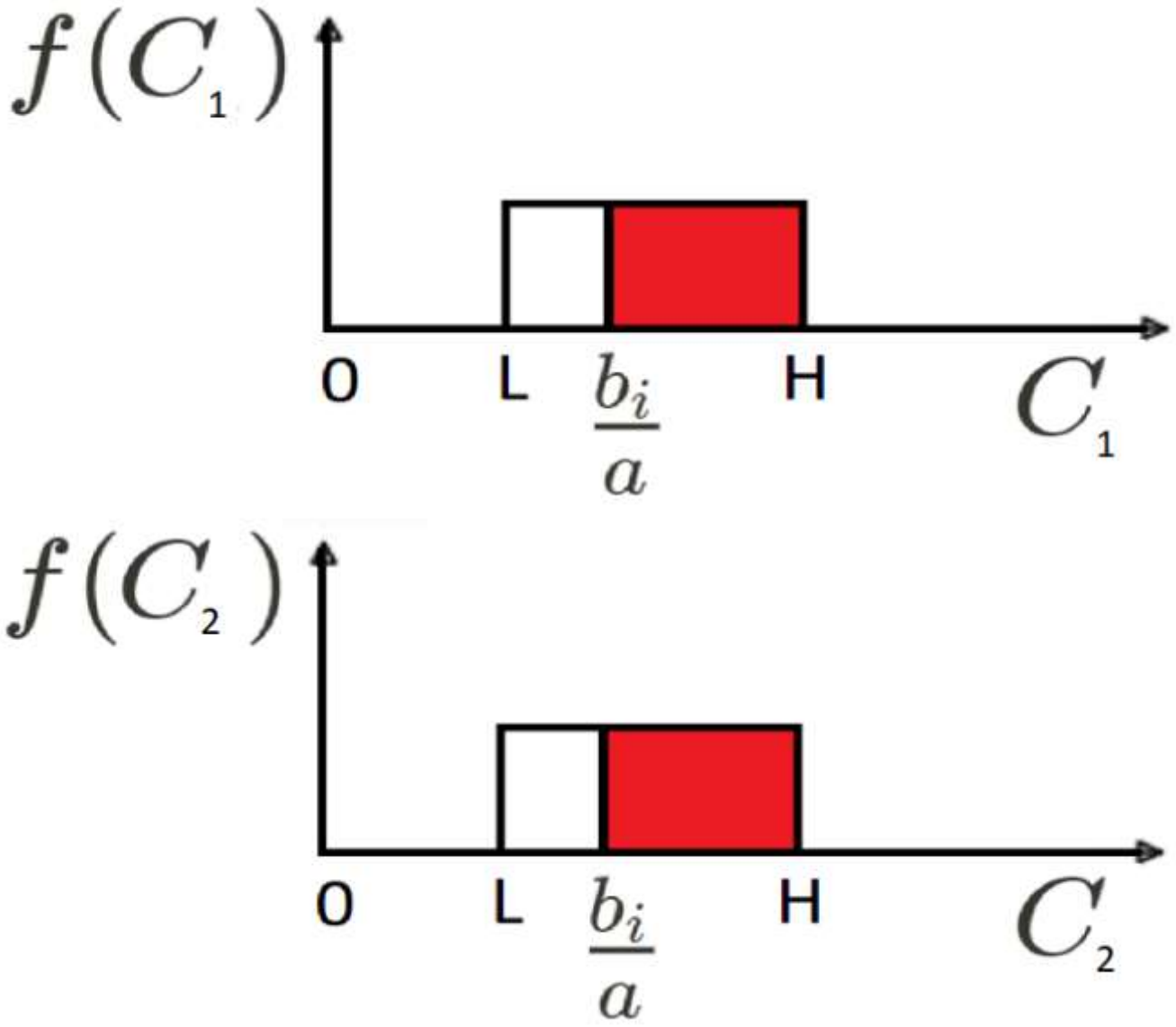

$f(C_1)$
O
L
$\frac{b_i}{a}$
H
C_1
$f(C_2)$
O
L
$\frac{b_i}{a}$
H
C_2

Fig. 3.1 Illustration of tender when competitors of the seller i have the same uniform distribution for their cost: $L_1 = L_2 = L$ and $H_1 = H_2 = H$.

