ONLINE OPTIMIZATION OF AN ENERGY STORAGE SYSTEM FOR ENERGY ARBITRAGE WITH RESERVE COMMITMENT

ABDESSAMAD EL KABID, AHMED MANSOUR BOURASSINE, LUCAS LEVY

1. Introduction

1.1. **Context.**

1.1.1. *Industrial motivation.* Battery energy storage system is an energy storage technology that uses a group a batteries connected to the electric grid to store energy. Given the possibility that an energy supply can experience fluctuations due to weather, blackouts or other unexpected reasons, battery systems are vital for businesses and private customers to achieve a continual power flow.

The electricity market is divided in sub-markets, and we focus on the intraday market, on which electricity can be bought and sold continuously for delivery on the same day. It enables the continuous trading of electricity throughout the day and allows its participants to trade electricity closer to real time. It provides flexibility and responsiveness to changes in supply and demand that occur within the trading day.

In this setting, an actor who possess batteries makes some reserves commitments to a transmission system operator the day before, contracting to provide or to absorb an unknown quantity of energy at each hour. This quantity is bounded by the commitments they made. In addition, the owner of the battery can decide at each time step to buy or sell more energy.

Optimal operation strategies, considering market rules and technical constraints, are crucial for maximizing economic profits or minimizing participation costs. Being one of those strategies, energy arbitrage for such an actor will consist of buying energy when prices are low and selling when prices are high. This task is actually non-trivial because of the uncertainties regarding the future prices and demand from the transmission system operator.

1.1.2. *Scientific motivation.* This problem has been previously studied by Sossou Edou [1] as part of a research initiative from the PGMO (Gaspard Monge Program for Optimization, operations research and their interactions with data science). In this paper, the author follows a stochastic approach, where the prices and demand are random variables following some distribution. The goal is then to minimize the expected value of the operating cost.

Here, another approach is proposed, namely, *competitive analysis.* No assumptions are made on price and demand distribution. Instead, we reason on all the instances of values prices and demands can take, and try to minimize the operating cost in the worst case. A more detailed description is given in the following section.

1.2. **Problem formulation and chosen approach.** We aim at solving the following optimization problem

$$(\mathrm{P})\qquad \begin{array}{ll} \text{minimize} & \displaystyle\sum_{t=1}^{T} P_t(x_t+R_t) \\ \text{s.t.} & \\ & s_t = s_{t-1} + x_t + R_t \hfill \forall t \in [T] \\ & s_t \in [0,S] \hfill \forall t \in [T] \\ & x_t + R_t \in [-C^{\max}, +C^{\max}] \hfill \forall t \in [T] \\ & x_t \text{ depends only on } R_1,\dots,R_{t-1},P_1,\dots,P_t \quad \forall t \in [T] \end{array}$$

The R_t and P_t are unknown beforehand but are respectively in the intervals $[-R^{\max}, +R^{\max}]$ and $[p^{\min}, p^{\max}]$ with $p^{\min} \geqslant 0$, which are known. For each $t \in [T]$, the informations are revealed and the decision are taken in the following order: first, the price P_t is revealed, then the agent has to choose the quantity x_t they want to sell or but, and finally the imposed demand R_t is revealed.

A competitive analysis approach is followed. In this setting, the parameters $s_0, S, C^{\max}$, $R^{\max}, p^{\min}$ and $p^{\max}$ are known beforehand and the instances $I = (R_{t\in[T]}, P_{t\in[T]})$ are the the unknown values that can be taken. Given an algorithm ALG and an instance I, we denote $ALG(I)$ the value obtained by the online algorithm in the instance I. We also denote $OPT(I)$ the optimal value that can be obtained, in the offline version of the problem, where I is known.

Then, for a minimization problem, an algorithm ALG is *c-competitive* if for all instances I, we have $ALG(I) \leqslant c\,OPT(I)$. The *competitivity ratio* of the algorithm is the infimum of c such that it is c-competitive.

1.3. **Contributions.** When the prices are constant, we totally answer the question

Theorem 1. *With constant positive price and unknown demand R. If $\min(C^{\max}, S) \geq 2R^{\max}$, then the algorithm choosing at time t*

$$x_t = -\min(C^{\max}, s_{t-1}) + R^{\max}$$

has a competitive ratio of

$$c = \frac{\max(2R^{\max} - s_0, -T(C^{\max} - 2R^{\max}))}{\max(-s_0, -TC^{\max})}$$

This is the best competitive ratio that can be achieved.

When there is no reserve commitment ($R^{\max} = 0$), we also totally answer the question

In the general case, we only have a dynamic programming formulation for the algorithm earning the best minimal guaranteed value, regardless of the competitive ratio.

Theorem 2. *Define by induction the functions $W_1, \dots, W_{T+1}$:*

$$W_t(s) = \sup_{p\in[p^{\min},p^{\max}]} \inf_{|x|\leqslant C^{\max}-R^{\max}} \sup_{|r|\leqslant R^{\max}} \big(p(x+r) + W_{t+1}(s+x+r)\big) \quad \forall t,\ \forall s \in [0,S]$$

$$W_t(s \notin [0,S]) = +\infty \quad \forall t$$

$$W_{T+1}(s \in [0,S]) = 0$$

If $\min(C^{\max}, S) \geq 2R^{\max}$, *then the algorithm choosing at time* t

$$x_t \in \underset{|x| \leq C^{\max}-R^{\max}}{\arg\min} \sup_{|r| \leq R^{\max}} \big(P_t(x+r) + W_{t+1}(s_{t-1}+x+r)\big)$$

has the best minimal guaranteed value, which is $W_1(s_0)$.

Theorem 3. *Let* $\theta := p^{max}/p^{min}$ *(or* U/L *in the case we only know an interval* $[L,U]$ *for the prices). In the case where no demand is observed, suppose that* $S_0 + TC^{max} - S \leqslant 0$ *and that* $S_0 \leqslant C^{max}$.
Algorithm 1 achieves an optimal competitive ratio of $(1 + \ln\theta)$.
The proof is the section 4.

2. Acceptable algorithm

An algorithm is *acceptable* if for every instance I, the inequalities $s_t \in [0, S]$ and $|x_t + R_t| \leq C^{\max}$ are checked at every time t.

Proposition 4. *An algorithm is acceptable if and only if for every time* t *we have*

$$x_t \in [\max(-C^{\max} + R^{\max}, -s_{t-1} + R^{\max}), \min(C^{\max} - R^{\max}, S - s_{t-1} - R^{\max})] \, .$$

Proof. Let's first suppose that x_t is in the interval for every t. Then, for every instance I, we have

First, $x_t + R_t \geq -C^{\max} + R^{\max} - R^{\max}$ and $x_t + R_t \leq C^{\max} - R^{\max} + R^{\max}$ so $|x_t + R_t| \leq C^{\max}$. Also, $s_t \geq s_{t-1} + (-s_{t-1} + R^{\max}) - R^{\max} = 0$ and $s_t \leq s_{t-1} + (S - s_{t-1} - R^{\max}) - R^{\max} = S$ so $s_t \in [0, S]$.

Therefore the algorithm is acceptable.

