Group: 6487 - Stochastic Modeling GWP 3

Shivansh Kumar, Nikita Berezin, Viacheslav Karpov

August 13, 2024

Step 1

Our group have selectively read a paper "Risk-aware multi-armed bandit problem with application to portfolio selection" by Huo et.al [1]. This paper explores the risk-aware multi-armed bandit (MAB) problem and its application to portfolio selection in financial markets. The authors develop a sequential portfolio selection algorithm that balances reward maximization and risk minimization using integrating graph theory and a coherent risk measure.

Step 2

We thoroughly read the paper titled "Risk-aware multi-armed bandit problem with application to portfolio selection" by Huo et.al [1] and we are going to describe the portfolio selection problem as a multi-armed bandit problem:

We can view the portfolio selection problem as a multi-armed bandit problem with the following mapping:

- **Arms:** Each asset in the basket of K selected assets represents an arm of the bandit.
- **Actions:** Choosing a portfolio ω_t = (ω_1,t, ..., ω_K,t) is equivalent to pulling multiple arms simultaneously with different weights.
- **Rewards:** The reward at each trial is the weighted sum of returns ω_tRt, where Rt = (R1,t, ..., RK,t) are the returns of the assets.
- **Exploration vs. Exploitation:** The algorithm must balance exploring different asset combinations and exploiting the ones that have performed well historically.
- **Learning:** The algorithm learns the assets' expected returns over time, adjusting the portfolio weights accordingly.

Now Let's elaborate on it using a pseudocode in Algorithm 1 :

Algorithm 1 Sequential Portfolio Selection

procedure SEQUENTIALPORTFOLIOSELECTION(δ, N)
  historical_returns $\leftarrow$ GETHISTORICALRETURNS(δ)
  $K \leftarrow$ SELECTASSETS(*historical_returns*)
  cumulative_reward $\leftarrow 0$
  for $t \leftarrow 1$ to N **do**
    $\omega_t \leftarrow$ CHOOSEPORTFOLIO(K, *historical_returns*, t)
    $R_t \leftarrow$ OBSERVERETURNS(K)
    reward $\leftarrow$ CALCULATEREWARD(ω_t, R_t)
    cumulative_reward $\leftarrow$ *cumulative_reward* + *reward*
    UPDATEHISTORICALDATA(*historical_returns*, R_t)
  end for
  return *cumulative_reward*
end procedure
procedure GETHISTORICALRETURNS(δ)
  return matrix of historical returns for all assets over δ periods
end procedure
procedure SELECTASSETS(*historical_returns*)
  return K ▷ Number of selected assets
end procedure
procedure CHOOSEPORTFOLIO(K, *historical_returns*, t)
  return ω_t ▷ Vector of weights summing to 1
end procedure
procedure OBSERVERETURNS(K)
  return R_t ▷ Vector of K returns
end procedure
procedure CALCULATEREWARD(ω_t, R_t)
  return $\sum_{i=1}^{K} \omega_{i,t} R_{i,t}$
end procedure
procedure UPDATEHISTORICALDATA(*historical_returns*, R_t)
  Update *historical_returns* with new data R_t
end procedure
$\delta \leftarrow 100$ ▷ Number of historical periods
$N \leftarrow 1000$ ▷ Number of investment periods
result $\leftarrow$ SEQUENTIALPORTFOLIOSELECTION(δ, N)
print "Cumulative reward after N periods: *result*"

Step 3

After that now we are going to collect sample data to work further, the date assigned to us for the daily return during Sep 2008 and Oct 2008.

Each Team member was assigned a different task :

- **Team member A:** collected the data for 15 financial institutions (JPM, WFC, BAC, C, GS, USB, MS, KEY, PNC, COF, AXP, PRU, SCHW, BBT, STI) using Yahoo Finance [2] for the assigned period.

- **Team member B:** collected the data for 15 non-financial institutions (KR, PFE, XOM, WMT, DAL, CSCO, HCP, EQIX, DUK, NFLX, GE, APA, F, REGN, CMS) using Yahoo Finance [2] for the assigned period.

- **Team member C:** Was assigned to combine the data into a Python time series data structure to help further analysis, the member was also assigned to compute the daily return of all the 30 series.

  In order to summarize the key information about considered companies we prepared below the table with tickers along with the company names, industries, and sectors corresponding to the provided stock tickers:

Ticker	Company Name	Industry	Sector
PM	Philip Morris International Inc.	Tobacco	Consumer Staples
WFC	Wells Fargo & Co.	Banks	Financials
BAC	Bank of America Corp.	Banks	Financials
C	Citigroup Inc.	Banks	Financials
GS	Goldman Sachs Group Inc.	Capital Markets	Financials
USB	U.S. Bancorp	Banks	Financials
MS	Morgan Stanley	Capital Markets	Financials
KEY	KeyCorp	Banks	Financials
PNC	PNC Financial Services Group Inc.	Banks	Financials
COF	Capital One Financial Corp.	Consumer Finance	Financials
AXP	American Express Co.	Consumer Finance	Financials
PRU	Prudential Financial Inc.	Insurance	Financials
SCHW	Charles Schwab Corp.	Capital Markets	Financials
BBT	BB&T Corp. (now part of Truist Financial)	Banks	Financials
STI	SunTrust Banks Inc. (now part of Truist Financial)	Banks	Financials
KR	The Kroger Co.	Food & Staples Retailing	Consumer Staples
PFE	Pfizer Inc.	Pharmaceuticals	Health Care
XOM	Exxon Mobil Corp.	Oil, Gas & Consumable Fuels	Energy
WMT	Walmart Inc.	Food & Staples Retailing	Consumer Staples
DAL	Delta Air Lines Inc.	Airlines	Industrials
CSCO	Cisco Systems Inc.	Communications Equipment	Information Technology
HCP	HCP Inc. (now Healthpeak Properties Inc.)	Health Care REITs	Real Estate
EQIX	Equinix Inc.	Specialized REITs	Real Estate
DUK	Duke Energy Corp.	Electric Utilities	Utilities
NFLX	Netflix Inc.	Entertainment	Communication Services
GE	General Electric Co.	Industrial Conglomerates	Industrials
APA	APA Corp. (formerly Apache Corp.)	Oil, Gas & Consumable Fuels	Energy
F	Ford Motor Co.	Automobiles	Consumer Discretionary
REGN	Regeneron Pharmaceuticals Inc.	Biotechnology	Health Care
CMS	CMS Energy Corp.	Multi-Utilities	Utilities

Table 1: List of Companies with Tickers, Industries, and Sectors

As we can see the majority of these companies are from Financial sector.