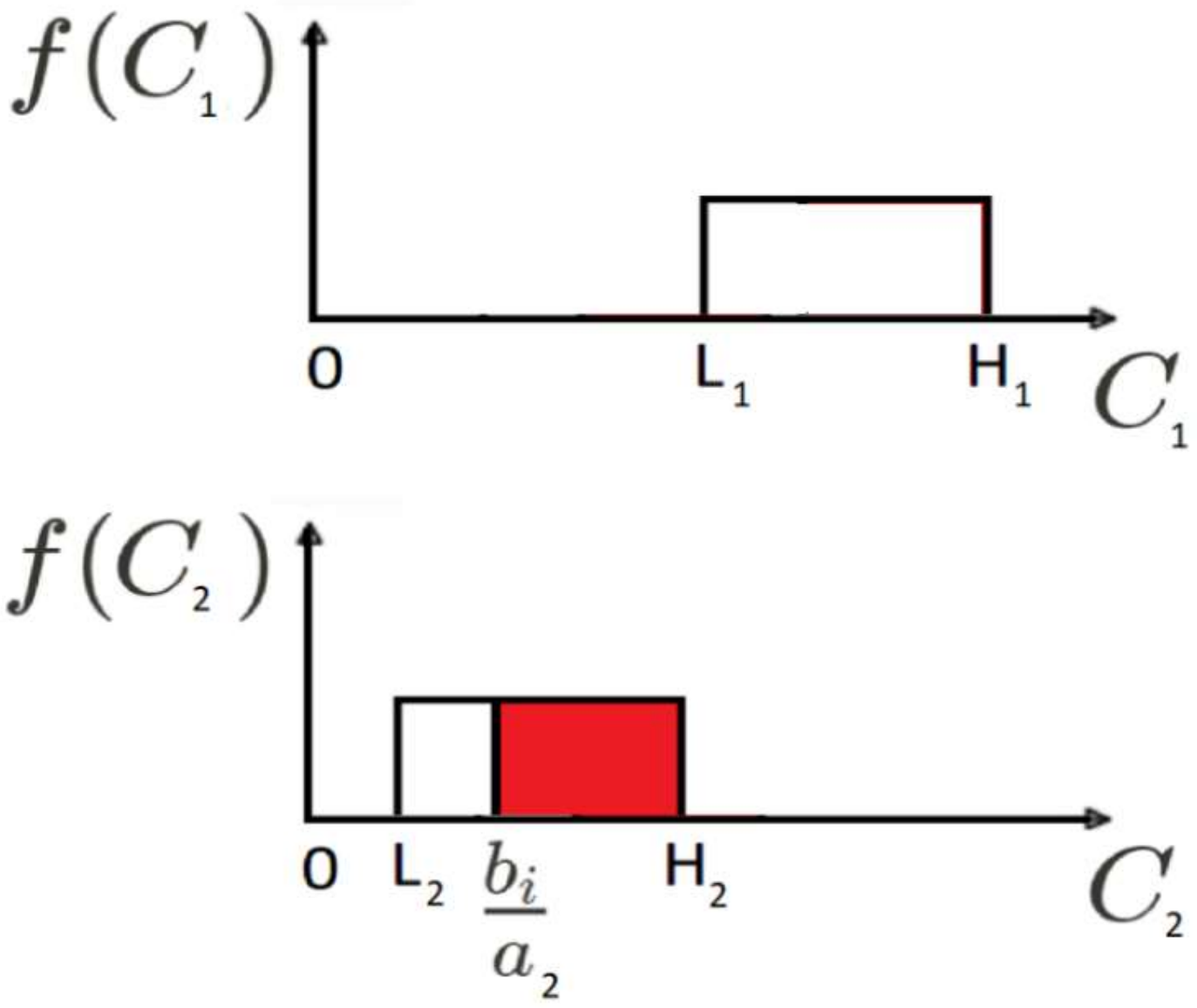

$f(C_1)$
0
L_1
H_1
C_1
$f(C_2)$
0
L_2
$\frac{b_i}{a_2}$
H_2
C_2

Fig. 3.2 Illustration of the tender when competitors of seller i have a non-overlap uniform distribution of their cost, $H2 < L1$. This case is equivalent to the case when we have only seller i and $j = 2$ sellers in the tender (see Section 2.1).