If one inequality is not checked for some x_t, we can easily build an instance $R_t = \pm R^{\max}$ such that either $s \notin [0, S]$ or $|x_t + R_t| > C^{\max}$, that would make the algorithm not acceptable. □

Remark 5. *From the definition, it follows that an unacceptable algorithm has a competitive ratio of* $+\infty$.

Proof. Let ALG be an unacceptable algorithm. There exists an instance I of the problem such that x is not an acceptable solution. So the value of the maximization problem for the instance I is $-\infty$. So there is no $c \in \mathbb{R}$ such that $c\ ALG(I) \geq OPT(I)$. ALG is not c-competitive. Therefore the competitive ratio of ALG is $\inf \varnothing = +\infty$. □

Proposition 6. *If* $\min(C^{\max}, S) \geqslant 2R^{\max}$ *, then there exists an acceptable algorithm.*

Proof. Let's take the algorithm

$$x_t = -\min(C^{\max}, s_{t-1}) + R^{\max}, \quad \forall t \tag{1}$$

Let's show by induction that for all instance I and all time t, $s_t \in [0, S]$. Let's t be such that $s_{t-1} \in [0, S]$.

- If $C^{\max} \leqslant s_{t-1}$, then $x_t = -C^{\max} + R^{\max}$ and we have

$$s_t = s_{t-1} + x_t + R_t \geqslant C^{\max} + (-C^{\max} + R^{\max}) - R^{\max} = 0$$

$$s_t = s_{t-1} + x_t + R_t \leqslant S + (-C^{\max} + R^{\max}) + R^{\max} \leqslant S$$

- If $C^{\max} \geqslant s_{t-1}$, then $x_t = -s_{t-1} + R^{\max}$ and we have

$$s_t \geqslant s_{t-1} + (-s_{t-1} + R^{\max}) - R^{\max} = 0$$
$$s_t \leqslant s_{t-1} + (-s_{t-1} + R^{\max}) + R^{\max} = 2R^{\max} \leqslant S$$

So for all t, we have $s_t \in [0, S]$.

Moreover, for all t and instance I,

- If $C^{\max} \leqslant s_{t-1}$, then $x_t = -C^{\max} + R^{\max} \in [\pm(C^{\max} - R^{\max})]$.
- Else, $x_t = -s_{t-1} + R^{\max} \in [-C^{\max} + R^{\max}, R^{\max}]$.

So $|x_t + R_t| \leqslant C^{\max}$. In conclusion, the algorithm is acceptable. □

3. Constant price, R unknown

With constant positive prices, the problem P becomes

$$\text{(CP)} \quad \begin{aligned} \text{minimize} \quad & \sum_{t=1}^{T} (x_t + R_t) \\ \text{s.t.} \quad & \\ & s_t = s_{t-1} + x_t + R_t & \forall t \in [T] \\ & s_t \in [0, S] & \forall t \in [T] \\ & x_t + R_t \in [-C^{\max}, +C^{\max}] & \forall t \in [T] \\ & x_t \text{ depends only on } R_1, \ldots, R_{t-1} & \forall t \in [T] \end{aligned}$$

Theorem 7. *The algorithm described in 1 choosing*

$$x_t = \max(-C^{\max}, -s_{t-1}) + R^{\max}, \quad \forall t$$

is acceptable and has a competitive ratio of

$$c = \frac{\max(2R^{\max} - s_0, -T(C^{\max} - 2R^{\max}))}{\max(-s_0, -TC^{\max})}$$

This is the best competitive ratio that can be achieved.

Lemma 8. *The offline problem has an optimal value of*

$$OPT = \max(-TC^{\max}, -s_0)$$

regardless of the values taken by R.

Proof. By setting $X_t = x_t + R_t$, the offline problem can be written as

$$\text{(CP-offline)} \quad \begin{aligned} \text{minimize} \quad & \sum_{t=1}^{T} X_t \\ \text{s.t.} \quad & \\ & s_t = s_{t-1} + X_t & \forall t \in [T] \\ & s_t \in [0, S] & \forall t \in [T] \\ & X_t \in [-C^{\max}, +C^{\max}] & \forall t \in [T] \end{aligned}$$

The problem then no longer depends on the values taken by the R_t. The optimum is attained by chosing $X_t = \max(-C^{\max}, -s_{t-1})$ at time t.
We then have $s_t = \max(s_{t-1} - C^{\max}, 0)$ and finally

$$OPT = s_T - s_0 = \max(-TC^{\max}, -s_0) \quad □$$

Lemma 9. *The algorithm described in 1 can always do better than*

$$ALG(R) \leqslant \max(2R^{\max} - s_0, -T(C^{\max} - 2R^{\max}))$$

regardless of the values taken by R.

Proof. Show by induction that for all t, $s_t \leqslant \max(2R^{\max}, s_0 - t(C^{\max} - 2R^{\max}))$

- Base case: $s_0 \leqslant \max(2R^{\max}, s_0)$
- Induction case: Let t be such that $s_{t-1} \leqslant \min(2R^{\max}, s_0 - (t-1)(C^{\max} - 2R^{\max}))$
  - If $s_{t-1} \leqslant C^{\max}$, then $x_t = -s_{t-1} + R^{\max}$, and $s_t = s_{t-1} + x_t + R_t \leqslant s_{t-1} + (-s_{t-1} + R^{\max}) + R^{\max} = 2R^{\max}$.
  - Else, $s_{t-1} \geqslant C^{\max} \geqslant 2R^{max}$ so $s_{t-1} \leqslant s_0 - (t-1)(C^{\max} - 2R^{\max})$ and $x_t = -C^{\max} + R^{\max}$. Then, $s_t \leqslant s_0 - (t-1)(C^{\max} - 2R^{\max}) - C^{\max} + R^{\max} + R^{\max} = s_0 - t(C^{\max} - 2R^{\max})$.

  So $s_t \leqslant \max(2R^{\max}, s_0 - t(C^{\max} - 2R^{\max}))$

Therefore, $ALG(R) = s_T - s_0 \leqslant \max(2R^{\max} - s_0, -T(C^{\max} - 2R^{\max}))$ □

Lemma 10. *For every acceptable algorithm ALG', the instance $R^* = (R^{\max})_{1 \leq t \leq T}$ satisfies*

$$ALG'(R^*) \geqslant \max(2R^{\max} - s_0, -T(C^{\max} - 2R^{\max}))$$

Proof. First prove that $ALG'(R^*) \geqslant 2R^{\max} - s_0$.
ALG' is acceptable, so $x_T \geqslant -s_{T-1} + R^{\max}$. So $s_T = s_{T-1} + x_T + R^*_t \geqslant s_{T-1} + (-s_{t-1} + R^{\max}) + R^{\max} = 2R^{\max}$, and $ALG'(R^*) = s_T - s_0 \geqslant 2R^{\max} - s_0$.

Now prove that $ALG'(R^*) \geqslant -T(C^{\max} - 2R^{\max})$. ALG' is acceptable, so for all t, we have $x_t \geqslant -C^{\max} + R^{\max}$. So $s_t \geqslant s_{t-1} + (-C^{\max} + R^{\max}) + R^*_t = s_{t-1} - C^{\max} + 2R^{\max}$. By direct induction, we have $ALG'(R^*) = s_T - s_0 \geqslant -T(C^{\max} - 2R^{\max})$ □

Once we have these three lemmas, the demonstration of theorem 7 follows quite directly.

