To determine the **most advantageous options to exercise** in a **portfolio of 1,000,000 options**, we can apply several filtering and ranking strategies based on **option moneyness, intrinsic value, and expected profit**. Here’s a structured approach:

**1. Classifying Options: Moneyness & Profitability**

Options can be categorized as:

* **In the Money (ITM)** – Profitable to exercise (for calls: S>KS > KS>K, for puts: S<KS < KS<K)
* **At the Money (ATM)** – Borderline case (S≈KS \approx KS≈K)
* **Out of the Money (OTM)** – Not profitable to exercise (for calls: S<KS < KS<K, for puts: S>KS > KS>K)

**Formula to Check If Exercising is Profitable:**

For **call options**:

Profit=S−K−C\text{Profit} = S - K - CProfit=S−K−C

For **put options**:

Profit=K−S−P\text{Profit} = K - S - PProfit=K−S−P

where:

* SSS = Current stock price
* KKK = Strike price
* CCC = Call option premium (cost of buying the option)
* PPP = Put option premium

**Exercise only if profit is positive.**

**2. Ranking Options: Expected Value Approach**

If options are **European-style**, they can only be exercised at expiration, so we look at their expected profit at expiration using Black-Scholes.

For **American-style options**, they can be exercised early, so we must check whether exercising now is better than holding for future value. The decision should consider:

* **Delta (Δ)** – Sensitivity to stock price changes
* **Theta (Θ)** – Time decay of option value
* **Implied Volatility (σ)** – Higher volatility means greater uncertainty, affecting early exercise decisions

**Priority Exercise Rules:**

* **Exercise deep ITM options with low time value (near expiration).**
* **Hold options if Black-Scholes predicts higher future value.**

**3. Computational Strategy for 1,000,000 Options**

Given the large dataset, parallelization and GPU acceleration (TensorFlow, CuPy, or Numba) can help speed up computation.

**Steps for Efficient Processing:**

1. **Load and preprocess option data** (tickers, prices, expirations, premiums).
2. **Calculate Black-Scholes price for each option** (using scipy.stats.norm or GPU acceleration).
3. **Filter ITM options** – Ignore OTM options immediately.
4. **Sort ITM options by expected profit and time value** – Rank them based on best exercise candidates.
5. **Optimize for capital constraints** – If exercising all ITM options isn’t feasible, prioritize the most profitable ones.

**4. Optimizing Execution With ML (Optional)**

* **Use LSTMs or GARCH models** to predict volatility more accurately.
* **Reinforcement Learning (RL) models** can learn optimal early exercise strategies dynamically.

**Would You Like Code for Filtering & Ranking Options?**

Let me know if you want a Python script to process a **1,000,000-option portfolio efficiently**!

WE HAVE THIS CODE!!

Option-Filter\_Ranking.ipynb

**What This Script Does:**

1. **Loads option data** from a CSV file (large\_options\_data.csv).
2. **Calculates intrinsic value & expected profit** using Black-Scholes for calls & puts.
3. **Filters In-the-Money (ITM) options** – Only profitable ones are considered.
4. **Ranks ITM options** based on the highest profit.
5. **Uses parallel computation** to handle **1,000,000+ options efficiently**.

**How It Works Efficiently**

✅ **Numba JIT Compilation** → @njit(parallel=True) speeds up computations.  
✅ **Vectorized NumPy operations** → Avoids slow loops.  
✅ **Efficient Filtering & Sorting** → Only keeps profitable ITM options.

**Next Steps**

* Run this script after generating **large\_options\_data.csv**.
* Analyze **most\_advantageous\_options.csv** for top exercise candidates.
* Want **GPU acceleration (TensorFlow or CuPy)?** Let me know!