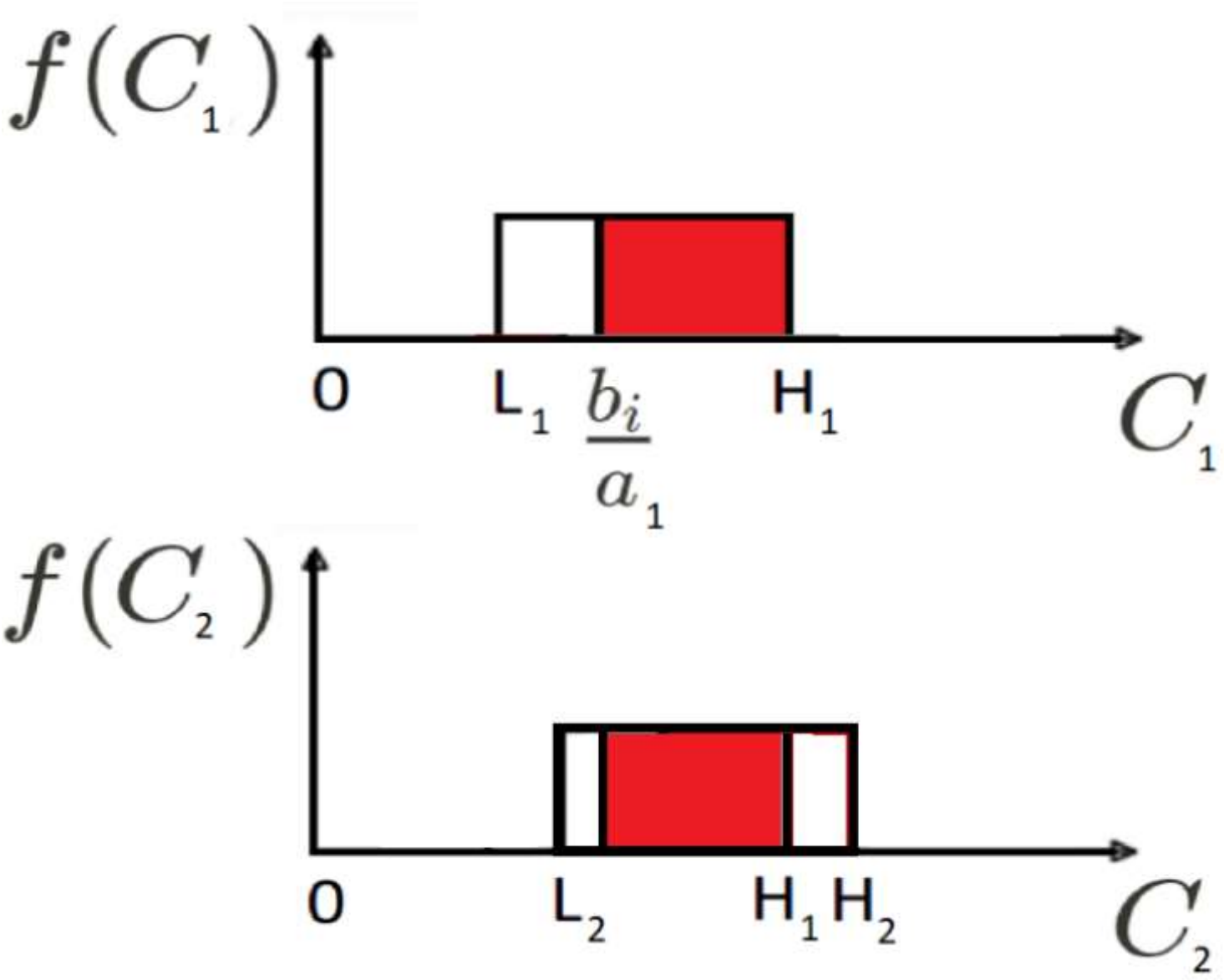

$f(C_1)$
O
L_1
$\frac{b_i}{a_1}$
H_1
C_1
$f(C_2)$
O
L_2
H_1 H_2
C_2

Fig. 3.3 Illustration of the tender when $H_1 > L_2 > L_1$ and $H_2 > H_1$

Untitled

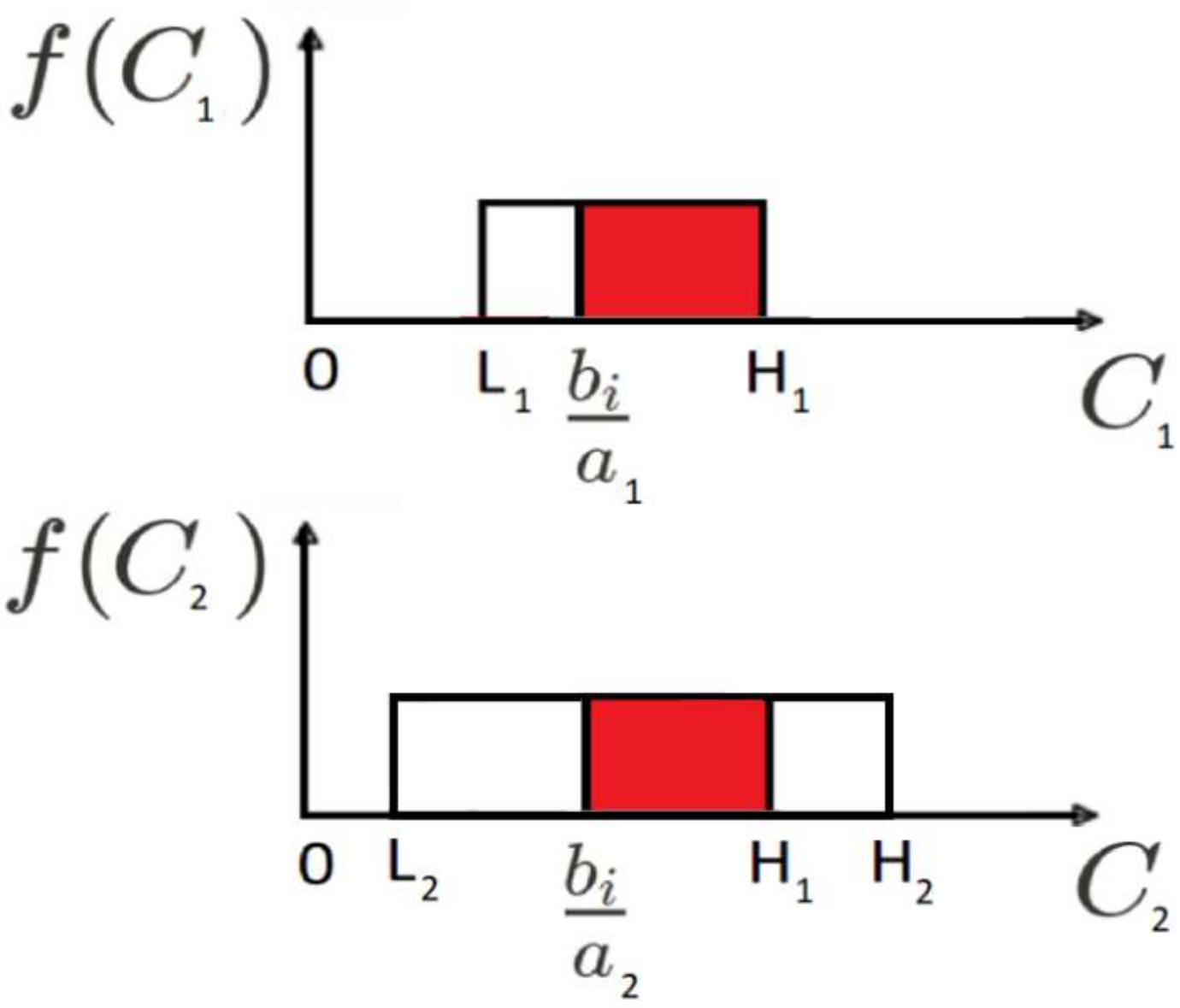

$f(C_1)$
0
L_1
$\frac{b_i}{a_1}$
H_1
C_1
$f(C_2)$
0
L_2
$\frac{b_i}{a_2}$
H_1
H_2
C_2

Fig. 3.4 Illustration of the tender when $L_2 < L_1$ and $H_2 > H_1$

When we have a tender with three sellers *i* and $j = 1, 2$. The sellers *j* have a uniform distribution of costs such as $C_j \sim \mathscr{U}(L_1, H_1)$, and $C_k \sim \mathscr{U}(L_2, H_2)$, respectively. Therefore, the expected utility function for b_i in the intersection of intervals $(L_1, H_1)$and (L_2, H_2) (we assume it is not empty) can be written in the following form:

$\mathbb{E}[U(b_i, b_j, b_k, C_i)] = \log(b_i - C_i) + \sum_{j=1}^{2}[\log(a_j H_j - b_i) - \log(a_j(H_j - L_j))]$

In order to find the optimal price for seller i we need to find the derivative of the expected utility function, set it to zero and solve it with respect to b_i.