Proof of Theorem 7. The acceptability of the algorithm is proven in 1. From lemmas 8 and 9, we have that for every instance R, $ALG(R) \leqslant c\, OPT(R)$ with

$$c = \frac{\max(2R^{\max} - s_0, -T(C^{\max} - 2R^{\max}))}{\max(-s_0, -TC^{\max})}$$

By definition, it means that the algorithm is c-competitive.

From Lemma 10, there does not exist a constant $c' < c$ such that any acceptable algorithm is c'-competitive.

Therefore the competitivity ratio of the algorithm is c and there is no constant $c' < c$ and other acceptable algorithm ALG' such that ALG' is c'-competitive. □

4. No demand, unknown prices

If there is no demand, the problem is equivalent to :

$$
\text{(SP)}\qquad
\begin{aligned}
&\max \sum_{t=1}^{T} P_t y_t \\
&\text{s.t. } \forall t,\ S_0 + tC^{max} - S \leqslant \sum_{1}^{t} y_t \leqslant S_0 + tC^{max} \\
&(s_t = -\sum_{1}^{t} y_t + tC^{max} \in [S_0, S]) \\
&0 \leqslant y_t \leq 2C^{max} \\
&y_t (= C^{max} - x_t) \text{ depends only on the past} \quad \forall t \in [T]
\end{aligned}
$$

If we assume that $S_0 + TC^{max} - S \leqslant 0$ **and that** $S_0 \leqslant C^{max}$ where T is the horizon, the problem becomes :

$$
\text{(SP')}\qquad
\begin{aligned}
\text{maximize} \quad & \sum_{t=1}^{T} P_t y_t \\
\text{subject to} \quad & \sum_{i=1}^{t} y_i \leqslant S_0 + tC^{max} = b(t), \quad \forall t \\
& 0 \leqslant y_t, \quad \forall t \\
& y_t \text{ depends only on the past}, \quad \forall t \in [T]
\end{aligned}
$$

That is because
$\forall t,\ \sum_1^t y_t \leqslant S_0 + tC^{max} \implies \forall t,\ y_t \leqslant \min(S_0 + tC^{max}) = S_0 + C^{max} \leqslant 2C^{max}$

Decoupling Consider the folllowing problem (inspired by the work of [4]):

$$
\text{(DC)}\qquad
\begin{aligned}
\text{maximize} \quad & \sum_{j=1}^{n} \sum_{i=j}^{n} p(i) y(j,i) \\
\text{subject to} \quad & \sum_{i=j}^{n} y(j,i) \leqslant b(j) - b(j-1), \quad j = 1, 2, ..., n \\
\text{variables} \quad & y(j,i) \geqslant 0, \quad i \geqslant j.
\end{aligned}
$$

Lemma 11. *DC is equivalent to SP'.*
The proof of the general case is shown in Extensions.

The following lemma (Corollary 2.3 of [4]) will be useful:

Lemma 12. *Assume $b(j)$ is increasing as j increases. Let $y^*(j,i), i \geqslant j$, be an optimal solution for Problem DC. Then,*

$$y^*(i) = \sum_{j=1}^{i} y^*(j,i), \quad i = 1, 2, ..., n,$$

is an optimal solution for Problem SP'. In addition, the optimal value of Problem SP' is equal to that of DC.

The problem DC can be decoupled by solving multiple subproblems :

$$\text{(SubPj)} \qquad \begin{aligned} \text{maximize} \quad & \sum_{i=j}^{n} p(i)x(j,i) \\ \text{subject to} \quad & \sum_{i=j}^{n} x(j,i) \leqslant b(j) - b(j-1) \\ \text{variables} \quad & x(j,i) \geqslant 0, \quad i \geqslant j. \end{aligned}$$

We can then solve each subproblem a generalized one way trading problem (GOT) [3]. We use the online threshold Algorithm 1 to do that.

Algorithm 1 Online Threshold-Based Algorithm with Threshold Function ϕ (OTA_ϕ) for DC

```
**Input:** Threshold function $\phi := \{\phi(\cdot)\}$, and initial knapsack utilization $w^{(1)} = 0$;
**while** item $i$ arrives and verify the constraints **do**
    observe $P_i$ and constraints;
    For each $j \leqslant i$ (do one step in the solving process of SubPj)
        $$y(j,i) = \begin{cases} \phi_j^{-1}(P_i) - w_j^{(i)} & P_i \geqslant \phi_j(w_j^{(i)}) \\ 0 & P_i < \phi_j(w_j^{(i)}) \end{cases}$$
    update the utilization $w_j^{(i+1)} = w_j^{(i)} + y(j,i),.$
    $y_i = \sum_{j=1}^{i} y(j,i)$
    $x_i = C^{max} - y_i$
```

Each subproblem is solved in an optimal way as shown by the following theorem (Thm3.6 of [3]):

Theorem 13. *Suppose that $S_0 + TC^{max} - S \leqslant 0$ and that $S_0 \leqslant C^{max}$. When the threshold function of OTA_ϕ for SubPj is*

$$\phi_j^*(w) = \begin{cases} L & w \in [0, \beta^*) \\ Le^{(1+\ln\theta)w/b-1} & \text{if } w \in [\beta^*, b] \\ \infty & \text{if } w > b \end{cases}, \tag{2}$$

where $\beta^* = \frac{b}{\alpha_{\phi^*}}$ is the utilization threshold, where $b = b(j) - b(j-1)$, the competitive ratio of OTA_{ϕ^*} is $\alpha_{\phi^*} = 1 + \ln\theta$, which is the optimal competitive ratio of SubPj (an instance of GOT).

This competitve ratio is the best for OTAs. (The construction of the threshold function is based on a primal dual analysis, consult the work of [3]).

We use $d(j,i), i \geqslant j$, to denote the sold amount of the online algorithm for subproblem j at time slot i.

By the proposed decoupling strategy, the sold amount at time slot j is set to $\sum_{i=1}^{j} d(j,i)$. The profit earned by the online algorithm is

$$\sum_{i=1}^{n} p(i) \sum_{j=1}^{i} d(j,i) = \sum_{j=1}^{n} \sum_{i=j}^{n} p(i) d(j,i). \tag{3}$$

One can achieve a competitive ratio of $\ln(\theta)+1$ for each subproblem, as shown in theorem 13.

Then, the competitive ratio of the online algorithm satisfies

$$\begin{aligned} \text{cr} &= \frac{\sum_{j=1}^{n} OPTj}{\sum_{j=1}^{n} \sum_{i=j}^{n} p(i) d(j,i)} \\ &\leqslant \frac{\sum_{j=1}^{n} OPTj}{\sum_{j=1}^{n} \frac{OPTj}{1+\ln\theta}} \\ &= 1+\ln\theta \end{aligned}$$

That means that the online threshold algorithm is $(\ln(\theta)+1)$-competitive, achieving the optimal performance by Lemma 12. Thus proving the theorem 3.