We decided to use the Pandas DataFrame as a container for all historical data for considered tickers. The individual tickers' quotes are located in columns and we used data as index for this table. Therefore in rows we stored information for any specific day. That allows as easily manipulate data and perform various transformations. Below in Figure 1 is presented a fragment of this DataFrame:

Date	JPM	WFC	BAC	C	GS	USB	MS	KEY	PNC	COF	...
2008-09-03	0.018466	-0.006408	0.030322	0.026164	0.013851	0.017918	0.021065	0.008730	0.009259	0.016474	...
2008-09-04	-0.045329	-0.043211	-0.071602	-0.066803	-0.040033	-0.039454	-0.043396	-0.062156	-0.018483	-0.051029	...
2008-09-05	0.044579	0.051567	0.053268	0.042077	0.014543	0.034440	0.025285	0.088088	0.021168	0.031849	...
2008-09-08	0.049243	0.075641	0.077567	0.065548	0.039757	0.036652	0.046180	0.059368	0.033517	0.089912	...
2008-09-09	-0.050060	-0.071216	-0.063634	-0.070866	-0.047487	-0.055392	-0.066328	-0.042941	-0.051185	-0.066284	...
2008-09-10	-0.001773	0.017004	-0.003690	-0.010593	-0.025237	-0.006550	-0.036634	-0.044867	-0.027316	0.001758	...
2008-09-11	0.057106	0.067823	0.020370	-0.003748	-0.003553	0.056201	-0.005396	0.019108	0.030482	0.016674	...

Figure 1: Consolidated DataFrame (fragment)

Here in Figure 2, we can see the Combined data summary statistics.

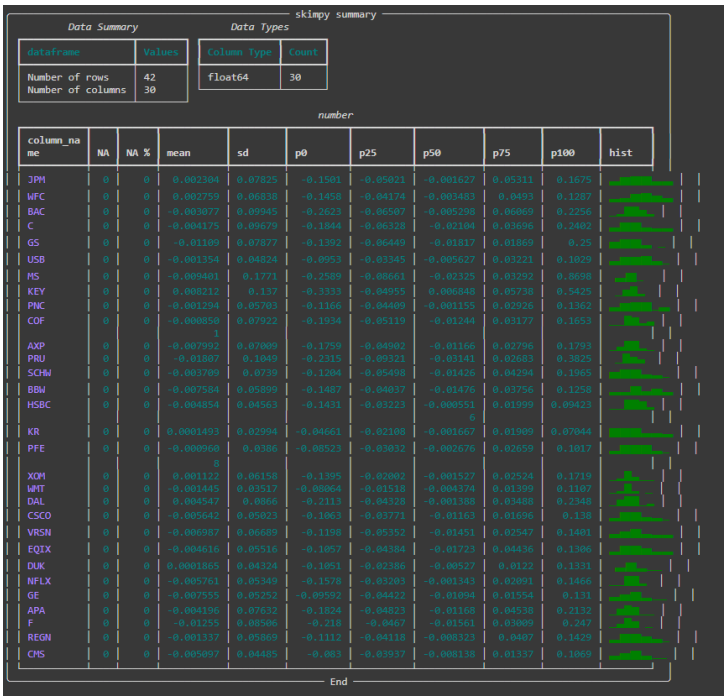

Figure 2: Combined Data

Step 4

We performed correlation analysis and hierarchical clustering on the daily returns data to understand the relationships between different assets. Figure 3 shows the correlation heatmap of asset returns along with a final agglomerative hierarchical clustering dendrogram.

To further analyze the structure of our asset portfolio, we employed hierarchical clustering. This technique creates a hierarchy of clusters, allowing us to examine the relationships between assets at different levels of similarity. We used agglomerative hierarchical clustering, which follows a bottom-up approach:

1. Start with each asset as a separate cluster.
2. Compute the distances (dissimilarities) between all pairs of clusters.
3. Merge the two clusters closest to each ither according to the selected distance metric.
4. Repeat steps 2 and 3 until all assets are in a single cluster.

For our analysis, we used the correlation distance (1 - correlation) as the dissimilarity measure and Ward's method as the linkage criterion. Ward's method minimizes the total within-cluster variance, making it particularly suitable for financial data where we want to group assets with similar return patterns.

The resulting hierarchical structure is visualized using a dendrogram (Figure 3), which shows how the clusters are formed at each step of the algorithm.

A good stopping criteria for our case would be to stop the joining iterations once the algorithm decreases the number of clusters to only 4. This number is optimal from the perspective of the inter-cluster distance that will be eliminated by joining clusters 3 and 4 together. Based on their distance, they are too different to be merged, so the clusterisation process should be terminated. The final clusters will contain the following securities:

- **Clusters 1 & 2**: Outlier clusters containing 5 securities (might be also manually merged into the single cluster)
- **Cluster 3**: Banking and Investment sector
- **Cluster 4**: Non-financial multi-sector securities