The derivative of the expected utility

$\frac{\partial \log EU_i(b_i,b_j,S_i)}{\partial b_i} = \frac{1}{b_i - C_i} - \frac{1}{a_j H_j - b_i} - \frac{1}{a_k H_k - b_i} = 0$

```
import sympy as sy
sy.init_printing()

import numpy as np

b_i, C_i, a, H_1, H_2 = sy.symbols('b_i C_i a H_1 H_2')   #Define Symbols
sy.solve(1/(b_i-C_i) - 1/(a*H_1 - b_i) - 1/(a*H_2 - b_i), b_i,
         force=True, manual=True, set=True)
```

$\left([b_i], \left\{ \frac{C_i}{3} + \frac{H_j a_j}{3} + \frac{H_k a_k}{3} - \frac{\sqrt{C_i^2 - C_i H_j a_j - C_i H_k a_k + H_j^2 a_j^2 - H_j H_k a_j a_k + H_k^2 a_k^2}}{3} \right\} \right)$

or

$$\left([b_i], \left\{ \frac{B_j}{3} + \frac{B_k}{3} + \frac{C_i}{3} - \frac{\sqrt{B_j^2 - B_j B_k - B_j C_i + B_k^2 - B_k C_i + C_i^2}}{3} \right\} \right)$$

where $B_j = a_j H_j$ and $B_k = a_k H_k$.

In practice, the cost or price of competitors might not be distributed uniformly and probability density can be only estimated from the historical sample of prices (see Fig. 3.5) but the probability of winning is still equal to 1 - CDF(bid price):

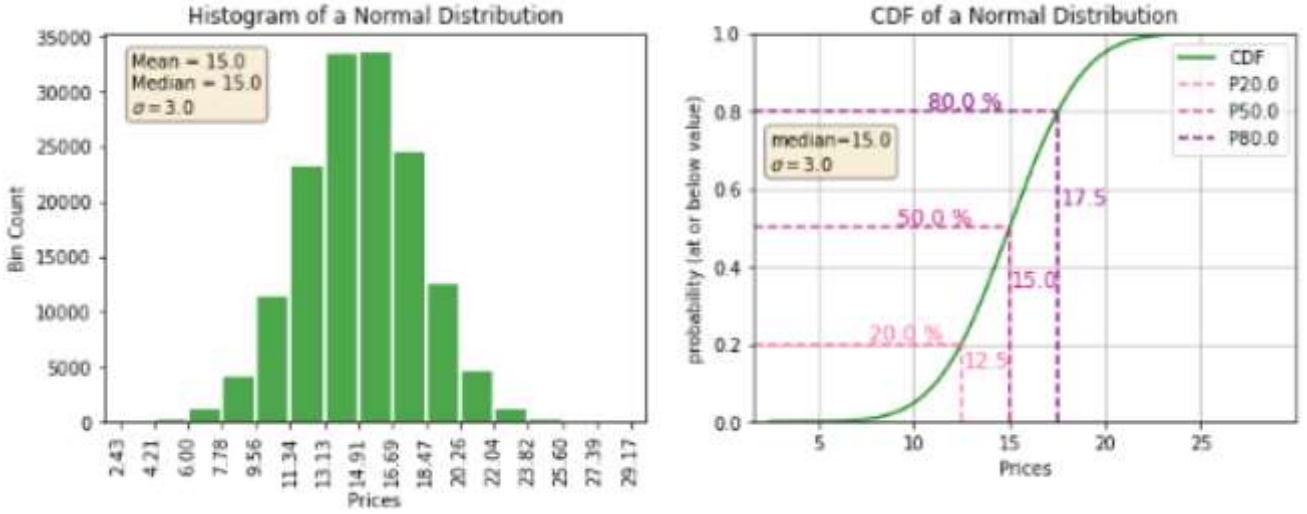


Fig. 3.5 Illustration of a historical sample of prices. We use the normal distribution as illustration only

In the above example we use normal distribution for illustration purpose only.

4. Real Example

For example, we have a real dataset that is given in Table 3 and Fig 4.1. The task is to obtain the optimal bidding price for 1 Aug 2020 using the previous 10 months (01/10/2019 - 01/07/2020).

In this example we do not assume that our competitor prices are normally distributed. It means we cannot solve the problem analytically without seeing the data but we still can find the best price using expectation maximazation approach.

Table 3 Real example for estimating the probability of winning using 10 historical prices for each competitor. first_price is recommended price for bidding and first_prob is the probability of winning using these historical prices.