Extensions.
If $S_0 + iC^{max} - S \leqslant 0 \quad \forall i$ and $S_0 \leqslant C^{max}$ aren't necessarily verified, we can do the same decoupling strategy and if $a(j) - a(j-1) \leqslant b(j) - b(j-1)$ for every j then our problem is *equivalent* to the following (the knapsack's capacity is bounded from both sides)

$$\text{(DFull)} \qquad \begin{aligned} \text{maximize} \quad & \sum_{j=1}^{n} \sum_{i=j}^{n} p(i) y(j,i) \\ \text{subject to} \quad & a(j) - a(j-1) \leqslant \sum_{i=j}^{n} y(j,i) \leqslant b(j) - b(j-1), \quad j = 1,2,...,n \\ \text{variables} \quad & y(j,i) \geqslant 0, \quad i \geqslant j. \end{aligned}$$

where in our case $a(j) = S_0 + jC^{max} - S$

Proof of equivalence. First, by rearranging the items, we rewrite the objective of the Problem DFull as

$$\sum_{i=1}^{n} p(i) \sum_{j=1}^{i} y(j,i).$$

Second, we show that by having a feasible solution to Problem DFull, we can construct a feasible solution to Problem SP. Assume that $x(j,i) \geqslant 0,\ i \geqslant j$ is a feasible solution. Then, for $l = 1, 2, \ldots, n$, we have

$$a(l) \leqslant \sum_{l=1}^{l} \sum_{i=1}^{l} p(i)x(j,i) \leqslant b(l).$$

Let $x(i) = \sum_{j=1}^{i} x(j,i)$. We have

$$a(l) \leqslant \sum_{l=1}^{l} p(i)x(i) \leqslant b(l).$$

That is, for each feasible solution $x(j,i) \geqslant 0,\ i \geqslant j$, we can find a feasible solution $x(i) = \sum_{j=1}^{i} x(j,i)$ for Problem SP, and the value under $x(i)$ is equal to that of Problem DFull under $x(j,i)$.

Third, we show that for a feasible solution $x(i)$ to Problem SP, we can construct a feasible solution to Problem DFull, i.e., $x(j,i) \geqslant 0,\ i \geqslant j$. Let $x(1,i) = x(i)$ for $i = 1, 2, \ldots, n$, and $x(j,i) = 0$ for $j > 1$. Obviously, $\sum_{j=1}^{i} x(j,i) = x(i)$ for $i = 1, 2, \ldots, n$. In each step, when we increase/decrease $x(i)$ by δ, we must correspondingly decrease/increase $x(j,i)$ (for some $j \neq i$) by δ. In this way, we can guarantee that $x(i) = \sum_{j=1}^{i} x(j,i)$ all the time. Then, if there is a j such that $\sum_{i=j}^{n} p(i)x(j,i) > b(j) - b(j-1)$, we have that there must be some i such that $\sum_{n=i}^{l} p(i)x(l,i) < b(l) - b(l-1)$. Otherwise, we have

$$b(n) = \sum_{l=1}^{n} b(l) - b(l-1) < \sum_{l=1}^{n} p(i)x(l,i) \leqslant \sum_{i=1}^{n} p(i) \sum_{l=1}^{i} x(l,i) = \sum_{i=1}^{n} p(i)x(i),$$

contradicting the assumption for $x(i)$. In this way, we can always decrease $x(j,i)$ (and increase $x(l,i)$) such that

$$\sum_{i=1}^{n} p(i)x(j,i) \leqslant b(j) - b(j-1) \quad \text{for all } j.$$

The exact same logic applies to the left bound, which insures that

$$a(j) - a(j-1) \leqslant \sum_{i=1}^{n} p(i)x(j,i) \quad \text{for all } j.$$

That is, we can find $x(j,i)$ which satisfies

$$x(i) = \sum_{j=1}^{i} x(j,i)$$

and meanwhile the constraints in Problem DFull. Obviously, the value is also equal.

Conjecture We conjecture that DFull can be solved with the same approach with a competitive ratio under $2(1 + \ln\theta)$.

Paths of progress

- **Improving the bounds' estimates at each step if the prices follow a structure**

  What if we have different estimations of the bounds for prices for the different subproblems (meaning for each j we have bounds L_j and U_j for the prices from j to the horizon). Let $\theta_j = \frac{U_j}{L_j}$

  Then:

$$\begin{aligned} \text{cr} &= \frac{\sum_{j=1}^{n} OPTj}{\sum_{j=1}^{n}\sum_{i=j}^{n} p(i)d(j,i)} \\ &= \frac{\sum_{j=1}^{n} OPTj}{\sum_{j=1}^{n} \frac{OPTj}{1+\ln\theta_j}} \end{aligned}$$

  Thus:

$$\frac{1}{n}\sum_{j=1}^{n}(1+\ln\theta_j) \leqslant \text{cr} \leqslant \frac{n}{\sum_{j=1}^{n}\frac{1}{(1+\ln\theta_j)}}$$

  Using Chebyshev's sum inequality two times.

- **Using an Online Algorithm for Fractional Knapsack in the Random Order Model to solve the subproblems**

  Jilberti et al. in [2] show a competitve ratio of 4.39 in the random order model, using a deterministic algorithm. Therefore we can use the algorithm they designed for OFKP to solve each subproblem from the decoupling which will give a competitve ratio bounded by 4.39 by the following remark .

  Remark 14. *In the general case of subproblems:*

$$\begin{aligned} cr &= \frac{\sum_{j=1}^{n} OPTj}{\sum_{j=1}^{n}\sum_{i=j}^{n} p(i)d(j,i)} \\ &\leqslant \max_{j=1,2,\ldots,n} \frac{OPTj}{\sum_{i=j}^{n} p(i)d(j,i)} \\ &\leqslant \max_{j=1,2,\ldots,n} cr_j, \end{aligned}$$

  where cr_j is the competitive ratio for the j-th subproblem.

5. Unknown Rt , Known Prices

Our goal is to decide the optimal storage levels s_t at each time step to maximize the total gain G, considering:

- Gains from buying low and selling high, influenced by fluctuating prices.
- Random perturbations R_t, which alter the intended storage levels.

The algorithm employs dynamic programming to determine the optimal storage actions over time, accounting for randomness and system constraints.

At each time step "t", the algorithm evaluates all possible storage levels and updates a gain matrix gain_matrix$[t][s]$, which represents the maximum gain achievable at time "t" when the storage level is "s".

When transitioning to storage level "i" at time "T" from storage level "j" at time "$T-1$", we need to ensure that this transition is feasible, considering the random perturbations that can occur due to $R_{\max}$.

Due to the random perturbation $R_{\max}$, when we are at level j, the actual storage level can be anywhere within the range:

$$a \in [j - R_{\max},\, j + R_{\max}]$$

The change in storage level from "a" to "i" must not exceed the maximum rate $C_{\max}$:

$$|i - a| \leqslant C_{\max}$$

To ensure this for all possible "a", we consider the worst-case scenarios where "a" is at its extreme values. Therefore, the intentional change (without randomness) must satisfy:

$$|i - j| \leqslant C_{\max} - R_{\max}$$

This constraint ensures that even after the worst-case random perturbation, we still can get to "i" after aiming for "j".