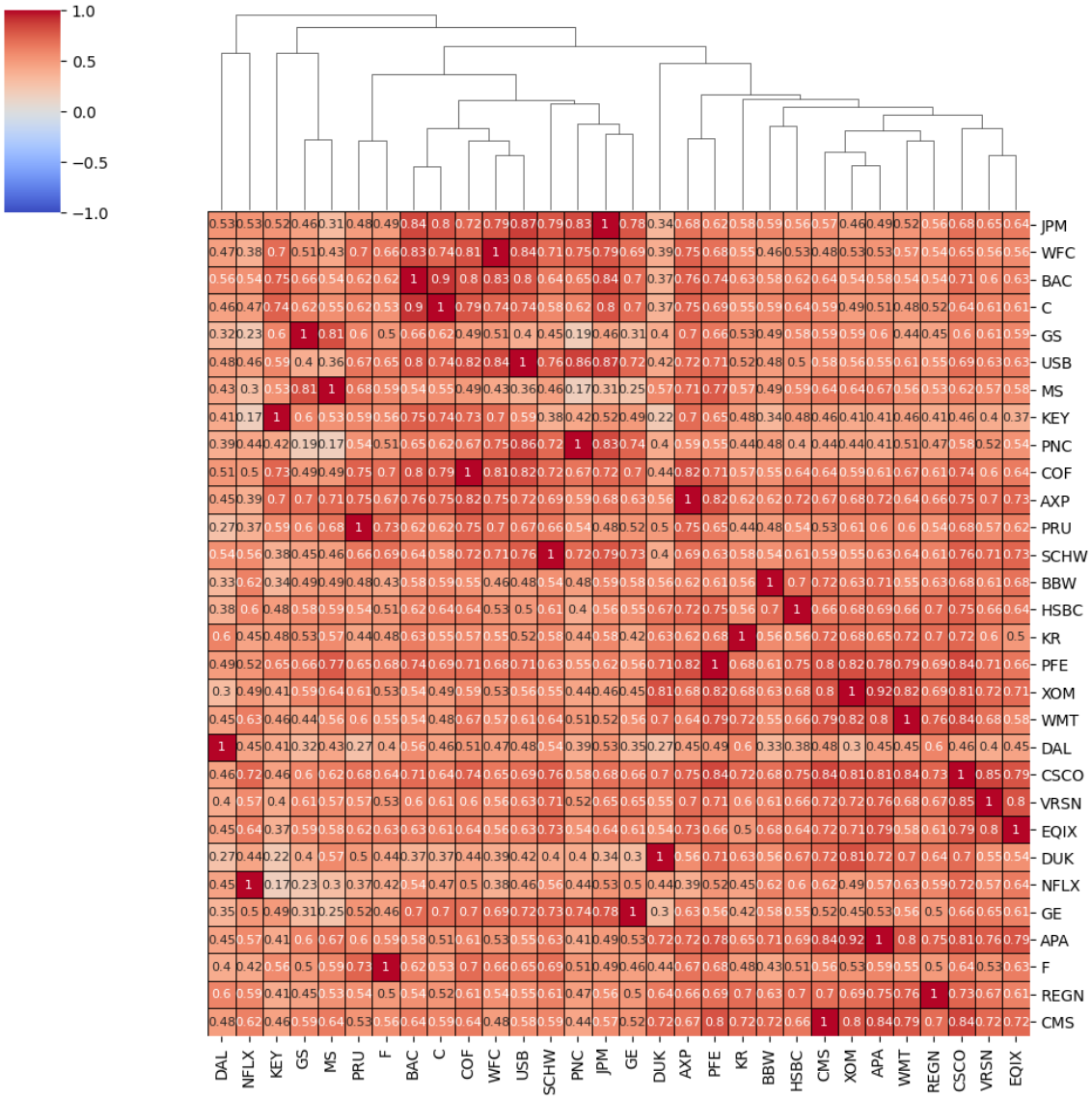

Figure 3: Hierarchical Clustering Dendrogram

Following the final clustering dendogram we can conclude, that the securities inside the same sector are closely correlated together not only by their returns but also by correlation portraits. This observation might reveal the optimal number of securities to be held in the single portfolio to reach a good diversification. These insights guide our asset selection process for the multi-armed bandit algorithm, ensuring a diverse and balanced portfolio that accounts for the underlying correlation structure of the market.

Step 5 & 6

As we have done our correlation and cluster analysis in the previous section, Let's move into the implementation of the multi-armed bandit algorithm, we are going to start with UCB algorithm [3] because we need to ensure a certain degree of exploration in the actions, as we are always uncertain about our estimates of the action values. The greedy actions are the currently optimal ones from a backward-looking perspective, but some of the other actions may be better, even though we don't know it yet. The standard ε-greedy action selection forces the exploration of non-greedy actions but without any particular preference.

To overcome this limitation and prioritize the exploration of actions that we are more uncertain about, we can follow the next optimal choice, which is given by the following formula:

$$A_t = \arg\max_a \left\{ Q_t(a) + c\sqrt{\frac{\ln t}{N_t(a)}} \right\}$$

Here, the new term adjusts the action selection based on the degree of "unexploration" of the action, relative to the number of steps completed. The parameter $c > 0$ controls the degree of exploration. If $N_t(a) = 0$ (i.e., the action has not been tried yet), the action is considered to be a maximizing action to avoid the ratio becoming infinite.

We are going to use this procedure, which prioritizes the exploration of uncertain actions, in our application of stock-picking strategies. Now we going to move into the implementation of the UCB (Upper Confidence bound) algorithm using selected cluster portfolio daily return data data, But let's first understand the UBC algorithm by the following pseudocode in Algorithm 2:

Algorithm 2 Upper Confidence Bound (UCB) Algorithm for Stock Picking

Input:
$k \leftarrow$ number of stocks
$T \leftarrow$ total number of trading days
$R[i,t] \leftarrow$ daily return of stock i on day t
Initialize:
$N[i] \leftarrow 0$ for all $i \in \{1, 2, \ldots, k\}$ ▷ Number of times stock i has been selected
$R_{\text{total}}[i] \leftarrow 0$ for all $i \in \{1, 2, \ldots, k\}$ ▷ Total return obtained from stock i
for each trading day $t = 1, 2, \ldots, T$ **do**
  Calculate UCB values:
  for each stock $i \in \{1, 2, \ldots, k\}$ **do**
    if $N[i] = 0$ **then**
      $\text{UCB}[i] \leftarrow \infty$ ▷ Ensure unselected stocks are considered
    else
      $\text{UCB}[i] \leftarrow \frac{R_{\text{total}}[i]}{N[i]} + C\sqrt{\frac{\ln(t)}{N[i]}}$ ▷ Use the modified UCB formula
    end if
  end for
  Select the stock with the highest UCB value:
  $S \leftarrow \arg\max_i \text{UCB}[i]$ ▷ Choose the stock with the maximum UCB value
  Observe the daily return of the selected stock:
  $r \leftarrow R[S,t]$ ▷ Observe the daily return of stock S
  Update the counts and total returns:
  $N[S] \leftarrow N[S] + 1$ ▷ Increment the count for the selected stock
  $R_{\text{total}}[S] \leftarrow R_{\text{total}}[S] + r$ ▷ Add the observed return to the total return
end for
Return: The stock with the highest average return

Step 7 & 8

To compare with the UCB algorithm, we implemented the Epsilon-greedy algorithm, which is considered to be an alternative to UCB for balancing between the exploration and exploitation strategies, but both the approaches might be implemented in the same time. The Epsilon-greedy algorithm provides a simple yet effective strategy for balancing exploration and exploitation strategies by including an $epsilon$ probability of picking the next action randomly instead of exploiting the action with the highest expected reward. During the exploitation step the algorithm capitalizes on the knowledge it has previously accumulated, while during the exploration step the algorithm continues to gather information about all actions, potentially discovering better options that may have been overlooked. This simple strategy ensures that all actions are tried often, while still favoring actions that appear to be optimal based on current knowledge.

The Epsilon-greedy algorithm works as follows:

- With probability $1 - \epsilon$, the algorithm exploits the current knowledge by selecting the action with the highest estimated value (exploitation step).
- With probability ϵ, the algorithm explores by selecting a random action uniformly (exploration step).

The value of ϵ determines the balance between exploration and exploitation. A higher ϵ leads to more exploration and better adaptability to non-stationary environments, while a lower ϵ favors exploitation.

The pseudocode for the Epsilon-greedy algorithm applied to our portfolio selection problem is the following:

Algorithm 3 Epsilon-Greedy Algorithm for Portfolio Selection

RunEpsilonGreedy $R, K, \epsilon, \alpha, N$
Input:
$R[i, t]$: Matrix of asset returns
K: Number of assets
ϵ: Exploration probability
α: Step size
N: Number of trials
Initialize:
$Q \leftarrow \text{zeros}(K)$ ▷ Values for each asset action
for each trading day $t = 1, 2, \ldots, T$ **do**
    if random() $< \epsilon$ **then**
        $a_t \leftarrow \text{random\_choice}(K)$ ▷ Exploration step
    else
        $a_t \leftarrow \arg\max_a Q[a]$ ▷ Exploitation step
    end if
    Observe reward r_t for chosen asset a_t
    $Q[a_t] \leftarrow Q[a_t] + \alpha(r_t - Q[a_t])$
end for
return Q

The key formula in the Epsilon-greedy algorithm is the Q-value update rule:

$$Q_{t+1}(a_t) = Q_t(a_t) + \alpha(r_t - Q_t(a_t))$$

where:

- $Q_t(a_t)$ is the estimated value of action a_t at time t
- α is the learning rate or an update step size wrt to a deviation of a new action result compared to the previous value.
- r_t is the observed reward at time t

The Epsilon-greedy algorithm adapted a wide application in non-stationary environments. The constant exploration allows the algorithm to adapt to changes in the underlying reward distributions. In financial markets, where asset returns can change over time due to various factors, this adaptability is crucial. Also as a side effect, the random exploration provides a degree of robustness against getting stuck in local optima, which can be particularly valuable in volatile markets.

Step 9

In this section we will analyse results of UCB and $epsilon - greedy$ algorithms and compare our results with the conclusion made by Ho in his paper.

First of all, we prepared the comparison of UCB and $epsilon - greedy$ algorithms in Figure 4 relation to cumulative rewards.