Year_Month	Substitution_Group_Size_Group	Company	first_price	first_prob	first_utility	second_price	second_prob	second_utility	Medical Valley Invest	KRKA Sverige	Pfizer	Mylan	Ebb Medical	Accord Healthcare	Minimum Price
01/01/2019	112841_T18	4							7.48	7.49	7.57	7.52			7.48
01/02/2019	112841_T18	4							7.7	7.32	7.57	7.52			7.32
01/03/2019	112841_T18	5							7.31	7.56	7.57	7.52	7.15		7.15
01/04/2019	112841_T18	5							6.9	7.56	7.57	7.52	7.56		6.9
01/05/2019	112841_T18	5							7.7	6.29	7.57	7.52	7.01		6.29
01/06/2019	112841_T18	5							7	7.32	7.57	7.52	7.01		7
01/07/2019	112841_T18	5							7.7	6.98	7.57	7.52	6.15		6.15
01/08/2019	112841_T18	6							7.7	6.63	7.57	7.52	7.56	5.16	5.16
01/09/2019	112841_T18	6							6.9	7.56	7.57	7.52	6.22	5.16	5.16
01/10/2019	112841_T18	6							6.24	7.56	7.57	7.52	7.56	5.16	5.16
01/11/2019	112841_T18	6	5.15	1	5.15	6.21	1	6.21	7.71	5.57	7.57	7.52	7.04	5.16	5.16
01/12/2019	112841_T18	6	5.15	1	5.15	5.57	1	5.57	7.71	7.56	7.57	7.52	6.63	5.16	5.16
01/01/2020	112841_T18	6	5.15	1	5.15	5.57	1	5.57	7.71	7.56	7.57	7.52	4.64	5.91	4.64
01/02/2020	112841_T18	6	5.16	0.9	4.644	5.57	1	5.57	4.26	7.56	7.57	7.52	7.52	7.56	4.26
01/03/2020	112841_T18	6	4.25	1	4.25	5.57	1	5.57	7.7	6.8	7.57	7.52	7.52	5.33	5.33
01/04/2020	112841_T18	6	4.25	1	4.25	5.57	1	5.57	7.7	4.71	7.57	7.52	4.61	7.52	4.61
01/05/2020	112841_T18	6	4.25	1	4.25	5.57	0.9	5.013	7.7	3.76	7.57	7.52	7.56	7.52	3.76
01/06/2020	112841_T18	6	4.26	0.9	3.834	5.57	0.888888889	4.951111111	6.43	7.56	7.57	7.52	5.91	7.52	5.91
01/07/2020	112841_T18	6	4.26	0.9	3.834	5.57	0.888888889	4.951111111	7.4	4.54	7.57	7.52	7.56	7.52	4.54
01/08/2020	112841_T18	6	4.26	0.9	3.834	5.57	0.875	4.87375	7.51	7.56	7.57	7.52	5.77	7.52	5.77

Medical Valley Invest

6.24
7.71
7.71
7.71
4.26
7.7
7.7
7.7
6.43
7.4

P(>4.26) = 10/10 = 1

KRKA Sverige

7.56
5.57
7.56
7.56
7.56
6.8
4.71
3.76
7.56
4.54

P(>4.26) = 9/10 = 0.9

Pfizer

7.57
7.57
7.57
7.57
7.57
7.57
7.57
7.57
7.57
7.57

P(>4.26) = 1

Mylan

7.52
7.52
7.52
7.52
7.52
7.52
7.52
7.52
7.52
7.52

P(>4.26) = 1

Ebb Medical

7.56
7.04
6.63
4.64
7.52
7.52
4.61
7.56
5.91
7.56

P(>4.26) = 1

Accord Healthcare

5.16
5.16
5.16
5.91
7.56
5.33
7.52
7.52
7.52
7.52

P(>4.26) = 1

Fig 4.1 Illustration of sliding window approach and estimation of the probability of winning

The joint probability of winning for the recommended price, price = 4.26, is equal to:

$P(price\ 4.26\ is\ winning) = 1 \cdot 0.9 \cdot 1 \cdot 1 \cdot 1 \cdot 1 = 0.9$ (see Table 4).

Expected Utility = 4.26 * Joint Probability = $4.26 \cdot 0.9 = 3.834$ (see Table 4).

The recommended price was selected by finding the maximum value of the expected utility using the grid of candidate prices (see Fig 4.2).

Fig. 4.2 Illustration of optimal price calculation using Maximisation of Expected Utility.

Fig. 4.2 was obtained using output_df (see the following Python script and Fig. 4.3):

```
import numpy  as np
import pandas as pd
def recommended_price_fun(random_max_values, low_price, high_price,
                          step = 0.01, order_best_utility = 0):
    output = []
    max_utility = 0
    interval = np.arange(low_price, high_price, step)
    if len(interval) == 0:
       interval = [low_price]
    #
    for price in interval:
        prob = 1
        for i in random_max_values:
            if len(random_max_values[i]) > 0:
               num = len(random_max_values[i][random_max_values[i] > price])
               P_j = float(num) / len(random_max_values[i])
               prob = prob * P_j
        utility = price * prob
        elem = {"price": price, "probability": prob, "utility": utility}
        output.append(elem)
        #
    output_df = pd.DataFrame(output)
    output_df.sort_values(by = "utility", ascending = False, inplace = True)
    output_df = output_df.head(order_best_utility + 1)
    output_df = output_df.tail(1)
    #
    recommended_price = output_df["price"].values[0]
    recommended_prob  = output_df["probability"].values[0]
    max_utility       = output_df["utility"].values[0]
    #
```

```
    return output, recommended_price, recommended_prob, max_utility
#
random_max_values = {}
random_max_values["Medical Valley Invest"] =
                  np.array([6.24, 7.71, 7.71, 7.71, 4.26,  7.7,  7.7,  7.7, 6.43,  7.4])
random_max_values["KRKA Sverige"         ] =
                  np.array([7.56, 5.57, 7.56, 7.56, 7.56,  6.8, 4.71, 3.76, 7.56, 4.54])
random_max_values["Pfizer"               ] = np.array([7.57] * 10)
random_max_values["Mylan"                ] = np.array([7.52] * 10)
random_max_values["Ebb Medical"          ] =
                  np.array([7.56, 7.04, 6.63, 4.64, 7.52, 7.52, 4.61, 7.56, 5.91, 7.56])
random_max_values["Accord Helthcare"     ] =
                  np.array([5.16, 5.16, 5.16, 5.91, 7.56, 5.33, 7.52, 7.52, 7.52, 7.52])
#
step = 0.01
order_best_utility = 0
#
low_price  = None
high_price = None
#
for company in random_max_values:
    if (low_price is None ) or (low_price > np.min(random_max_values[company])):
        low_price = np.min(random_max_values[company])
    if (high_price is None ) or (high_price < np.max(random_max_values[company])):
        high_price = np.max(random_max_values[company])
#
low_price =  low_price - 0.01
#
output, price, prob, utility = recommended_price_fun(random_max_values,
                               low_price, high_price, step, order_best_utility)
print("Recommended price: ", price)
print("Probability of recommended price: ", prob)
print("Utility of recommended price: ", utility)
```

```
output_df = pd.DataFrame(output)
display(output_df)
```

```
Recommended price:  4.259999999999989
Probability of recommended price:  0.9
Utility of recommended price:  3.8339999999999903
```

	price	probability	utility
0	3.75	1.0	3.750
1	3.76	0.9	3.384
2	3.77	0.9	3.393
3	3.78	0.9	3.402
4	3.79	0.9	3.411
...	...	...	...
391	7.66	0.0	0.000
392	7.67	0.0	0.000
393	7.68	0.0	0.000
394	7.69	0.0	0.000
395	7.70	0.0	0.000