Thus, j must satisfy:

$$j \in [i - (C_{\max} - 2R_{\max}),\, i + (C_{\max} - 2R_{\max})]$$

We define:

$$A_i = \left\{ j \,\middle|\, |i - j| \leqslant C_{\max} - 2R_{\max} \right\}$$

and the feasible storage levels adjusted for random perturbations:

$$F = [R_{\max},\, S - R_{\max}]$$

The recursive dynamic programming formula is:

$$\mathcal{W}_T(i) = \max_{x \in A_i \cap F} \left\{ \begin{aligned} \min \Big[&\mathcal{W}_{T-1}(x) + (x - R_{\max} - i)P_T + P_{t-1} \cdot R_{\max}, \\ &\mathcal{W}_{T-1}(x) + (x + R_{\max} - i)P_T - P_{t-1} \cdot R_{\max} \Big] \end{aligned} \right\}$$

Explanation. :

(1) - $\mathcal{W}_T(i)$: The maximum gain achievable at time T when the storage level is i.
(2) - x: Possible previous storage levels at time $T-1$ that can transition to i at time T.
(3) - P_T: The price at time T.

The two expressions inside the min function represent the worst-case gains when the random perturbation R_t is $-R_{\max}$ and $R_{\max}$, respectively.

- When $R_t = -R_{\max}$, the actual storage level at $T-1$ is $x - R_{\max}$, so the net change to reach i is $(x - R_{\max}) - i$.
- When $R_t = R_{\max}$, the actual storage level at $T-1$ is $x + R_{\max}$, so the net change to reach i is $(x + R_{\max}) - i$.

The gain from moving from x to i is calculated by multiplying the net change in storage level by the price P_T:

$$\text{Gain} = (x + \delta R)(i) \cdot P_T - \delta R \cdot P_{T-1}$$

where $\delta R \in \{-R_{\max}, R_{\max}\}$. and the $-\delta R \cdot P_{T-1}$ it to account for paying for that change.

We use the min function to account for the worst-case scenario due to randomness, ensuring that the strategy is robust against any possible perturbation within $\pm R_{\max}$.

The max function selects the previous storage level x that yields the highest worst-case gain at the current storage level i.

Constraints.

- The set A_i ensures that the intentional change from x to i does not exceed $C_{\max} - R_{\max}$, accounting for the worst-case random perturbation.
- The feasible storage levels F ensure that storage levels remain within bounds adjusted for the maximum random perturbation, preventing the storage from exceeding its capacity or dropping below zero.

Algorithm Steps.

Initialization. The gain matrix and previous state matrix are initialized as follows:

$$\text{gain\_matrix}[0][s] = \begin{cases} 0 & \text{if } s = s_0 \\ -\infty & \text{otherwise} \end{cases}$$

$$\text{prev\_state}[0][s] = \begin{cases} s_0 & \text{if } s = s_0 \\ \text{None} & \text{otherwise} \end{cases}$$

Transition Function. For each time step t and storage level s, we consider possible transitions to storage level i at time $t+1$, constrained by the maximum charge/discharge rate and storage capacity:

$$i \in [s - (C_{\max} - 2R_{\max}),\ s + (C_{\max} - 2R_{\max})]$$

subject to:

$$R_{\max} \leqslant i \leqslant S - R_{\max}$$

We compute the possible gains considering the worst-case perturbations at the bounds $\pm R_{\max}$:

$$\text{gain1} = \text{gain\_matrix}[t][s] + P_t \cdot (s + R_{\max} - i)) - P_{t-1} \cdot R_{\max}$$

$$\text{gain2} = \text{gain\_matrix}[t][s] + P_t \cdot (s - R_{\max} - i)) + P_{t-1} \cdot R_{\max}$$

The worst-case gain is:

$$\text{worse\_gain} = \min(\text{gain1}, \text{gain2})$$

If worse_gain $>$ gain_matrix$[t+1][i]$, we update:

$$\text{gain\_matrix}[t+1][i] = \text{worse\_gain}$$

and record the previous state:

$$\text{prev\_state}[t+1][i] = s$$

Optimal Path Traceback. :

After populating the gain matrix, we find the storage level at the final time step T that maximizes the gain:

$$s_T = \arg\max_s \text{gain\_matrix}[T][s]$$

We then trace back the optimal path using the *prev_state* matrix:

$$s_{t-1} = \text{prev\_state}[t][s_t]$$

for $t = T, T-1, \ldots, 1$.

Notes. :

- Random Perturbations: The algorithm considers the worst-case random perturbations by using the min function, ensuring that the calculated gains are conservative and robust.
- Storage Constraints: Adjusting the feasible storage levels F by $R_{\max}$ prevents the storage from exceeding its capacity or dropping below zero after random perturbations.
- Charge/Discharge Limits: The intentional change in storage level is limited to $C_{\max} - 2R_{\max}$, ensuring that the total change (including randomness) is feasable.
- In the simulation loop, we apply the intended actions and then adjust the storage levels with random perturbations R_t:

Algorithm Steps.

```
(1) Initialize $gain\_matrix$ and $prev\_state$ for $t = 0$.
(2) For each time step $t = 0$ to $T - 1$:
    (a) For each storage level $s$:
        (i) For each possible next storage level $i$ within constraints:
            (A) Compute gain1 and gain2.
            (B) Determine worse_gain.
            (C) Update gain_matrix[$t + 1$][$i$] and prev_state[$t + 1$][$i$] if worse_gain is better.
(3) Trace back the optimal path from $t = T$ to $t = 0$.
```

Visualization. :

The algorithm's behavior is visualized through an animation that shows:

- The intended storage trajectory over time.
- The actual storage levels after applying random perturbations.
- The prices at each time step.

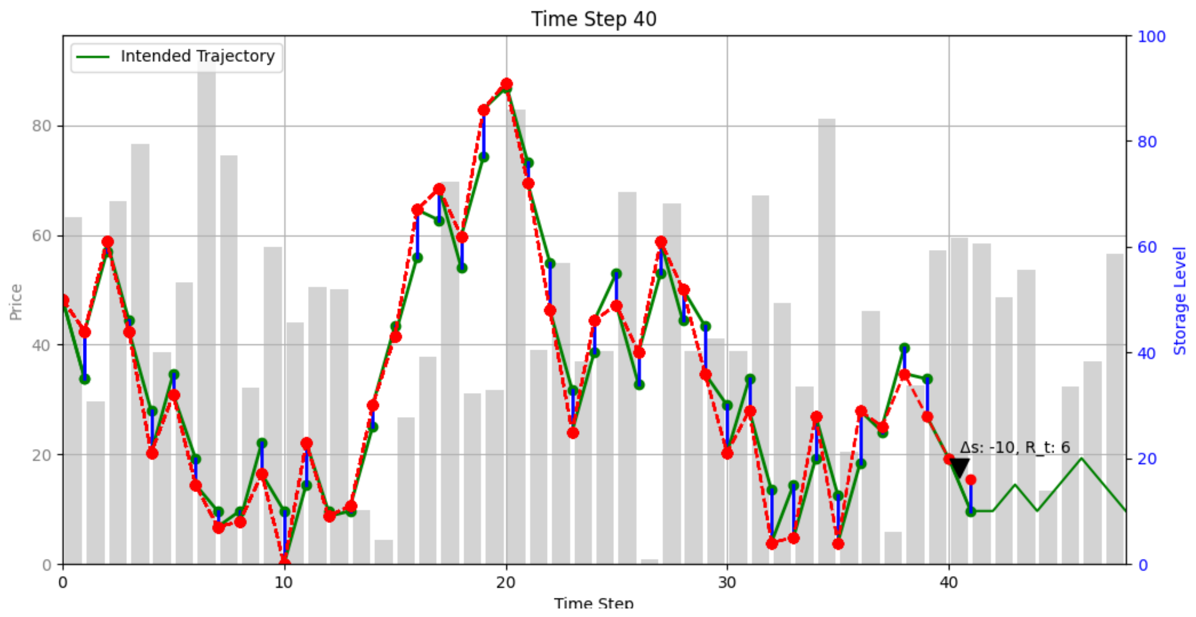

FIGURE 1. simulation

Animation Components.