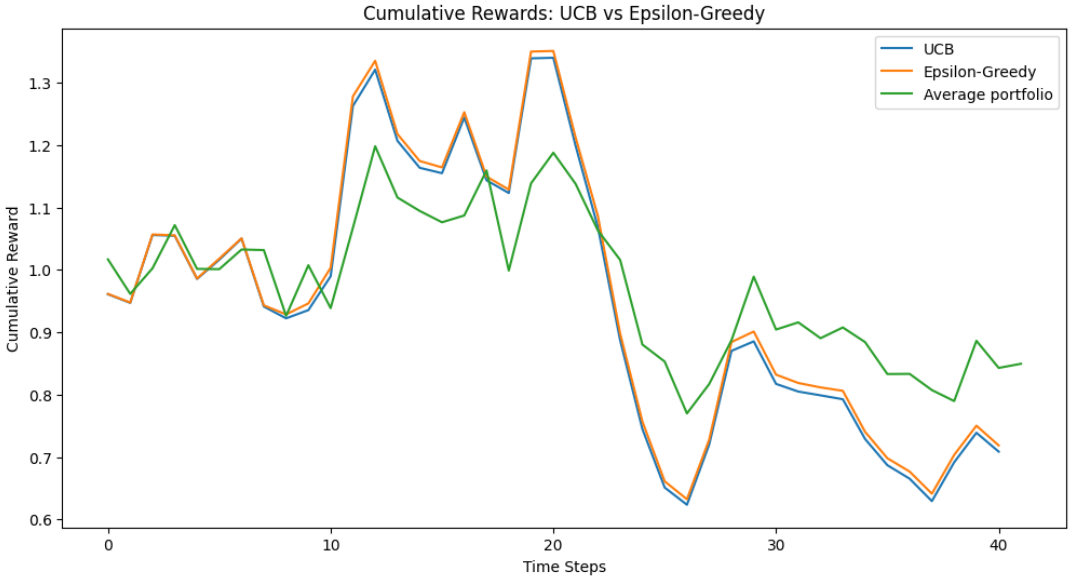

Figure 4: Comparison of Cumulative Rewards

It can be seen from the graph above that the $epsilon - greedy$ algorithm is slightly under-preformed UCB although both showed a strong correlation with each other. On the other hand, the $averageportfolio$ demonstrated better results: lover volatility, and on average it over-performed both UCB and $epsilon - greedy$ algorithms.
Another important characteristic of the considered algorithms is the Optimal Action frequency.This metric refers to how often an algorithm selects the best possible action that maximizes rewards. Therefore, this metric directly reflects the efficiency and effectiveness of the algorithm during periods of high volatility. A higher Optimal Action Frequency indicates that the algorithm is more consistently identifying and choosing the most beneficial actions, which is crucial for achieving superior cumulative rewards over time in Figure 5.

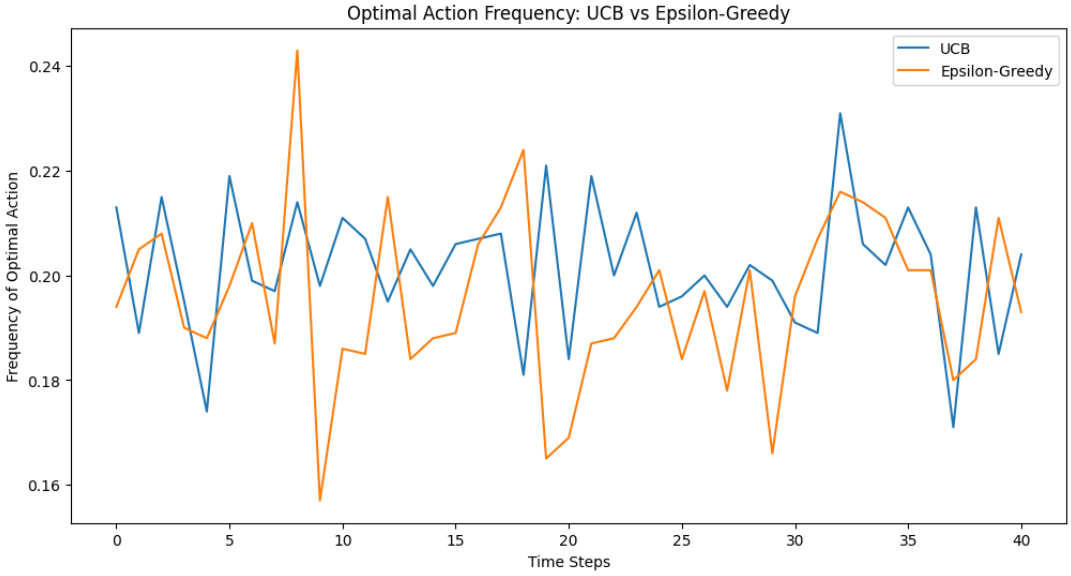

Figure 5: Comparison of Optimal Auction frequency

Despite their volatility over the considered period, both the UCB and $epsilon$-greedy algorithms demonstrated a similar average

value of Optimal Action Frequency, indicating that, on average, they were equally effective at identifying and selecting the best possible actions. This similarity in their ability to choose optimal actions means that, over time, despite the short-term fluctuations and differences in their volatility, the cumulative rewards generated by both algorithms ended up being quite similar.

The final metric we considered was the comparison of the Cumulative returns in Figure 6.

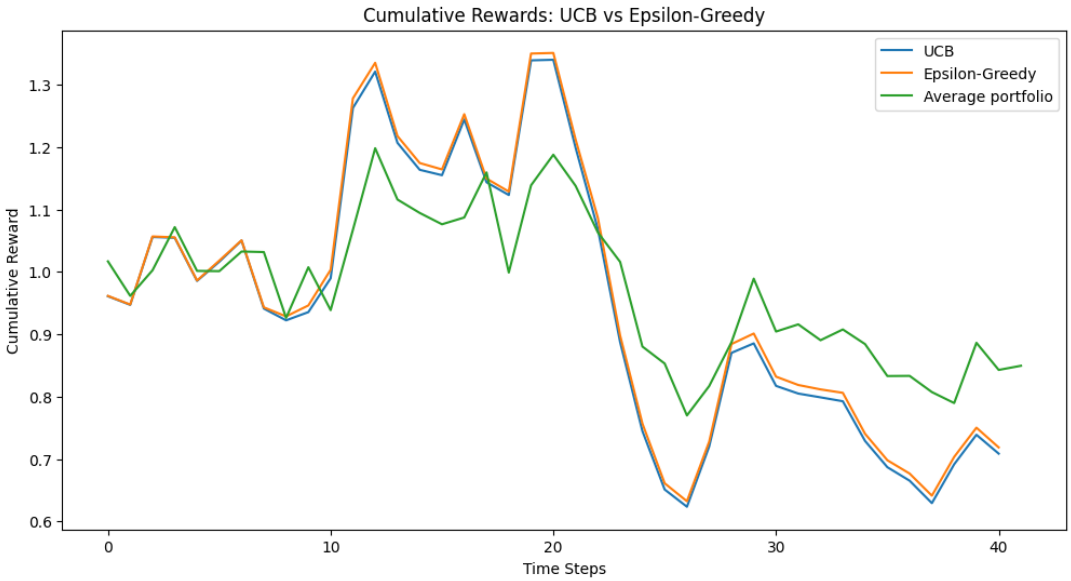

Figure 6: Comparison of Cumulative Return

As we can see from the graph, all algorithms finished with losses (cumulative return is below 1). The demonstrated behaviors mirrors the picture we observed on the graph comparing the Cumulative rewards: - UCB and $epsilon - greedy$ algorithms demonstrated similar behavior, outperformed the average portfolio in the beginning of considered period but suffered a severe draw-down (more than 60%) - $averageportfolio$ demonstrated lower volatility, lower drawdown and finished with higher cumulative return (lower cumulative loss).
Based on the prepared analysis we have made several observations:

- The selected time period between September 2008 and October 2008 is the active phase of Global Financial Crisis, triggered by the collapse of Lehman Brothers on 15 September 2008. It is characterised by unprecedented volatility and significant stock prices declines, particularly in the financial, banking, and real estate sectors. Banks faced severe liquidity shortages, and real estate-related assets plummeted in value due to widespread mortgage defaults, leading to massive stocks sell-offs and further amplifying market decline. All these factors impacted performance of our models in several ways.