396 rows × 3 columns

Fig. 4.3 Example of the output of price recommendation function called recommended_price_fun() using prices for 6 competitors.

The advantages of the above function are that it does not use any probability density assumption and is very fast.

Actually, we do not need to compute the joint probability. We can first find the minimum price for every date (see “min_price” column in Table 4) and then use this price to find the probability of winning and the expected utility.

Table 4, Illustration of real dataset with minimum price for every date.

	Medical Valley Invest	KRKA Sverige	Pfizer	Mylan	Ebb Medical	Accord Helthcare	min_price
0	6.24	7.56	7.57	7.52	7.56	5.16	5.16
1	7.71	5.57	7.57	7.52	7.04	5.16	5.16
2	7.71	7.56	7.57	7.52	6.63	5.16	5.16
3	7.71	7.56	7.57	7.52	4.64	5.91	4.64
4	4.26	7.56	7.57	7.52	7.52	7.56	4.26
5	7.70	6.80	7.57	7.52	7.52	5.33	5.33
6	7.70	4.71	7.57	7.52	4.61	7.52	4.61
7	7.70	3.76	7.57	7.52	7.56	7.52	3.76
8	6.43	7.56	7.57	7.52	5.91	7.52	5.91
9	7.40	4.54	7.57	7.52	7.56	7.52	4.54

The following Python script demonstrate that we will get the same recommended price if we use only minimum price instead of prices for all competitors (see Fig. 4.4) :

```
df = pd.DataFrame(random_max_values)
df["min_price"] = np.amin(df.values, axis=1)
#
random_max_values2 = {}
random_max_values2["min_price"] = df["min_price"].values
output2, price2, prob2, utility2 = recommended_price_fun(random_max_values2,
                                        low_price, high_price, step, order_best_utility)
#
print("Recommended price: ", price2)
print("Probability of recommended price: ", prob2)
print("Utility of recommended price: ", utility2)
output_df2 = pd.DataFrame(output2)
display(output_df2)
```

```
Recommended price:  4.259999999999989
Probability of recommended price:  0.9
Utility of recommended price:  3.8339999999999903
```

	price	probability	utility
0	3.75	1.0	3.750
1	3.76	0.9	3.384
2	3.77	0.9	3.393
3	3.78	0.9	3.402
4	3.79	0.9	3.411
...	...	...	...
391	7.66	0.0	0.000
392	7.67	0.0	0.000
393	7.68	0.0	0.000
394	7.69	0.0	0.000
395	7.70	0.0	0.000

396 rows × 3 columns

Fig. 4.4 Example of the output of price recommendation function called recommended_price_fun() using min_price only.

It can be seen that Fig.4.5 and Fig. 4.6 have the same output. **This results is very important because in some datasets we have only minimum price and do not have the price of competitors.**

The probability density of minimum price can be modelled using Extreme Value distribution. We convert our proble of minimum price in such a way that the random variable can be distributed according to Generalise Extreme Value Distribution (see Fig. 4.5):

$$\mathbb{E}\{U_i(b_i, b_{j=1,..,N}, C_i)\} = (b_i - C_i)P(win|b_i)$$
$$P(win|b_i) = P(b_i < \min(b_1, b_2, ..., b_N))$$
$$P(win|b_i) = P((-1) \cdot b_i > \max(-b_1, -b_2, ..., -b_N))$$
$$P(win|b_i) = P(\max(\tilde{b}_1, \tilde{b}_2, ..., \tilde{b}_N) < \tilde{b}_i)$$

where $\tilde{b}_k = (-1)b_k$.

Based on the value of ξ, the Generalised Extreme Value (GEV) distribution is equal to one of three distributions as follows: The Fr´echet, Gumbel, and the Weibull families correspond to the cases ξ > 0, ξ = 0, and ξ < 0, respectively.