- **Intended Trajectory** (Green Line): Shows the optimal path determined by the algorithm without randomness.
- **Actual Storage Level** (Red Line): Depicts the storage level after randomness is applied.
- **Randomness Effect** (Blue Vertical Lines): Illustrate the deviation from intended to actual storage levels due to R_t.
- **Prices** (Gray Bars): Display the price at each time step.

6. Minimal guaranteed value and Dynamic programming

Define by induction the functions $W_1, ..., W_{T+1}$:

$$W_t(s) = \sup_{p\in[p^{\min},p^{\max}]} \inf_{|x|\leqslant C^{\max}-R^{\max}} \sup_{|r|\leqslant R^{\max}} \big(p(x+r) + W_{t+1}(s+x+r)\big) \quad \forall t,\ \forall s \in [0,S]$$

$$W_t(s \notin [0,S]) = +\infty \quad \forall t$$

$$W_{T+1}(s \in [0,S]) = 0$$

Theorem 15. *If $\min(C^{\max}, S) \geqslant 2R^{\max}$, then the algorithm choosing*

$$x_t \in \operatorname*{arg\,min}_{|x|\leqslant C^{\max}-R^{\max}} \sup_{|r|\leqslant R^{\max}} \big(P_t(x+r) + W_{t+1}(s_{t-1}+x+r)\big) \quad \forall t$$

is acceptable and has the best maximal guaranteed value, which is $W_1(s_0)$.

In order to demonstrate this theorem, we will need several lemmas.

Proposition 16. *For all $t \in [T+1]$ and $s \in [0,S]$, we have $W_t(s) < +\infty$.*

Proof. Proceed by reverse induction. For $t = T+1$, $W_{T+1}(s \in [0,S]) = 0$.

Let t be such that $W_{t+1}(s \in [0,S]) < +\infty$. Let $s \in [0,S]$. Let $\hat{x} = \max(-C^{\max}, -s) + R^{\max}$. Now $s + \hat{x} \geqslant R^{\max}$ and $s + \hat{x} \leqslant \max(-C^{max}, -s) + R^{\max} + s \leqslant \max(s - R^{\max}, R^{\max})$. So $s + \hat{x} \in [R^{\max}, S - R^{\max}$.

Then, for all $|r| \leqslant R^{\max}$, $s+\hat{x}+r \in [0,S]$, so for all $p \in [p^{\min}, p^{\max}]$, $W_{t+1}(s+\hat{x}+r) < +\infty$. So $\sup_{|r|\leqslant R^{\max}} \big(p\hat{x} + pr + W_{t+1}(s+\hat{x}+r)\big) < +\infty$, then

$$\inf_{|x|\leqslant C^{\max}-R^{\max}} \sup_{|r|\leqslant R^{\max}} \big(px+pr+W_{t+1}(s+x+r)\big) \leqslant \sup_{|r|\leq R^{\max}} \big(p\hat{x}+pr+W_{t+1}(s+\hat{x}+r)\big) < +\infty$$

and finally $W_t(s) < +\infty$. □

Lemma 17. *Let X, Y be sets and $F\colon X \times Y \to \mathbb{R}$. Then*

$$\inf_{h\colon X\to Y} \sup_{x\in X} F(x, h(x)) = \sup_{x\in X} \inf_{y\in Y} F(x,y)$$

Proof. Let's proceed by double inequality. For all $h : X \to Y$ and $x \in X$, $F(x, h(x)) \geqslant \inf_{y\in Y} F(x,y)$. So $\sup_{x'\in X} F(x', h(x')) \geqslant \inf_{y\in Y} F(x,y)$, then

$$\sup_{x'\in X} F(x', h(x')) \geqslant \sup_{x\in X} \inf_{y\in Y} F(x,y)$$

And finally

$$\inf_{h:X\to Y} \sup_{x\in X} F(x, h(x)) \geqslant \sup x \in X \inf_{y\in Y} F(x,y)$$

Let $\varepsilon > 0$. For every $x \in X$, there exists $y_x \in Y$ s.t $F(x, y_x) \leqslant \inf_{y\in Y} F(x,y) + \varepsilon$. So $\sup_{x'\in X} F(x', y_{x'}) \leqslant \inf_{y\in Y} F(x,y) + \varepsilon$, for all x, then $\sup_{x\in X} F(x, y_x) \leqslant \sup_{x\in X}(\inf_{y\in Y} F(x,y) + \varepsilon)$. $x \mapsto y_x \in Y^X$ so $\inf_{h\colon X\to Y} \sup_{x\in X} F(x, h(x)) \leqslant \sup_{x\in X} F(x, y_x) + \varepsilon$. So finally

$$\inf_{h:X\to Y} \sup_{x\in X} F(x, h(x)) \leqslant \sup_{x\in X} \inf_{y\in Y} F(x,y) + \varepsilon$$

for all $\varepsilon > 0$, i.e.,

$$\inf_{h:X\to Y} \sup_{x\in X} F(x, h(x)) \leqslant \sup_{x\in X} \inf_{y\in Y} F(x,y) \quad \square$$

Proposition 18. *For all t and s,*

$$W_t(s) = \inf_{\substack{X_i \in \mathcal{X}_i^t \\ i \in [t,T]}} \sup_{\substack{R_t,\dots,R_T \in [\pm R^{\max}] \\ P_t,\dots,P_T \in [p^{\min}, p^{\max}]}} \left(\sum_{i=t}^{T} P_i(X_i(P_{[t,i]}, R_{[t,i-1]}, s) + R_i) \right) \tag{4}$$

where $P_{[t,i]} = (P_t, \dots, P_i)$ and

$$\begin{aligned}\mathcal{X}_i^t =& \{X_i : \mathbb{R}^{i-t+1} \times \mathbb{R}^{i-t} \times \mathbb{R} \to [\pm(C^{\max} - R^{\max})], \forall X_t, \dots, X_{i-1} \in \mathcal{X}_t^t \times \dots \times \mathcal{X}_{i-1}^t, \\ & s + X_t(P_t) + R_t + \dots + X_{i-1}(P_{[t,i-1]}, R_{[t,i-2]}, s) + R_{t-1} \in [0, S]\}\end{aligned}$$