- Financial markets were highly volatile due to the global financial crisis, which significantly increased the volatility of algorithms used for decision-making under uncertainty, such as UCB (Upper Confidence Bound) and $epsilon$-greedy.

- Both algorithms aim to balance exploration and exploitation to maximize cumulative rewards, but they differ in their approaches. The $epsilon - greedy$ algorithm introduces randomness in decision-making by choosing a random action with probability $epsilon$, leading to slightly over-performing the UCB algorithm, which systematically adjusts its action-selection based on the confidence intervals of rewards.

- Despite their differences, both algorithms showed strong correlation in cumulative rewards, reflecting similar responses to the market conditions. However, an $averageportfolio$ method, which involved a diversified and as result less aggressive strategy, demonstrated better average results during this turbulent period. This method had lower volatility, as it smoothed out the extreme fluctuations that affected more reactive algorithms like UCB and $epsilon$-greedy, resulting in a more stable and slightly higher overall performance.

In addition to that analysis, we also considered how our results correspond with that received by Ho in his paper. Below we presented the extract from the "Simulation results" section of Ho paper in Figure 7:

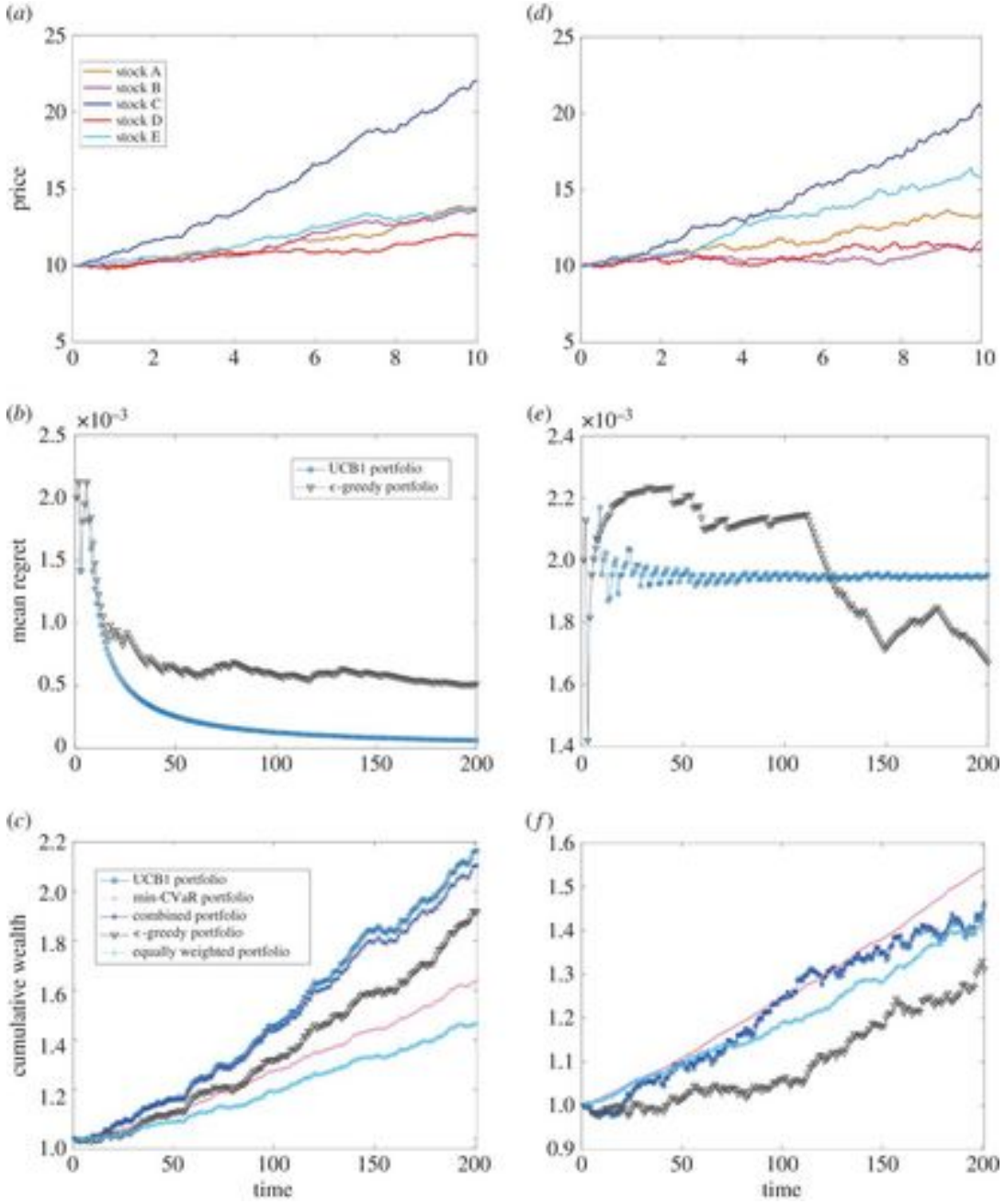

Figure 7: Results from Ho papaer

- The Ho study demonstrates that the UCB approach often achieves the highest cumulative wealth in portfolio selection but exhibits high variation in its performance, making it less consistent.
- The risk-aware portfolio, while yielding lower cumulative wealth, has more stable results with lower variations.
- By combining these approaches, a balanced portfolio is a middle ground between maximizing returns and minimizing risk. The effectiveness of the UCB1 approach diminishes in volatile markets, where the risk-aware portfolio tends to outperform. The selection of the parameter λ is crucial, as it must be tailored to market conditions to optimize the balance between risk and reward.