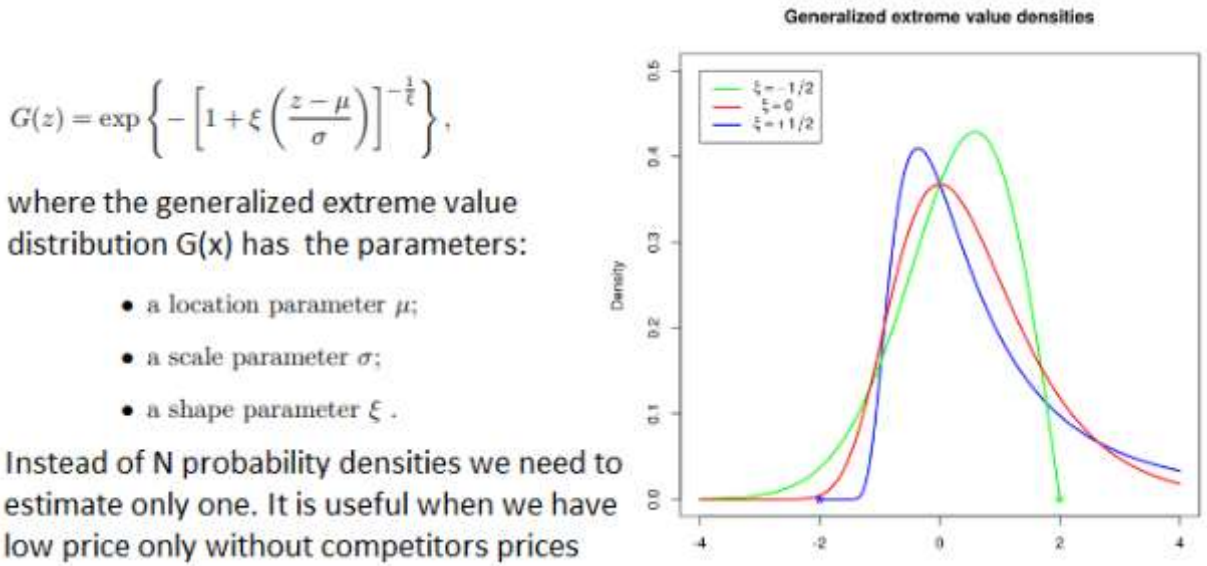


Fig. 4.5 Illustration of Generalise Extreme Value Distribution

In the following script we estimate extreme value distribution using historical "min_price" multiply by (-1):

```
from scipy import stats
import numpy  as np
import pandas as pd
def GEV_prices_selection(random_max_values, low_price, high_price, step = 0.01,
                         order_best_utility = 0):
    #
    key = list(random_max_values.keys())[0]
```

```
    #
    ste_parameters = stats.genextreme.fit((-1) * random_max_values[key])
    #
    output = []
    interval = (-1) * np.arange(low_price, high_price, step)
    #print("LOW, HIGH, STEP: ", low_price, high_price, step)
    #
    #print("interval: ", interval)
    for price in interval:
        #print("price: ", price)
        normalised_price = price # (price - y_mean) / y_std
        #
        prob = stats.genextreme.cdf(normalised_price, c =  ste_parameters[0],
                                    loc =  ste_parameters[1], scale =  ste_parameters[2])

        #print("prob: ", prob)
        #
        utility = np.abs(price) * prob
        #
        elem = {"price": np.abs(price), "probability": prob, "utility": utility}
        output.append(elem)
    #
    output_df = pd.DataFrame(output)
    output_df.sort_values(by = "utility", ascending = False, inplace = True)
    output_df = output_df.head(order_best_utility + 1)
    output_df = output_df.tail(1)
    #
    recommended_price = output_df["price"].values[0]
    recommended_prob  = output_df["probability"].values[0]
    max_utility       = output_df["utility"].values[0]
    #
    return output, recommended_price, recommended_prob, max_utility
#
EVT_random_max_values = {}
EVT_random_max_values["min_price"] = df["min_price"].values
#
```

```
low_price  = None
high_price = None
#
for company in EVT_random_max_values:
    if (low_price is None ) or (low_price > np.min(EVT_random_max_values[company])):
        low_price = np.min(EVT_random_max_values[company])
    if (high_price is None ) or (high_price < np.max(EVT_random_max_values[company])):
        high_price = np.max(EVT_random_max_values[company])
#
low_price =  low_price - 0.01
#
EVT_output, EVT_price, EVT_prob, EVT_utility = GEV_prices_selection(EVT_random_max_values,
                      low_price, high_price, step = 0.01, order_best_utility = 0)
EVT_output_df = pd.DataFrame(EVT_output)
print("Recommended price: ", EVT_price)
print("Probability of recommended price: ", EVT_prob)
print("Utility of recommended price: ", EVT_utility)
display(EVT_output_df)
```

We get a slighter different recommended price (see Fig 4.6)

```
Recommended price:  3.9699999999999953
Probability of recommended price:  0.9353616449092997
Utility of recommended price:  3.7133857302899154
```

	price	probability	utility
0	3.75	0.973970	3.652388
1	3.76	0.972709	3.657387
2	3.77	0.971406	3.662202
3	3.78	0.970060	3.666828
4	3.79	0.968670	3.671261
...	...	...	...
211	5.86	0.038633	0.226387
212	5.87	0.037083	0.217676
213	5.88	0.035582	0.209222
214	5.89	0.034129	0.201021
215	5.90	0.032724	0.193069

216 rows × 3 columns

Fig. 4.6 Output of GEV_prices_selection function that estimates Extreme Value distribution in order to make price recommendation

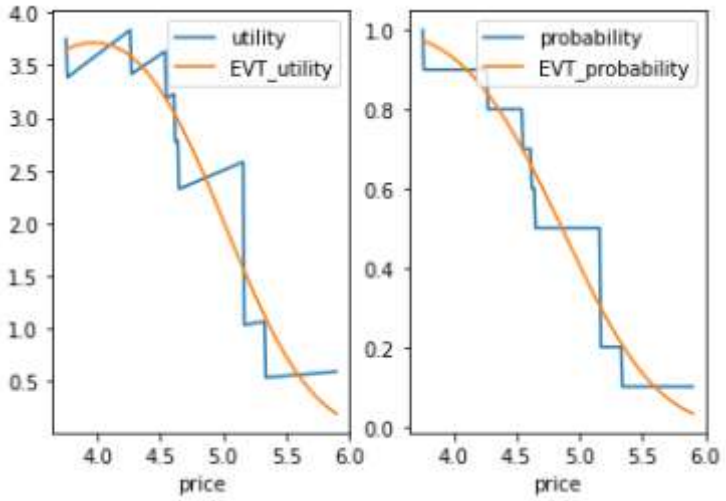


Fig. 4.7 Comparison of two methods of price recommendation using “min_price" that corresponds to results illustrated by Fig. 4.4 and Fig. 4.6. Blue line correspond to method when we compute the probability by counting how many historical prices are above the candidate for recommendation price. Red one is based on Extreme Value Distribution estimation- we first find parameters of extreme value distribution for negative price and then use cumulative distribution function to compute the probability of winning.