Proof. The base case $W_T(s \in [0,S]) = 0$ is direct. Let $t+1$ be such that 4 is verified. Then

$$W_t(s) = \sup_{p \in [p^{\min}, p^{\max}]} \inf_{|x| \leqslant C^{\max} - R^{\max}} \sup_{|r| \leqslant R^{\max}} \inf_{\substack{X_i \in \mathcal{X}_i^{t+1} \\ i \in [t+1,T]}} \sup_{\substack{R_{t+1},\dots,R_T \in [\pm R^{\max}] \\ P_{t+1},\dots,P_T \in [p^{\min}, p^{\max}]}}$$

$$\left(p(x+r) + \sum_{i=t+1}^{T} P_i(X_i(P_{[t+1,i]}, R_{[t+1,i-1]}, s+x+r) + R_i) \right)$$

Now, by applying twice lemma 17,

$$W_t(s) = \inf_{Y_t : \mathbb{R} \to [\pm C^{\max} - R^{\max}]} \inf_{\substack{Y_i : \mathbb{R}^2 \to \mathcal{X}_i^{t+1} \\ i \in [t+1,T]}} \sup_{\substack{|r| \leq R^{\max} \\ p \in [p^{\min}, p^{\max}]}} \sup_{\substack{R_{t+1},\dots,R_T \in [\pm R^{\max}] \\ P_{t+1},\dots,P_T \in [p^{\min}, p^{\max}]}}$$

$$\left(p(Y_t(p) + r) + \sum_{i=t+1}^{T} P_i(Y_i(P_t, R_t)(P_{[t+1,i]}, R_{[t+1,i-1]}, s+x+r) + R_i) \right)$$

By defining the bijection,

$$\begin{aligned}\phi : & \{\mathbb{R} \to [\pm(C^{\max} - R^{\max})]\} \times \prod_{i=t+1}^{T} \{\mathbb{R}^2 \to \mathcal{X}_i^{t+1}\} \to \prod_{i=t}^{T} \mathcal{X}_i^t \\ & (Y_i)_{i \leq t \leq T} \mapsto (X_i)_{i \leq t \leq T}\end{aligned}$$

such that $X_i(P_{[t,i]}, R_{[t,i-1]}, s) = Y_i(P_t, R_t)(P_{[t+1,i]}, R_{[t+1,i-1]}, s + X_{i-1}(P_{[t,i-1]}, R_{[t,i-2]}, s) + R_{i-1})$

we have

$$W_t(s) = \inf_{X_t \in \mathcal{X}_t^t} \inf_{\substack{X_i \in X_i^t \\ i \in [t+1,T]}} \sup_{\substack{|R_t| \leq R^{\max} \\ P_t \in [p^{\min}, p^{\max}]}} \sup_{\substack{R_{t+1},\dots,R_T \in [\pm R^{\max}] \\ P_{t+1},\dots,P_T \in [p^{\min}, p^{\max}]}}$$

$$\left(P_t(X_t(P_t) + R_t) + \sum_{i=t+1}^{T} P_i(X_i(P_{[t,i]}, R_{[t,i-1]}, s) + R_i) \right)$$

$$= \inf_{\substack{X_i \in \mathcal{X}_i^t \\ i \in [t,T]}} \sup_{\substack{R_t,\dots,R_T \in [\pm R^{\max}] \\ P_t,\dots,P_T \in [p^{\min}, p^{\max}]}} \left(\sum_{i=t}^{T} P_i(X_i(P_{[t,i]}, R_{[t,i-1]}, s) + R_i) \right)$$

which finishes the induction. □

Proof of Theorem 15. Begin by proving that the algorithm is acceptable.

The bounds $|x_t| \leq C^{\max} - R^{\max}$, $\forall t$ are direct. We then prove that $s_t \in [0, S]$, $\forall t$ by forward induction. The base case is direct, let t s.t that $s_{t-1} \in [0, S]$. By Lemma 16, we have $W_t(s_{t-1}) < +\infty$. So for all $p \in [p^{min}, p^{\max}]$ and $|r| \leq R^{\max}$, $p(x_t + r) + W_{t+1}(s_{t-1} + x_t + r) <$

$+\infty$. In particular, $W_{t+1}(s_{t-1}+x_t+R_t) < +\infty$, so by definition $s_t = s_{t-1}+x_t+R_t \in [0,S]$. Therefore $s_t \in [0,S]$ for all t and the algorithm is acceptable.

Now, by proposition 18 we have

$$W_1(s_0) = \inf_{\substack{X_t \in \mathcal{X}_1^t \\ t\in[1,T]}} \sup_{\substack{R_1,\dots,R_T \in [\pm R^{\max}] \\ P_1,\dots,P_T \in [p^{\min},p^{\max}]}} \Big(\sum_{t=1}^{T} P_t(X_t(P_{[1,t]}, R_{[1,t-1]}) + R_t) \Big) \tag{5}$$

Let ALG be an acceptable algorithm which consists in choosing x_t at time t. By problem definition, x_t only depends on $P_{[1,t]}$ and $R_{[1,t-1]}$. So there exists functions $X_1,\dots,X_T \in \mathcal{X}_1^t$ such that the algorithm chooses $X_t(P_{[1,t]}, R_{[1,t-1]})$ at time t. Now, under instance $I = (P_{[1,T]}, R_{[1,T]})$ of the problem, we have $ALG(I) = \sum_{t=1}^{T} P_t(X_t(P_{[1,t]}, R_{[1,t-1]}) + R_t)$. For all $\varepsilon > 0$, there exists an instance I s.t

$$ALG(I) > \sup_I \Big(\sum_{t=1}^{T} P_t(X_t(P_{[1,t]}, R_{[1,t-1]}) + R_t) \Big) - \varepsilon \geqslant W_1(s_0) - \varepsilon$$

In conclusion, for all $v > W_1(s_0)$, there exists an instance I such that $ALG(I) > v$. So $W_1(s_0)$ is the best maximal guaranteed value. □

We complete Theorem 15 with several results: a lower bound on the optimal value that can be easily computed a priori, and the convexity of the value function W_t.

Proposition 19. *For all $t \leq T$ and $s \in [0,S]$, $W_t(s) \geqslant p^{\min}(2R^{\max} - s)$.*

Proof. Let's prove it by recursive induction. Let's start with the base case $t = T$.

$$\begin{aligned}
W_T(s) &= \sup_{p\in[p^{\min},p^{\max}]} \inf_{|x|\leqslant C^{\max}-R^{\max}} \sup_{|r|\leqslant R^{\max}} \Big(px + pr + \begin{cases} 0 & \text{if } s+x+r \in [0,S] \\ +\infty & \text{otherwise} \end{cases} \Big) \\
&= \sup_{p\in[p^{\min},p^{\max}]} \inf_{|x|\leqslant C^{\max}-R^{\max}} \Big(px + \begin{cases} pR^{\max} & \text{if } s+x \in [R^{\max}, S-R^{\max}] \\ +\infty & \text{otherwise} \end{cases} \Big) \\
&= \sup_{p\in[p^{\min},p^{\max}]} \begin{cases} p(2R^{\max} - C^{\max}) & \text{if } s \geqslant C^{\max} \\ p(2R^{\max} - s) & \text{otherwise} \end{cases}
\end{aligned}$$

$$W_T(s) = \begin{cases} p^{\min}(2R^{\max} - C^{\max}) & \text{if } s \geqslant C^{\max} \\ p^{\min}(2R^{\max} - s) & \text{if } s \in [2R^{\max}, C^{\max}] \\ p^{\max}(2R^{\max} - s) & \text{if } s \leqslant 2R^{\max} \end{cases}$$

So $W_T(s) \geqslant p^{\min}(2R^{\max} - s)$ for all $s \in [0,S]$.

Let now be t such that $W_{t+1}(s) \geqslant p^{\min}(2R^{\max} - s)$ for all $s \in [0, S]$. Then for all s,

$$\begin{aligned}
W_t(s) &= \sup_{p\in[p^{\min},p^{\max}]} \inf_{|x|\leqslant C^{\max}-R^{\max}} \sup_{|r|\leqslant R^m} \Big(p(x+r) - W_{t+1}(s+x+r)\Big) \\
&\geqslant \sup_{p\in[p^{\min},p^{\max}]} \inf_{|x|\leqslant C^{\max}-R^{\max}} \sup_{|r|\leqslant R^{\max}} \Big((p-p^{\min})(x+r) + p^{\min}(2R^{\max}-s)\Big) \\
&= p^{\min}(2R^{\max}-s) + \sup_{p\in[p^{\min},p^{\max}]} \Big((p-p^{\min})(\inf_{|x|\leqslant C^{\max}-R^{\max}} x + \sup_{|r|\leqslant R^{\max}} r)\Big) \\
&= p^{\min}(2R^{\max}-s) + \sup_{p\in[p^{\min},p^{\max}]} (p-p^{\min})(2R^{\max}-C^{\max}) \\
&= p^{\min}(2R^{\max}-s) \qquad \square
\end{aligned}$$

Proposition 20. *For every t, W_t is convex on $[0, S]$.*

To prove Proposition 20, we will need two convexity lemmas.

Lemma 21 (Danskin's theorem)**.** *Let X be a convex set, Y be a set and $F : X \times Y \to \mathbb{R}$. If for all $y \in Y$, $x \mapsto F(x, y)$ is convex, then $x \mapsto \sup_{y\in Y} F(x, y)$ is convex.*

Proof. Let $x, x' \in X$ and $u \in [0, 1]$. For all $y \in Y$ we have

$$F(ux + (1-u)x', y) \leqslant uF(x, y) + (1-u)F(x', y) \leqslant u \sup_{y\in Y} F(x, y) + (1-u) \sup_{y\in Y} F(x', y)$$

So $\sup_y F(ux + (1-u)x', y) \leqslant u \sup_{y\in Y} F(x, y) + (1-u) \sup_{y\in Y} F(x', y)$. □

Lemma 22 (Convexity of the optimal value function)**.** *Let X and Y be convex sets and $f : X \times Y \to \mathbb{R}$. If F is jointly convex, then $x \mapsto \inf_y F(x, y)$ is convex.*

Proof. Let $x, x' \in X$ and $u \in [0, 1]$.

$$\begin{aligned}
\inf_{y\in Y} F(ux + (1-u)x', y) &\leqslant \inf_{\substack{y_1,y_2\in Y \text{ s.t} \\ y=uy_1+(1-u)y_2}} F(ux + (1-u)x', y) \\
&= \inf_{y,y'\in Y} F(ux + (1_u)x', uy + (1-u)y') \\
&\leqslant \inf_{y,y'\in Y} \big(uF(x, y) + (1-u)F(x', y')\big) \\
&= u \inf_{y\in Y} F(x, y) + (1-u) \inf_{y'\in Y} F(x', y') \qquad \square
\end{aligned}$$

Proof of proposition 20. We proceed by reverse induction. The base case $t = T+1$ is trivial.

Let t be such that W_{t+1} is convex. Let $p \in [p^{\min}, p^{\max}]$.

For every r, define $f_{t,p,r} : (x, s) \mapsto p(x+r)+W_{t+1}(s+x+r)$. Let $(x, s), (x's') \in [\pm(C^{\max} - R^{\max}] \times [0, S]$ and $u \in [0, 1]$.

$$\begin{aligned}
f_{t,p,r}(u(x, s) + (1-u)(x', s')) &= p(ux + (1-u)x' + r) + W_{t+1}(u(s+x) + (1-u)(s'+x') + r) \\
&= up(x+r) + (1-u)p(x'+r) \\
&\quad + W_{t+1}(u(s+x+r) + (1-u)(s'+x'+r)) \\
&\leqslant u\big(p(x+r) + W_{t+1}(s+x+r)\big) \\
&\quad + (1-u)\big(p(x'+r) + W_{t+1}(s'+x'+r)\big) \\
&= uf_{t,p,r}(x, s) + (1-u)f_{t,p,r}(x', s')
\end{aligned}$$

So $f_{t,p,r}$ is convex for all r. Therefore, by Danskin's theorem, $f_{t,p} : (x,s) \mapsto \sup_r f_{t,p,r}$ is convex. So, by lemma 22 the function $s \in [0,S] \mapsto \inf_x f_{t,p}(x,s)$ is convex for all p. Therefore, again by Danskin's theorem, $W_t : s \mapsto \sup_p \inf_x f_{t,p}(x,s)$ is convex. □

Several remarks can be deduced from this.

Remark 23. *The function $r \mapsto p(x+r) + W_{t+1}(s+x+r)$ is convex, therefore its supremum is attained on the boundaries of the interval $[\pm R^{\max}]$. So the recursive defintion of W_t is equivalent to*

$$W_t(s) = \sup_{p \in [p^{\min}, p^{\max}]} \inf_{|x| \leqslant C^{\max} - R^{\max}} \max_{\epsilon = \pm 1} \big(p(x + \epsilon R^{\max}) + W_{t+1}(s + x + \epsilon R^{\max}) \big) \quad \forall t, \ \forall s \in [0,S]$$

Remark 24. *The function $x \mapsto f_{t,p}(x,s)$, which the algorithm minimizes at time t when choosing x_t is convex.*

Conjecture 25. *For every t, W_t is decreasing and piecewise linear.*

References

[1] Rose Sossou Edou. Operating a battery at minimum cost for reserve commitment using energy arbitrage. Master's thesis, CERMICS, Ecole nationale des ponts et chaussées, 2024.

[2] Jeff Giliberti and Andreas Karrenbauer. Improved online algorithm for fractional knapsack in the random order model. *CoRR*, abs/2109.04428, 2021.

[3] Bo Sun, Ali Zeynali, Tongxin Li, Mohammad Hajiesmaili, Adam Wierman, and Danny H. K. Tsang. Competitive algorithms for the online multiple knapsack problem with application to electric vehicle charging, 2020.

[4] Lin Yang, Mohammad Hassan Hajiesmaili, and Wing Shing Wong. Online linear programming with uncertain constraints : (invited paper). *2019 53rd Annual Conference on Information Sciences and Systems (CISS)*, pages 1–6, 2019.