The above two plots (see Fig 4.7) was obtained using the following Python script:

```
import matplotlib.pyplot as plt
m = output_df2.merge(EVT_output_df, on = "price")
fig, axes = plt.subplots(1, 2)
m.plot("price", ["utility", "EVT_utility"], ax=axes[0])
m.plot("price", ["probability", "EVT_probability"], ax=axes[1])
```

Evaluation performance of the algorithm is based on observed Expected Utility (see Table 5). first_price is recommended price and "Minimum price" is historical observation of minimum price given tender and date. win is TRUE if first_price < "Minimum price" and FALSE otherwise.

Table 5 Evaluation of First Price Recommendation Performance

Year_Month	Substitution_Group_Size_Group	Company	first_price	win	Minimum Price	Utility
01/01/2019	112841_T18	4			7.48	
01/02/2019	112841_T18	4			7.32	
01/03/2019	112841_T18	5			7.15	
01/04/2019	112841_T18	5			6.9	
01/05/2019	112841_T18	5			6.29	
01/06/2019	112841_T18	5			7	
01/07/2019	112841_T18	5			6.15	
01/08/2019	112841_T18	6			5.16	
01/09/2019	112841_T18	6			5.16	
01/10/2019	112841_T18	6			**5.16**	
01/11/2019	112841_T18	6	5.15	TRUE	**5.16**	5.15
01/12/2019	112841_T18	6	5.15	TRUE	**5.16**	5.15
01/01/2020	112841_T18	6	5.15	FALSE	**4.64**	0
01/02/2020	112841_T18	6	5.16	FALSE	**4.26**	0
01/03/2020	112841_T18	6	4.25	TRUE	**5.33**	4.25
01/04/2020	112841_T18	6	4.25	TRUE	**4.61**	4.25
01/05/2020	112841_T18	6	4.25	FALSE	**3.76**	0
01/06/2020	112841_T18	6	4.26	TRUE	**5.91**	4.26
01/07/2020	112841_T18	6	4.26	TRUE	**4.54**	4.26
01/08/2020	112841_T18	6	4.26	TRUE	5.77	4.26

Probability of win = (1 + 1 + 0 + 0 + 1 + 1 + 0 + 1 + 1 + 1) / 10 = 0.7 where “TRUE” is equal to 1 and “FALSE” is equal to 0 (see column win).

We test our both algorithms for 9 Molecules where we use sliding window of size 10 for probability and expected utility estimaion and the following 11th sample for forecast or price recommendation (see Table 6.1 and 6.2). The total sample size given molecule is 10 samples (prices) plus the corresponding value of the column “n” (see Table 6.1 and 6.2). Therefore, the sample size is not sufficient to fit a complex model such as LSTM. That is why, the proposed algorithms are very useful. It seems the simplest algorithm perform the best - it gets the total observed expected utility (TOEU) for all nine molecules equal to 2130.60 that is larger than using Extreme Value distribution, TOEU = 862.34.

Table 6.1 Illustration of the price recommendation algorithm performance with simple probability calculation (see Fig. 4.4 and associated Python code). The total observed utility for nine molecules is equal to `2130.60`.

	Molecule	n	obtained utility	probability of win	previous utility	Uplift	quantile	order_best_utility
0	Aciclovir	26	46.404552	0.384615	43.510537	2.894015	0.95	0
1	Apixaban	11	139.721637	0.818182	1.781134	137.940502	0.95	0
2	Azithromycin	16	192.998870	0.687500	8.820963	184.177906	0.95	0
3	Methylprednisolone succinate	6	200.348890	0.833333	27.681874	172.667016	0.95	0
4	Octreotide	5	1125.635092	0.200000	16.143711	1109.491381	0.95	0
5	Oxaliplatin	20	45.031440	0.200000	37.288074	7.743365	0.95	0
6	Piperacillin+ Tazobactam	9	9.658323	0.444444	8.654131	1.004192	0.95	0
7	Sulfasalazine	8	3.800072	0.125000	0.791534	3.008539	0.95	0
8	Tigecycline	11	523.300777	0.818182	11.630117	511.670660	0.95	0

Table 6.2 Illustration of the price recommendation algorithm performance using Extreme Value Distribution Estimation (see Fig. 4.6 and associated Python code). The total observed utility for nine molecules is equal to `862.34`.

	Molecule	n	obtained utility	probability of win	previous utility	Uplift	quantile	order_best_utility
0	Aciclovir	26	41.860495	0.423077	29.613311	12.247184	1.0	1
1	Apixaban	11	38.871637	0.818182	1.781134	37.090502	1.0	1
2	Azithromycin	16	158.268870	0.687500	8.820963	149.447906	1.0	1
3	Methylprednisolone succinate	6	172.089112	0.666667	73.921085	98.168027	1.0	1
4	Octreotide	5	137.505092	0.200000	16.143711	121.361381	1.0	1
5	Oxaliplatin	20	34.651440	0.200000	24.800405	9.851035	1.0	1
6	Piperacillin+ Tazobactam	9	6.349162	0.222222	13.766646	-7.417485	1.0	1
7	Sulfasalazine	8	0.940072	0.125000	0.791534	0.148539	1.0	1
8	Tigecycline	11	453.070777	0.818182	11.630117	441.440660	1.0	1

Predictive distribution work (see other documents). Instead of historical observation, we can use the Bayesian timeseries approch to get predictive distribution of price forecast and then use this distribution to maximise the utility and get the recommended price. In another document we use the dataset with much larger sample size. Fscebook Prophet, Gaussian Processes, Bayesian LSTM and Transformers.