We can make several observation by comparing these conclusions made by Ho with the result which we observed:

- In contrast to the Ho analysis (where UCB approach achieves the highest cumulative return) in our case the average portfolio was more profitable.
- The above mentioned features of the considered period is one of the key reasons why our best performing algorithm is different comparing with the analysis prepared in the Ho paper. Moreover, one of the conclusions from the Ho paper supports our analysis, by pointing out that in case of high volatility UBS approach is less effective

Step 10

Now we are going to update our data sample of all the 30 series, we are going to collect data from June - July 2024. we are using the particular time frame as a proxy for the latest market data.

Each Team member was again assigned a different task :

- **Team member A:** again collected the updated data for 15 financial institutions (JPM, WFC, BAC, C, GS, USB, MS, KEY, PNC, COF, AXP, PRU, SCHW, BBT, STI) using Yahoo Finance [2] for the assigned period.
- **Team member B:** again collected the updated data for 15 non-financial institutions (KR, PFE, XOM, WMT,DAL, CSCO, HCP, EQIX, DUK, NFLX, GE, APA, F, REGN, CMS) using Yahoo Finance [2] for the assigned period.
- **Team member C:** was again assigned to combine the data into a Python time series data structure to help further analysis, the member was also assigned to compute the daily return of all the 30 series.

We collected stock data for selected companies from June 2024 to July 2024, you can see in Figure 8:

Date	JPM	WFC	BAC	C	GS	USB	MS	KEY	PNC	COF	...	CSCO
2024-06-04	-0.013180	-0.010785	-0.005015	-0.014373	0.000703	-0.019381	-0.008521	-0.017705	-0.005484	-0.019253	...	0.016506
2024-06-05	-0.009540	-0.000341	0.007056	0.012125	0.014013	0.003080	0.000828	-0.000721	0.002854	0.007436	...	-0.029523
2024-06-06	-0.001774	-0.016190	-0.006506	0.002752	-0.007754	-0.001535	0.001966	0.005051	0.002393	0.006569	...	0.004563
2024-06-07	0.015438	0.010913	0.008109	-0.002583	-0.006964	0.000000	0.002994	0.000000	0.005872	0.025152	...	-0.008436
2024-06-10	-0.001700	-0.007368	-0.002765	-0.005665	-0.002990	-0.006920	-0.007721	-0.018665	-0.020464	0.007868	...	0.000218
2024-06-11	-0.026301	-0.013810	-0.020418	-0.037278	-0.020461	-0.007742	-0.009960	-0.018288	-0.015980	-0.040880	...	-0.001745
2024-06-12	-0.014561	0.002976	0.014153	0.014711	0.009971	0.021587	0.002305	0.023100	0.015907	0.014504	...	-0.003277

Figure 8: Stock data (fragment)

Below we prepared a summary view for the collected stock data over the selected period in Figure 9:

skimpy summary

Data Summary

dataframe	Values
Number of rows	40
Number of columns	15

Data Types

Column Type	Count
float64	15

number

column_name	NA	NA %	mean	sd	p0	p25	p50	p75	p100	hist
KR	0	0	52.01	1.683	49.37	50.5	51.9	53.38	54.86	
PFE	0	0	28.33	1.148	26.6	27.4	27.96	29.1	31.39	
XOM	0	0	113.8	2.482	108.4	112.2	114	115.1	118.8	
WMT	0	0	68.57	1.548	65.82	67.35	68.26	69.92	71.03	
DAL	0	0	47.03	2.628	42.79	44.82	46.99	49.42	50.64	
CSCO	0	0	46.62	0.8967	45.13	45.83	46.79	47.15	48.52	
VRSN	0	0	178.6	2.976	174.2	176.6	178.2	180.2	187	
EQIX	0	0	770.6	19.15	741.9	756.8	764.8	779.1	812.6	
DUK	0	0	103.4	3.165	99.35	100.6	102.6	106.6	109.9	
NFLX	0	0	658.3	20.2	622.6	643	653	677.7	690.7	
GE	0	0	161.9	3.551	155.4	160	161.4	163	172	
APA	0	0	29.5	1.276	27.6	28.6	29.17	30.71	31.97	
F	0	0	12.42	0.9898	10.67	11.82	12.04	13.04	14.33	
REGN	0	0	1049	28.88	989.5	1033	1054	1070	1100	
CMS	0	0	60.2	1.569	58.24	58.97	59.98	61	64.6	

End

Figure 9: Summary of stock data from June 2024 to July 2024

We have calculated correlations of returns among the analysed assets and presented a correlation matrix below in Figure 10:

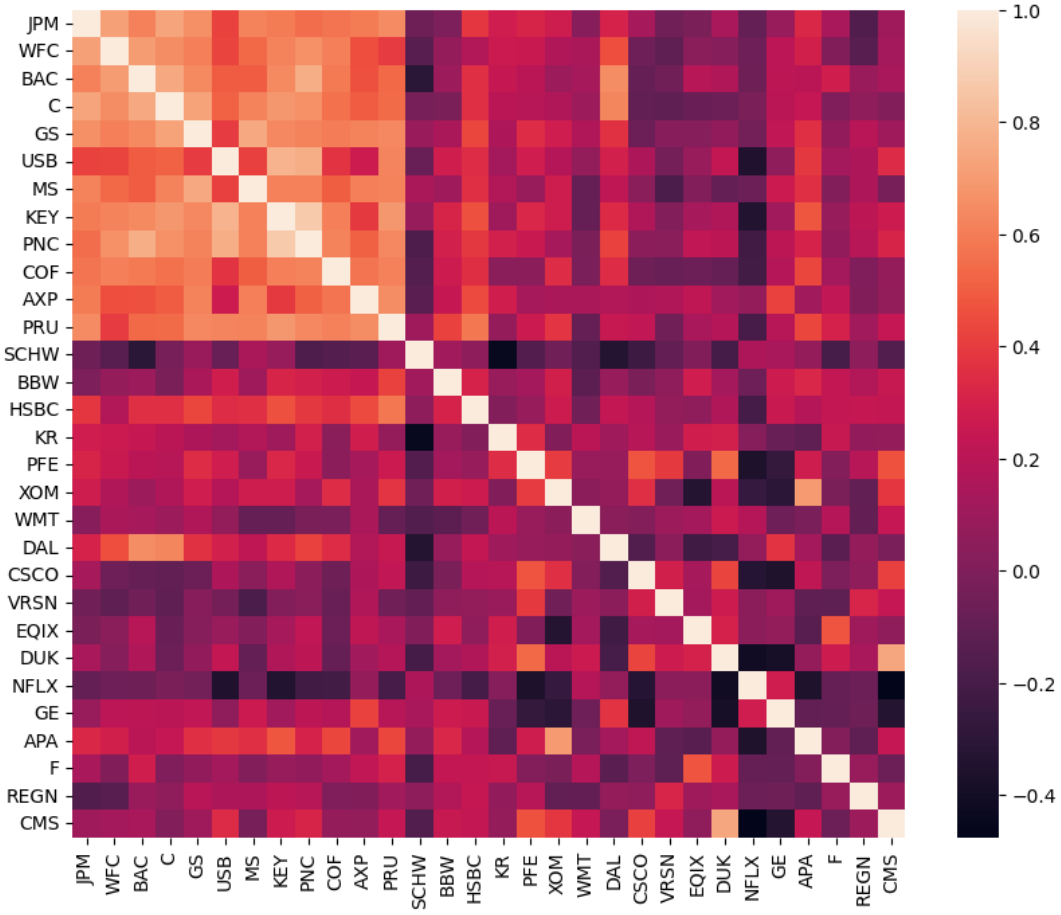

Figure 10: Correlation matrix of returns from June 2024 to July 2024

After that, we prepared the Hierarchical Clustering Dendrogram based on the calculated correlations in Figure 11:

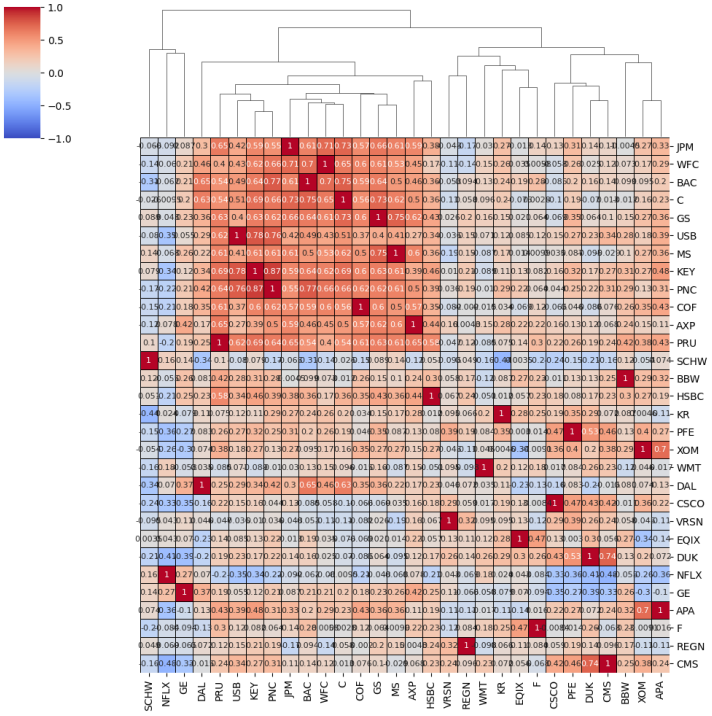

Figure 11: Hierarchical Clustering Dendrogram 2024

Comparing the clustering deprograms prepared for 2008 and 2024 data, we can see that the overall correlation is reduced not only among different clusters but also among companies in the same cluster. That is the expected behavior, because data for 2008 is impacted by steep market decline due to GFC, which led to increased correction for all public industries.

Step 11

In this section we will rerun the results of the stock selection allegorists for new data and compare the results of analysis over 2 different time frames:

- from September 2008 to October 2008
- from June 2024 to July 2024

Below is the result of our analysis for frequencies of optimal actions in Figure 12:

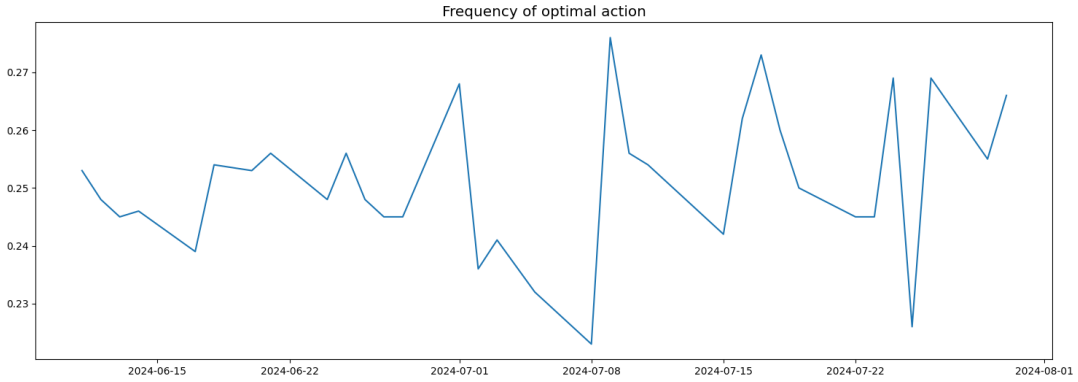

Figure 12: Frequencies of Optimal Actions

The average result is higher than random choice so we can conclude that the algorithm has some predictive power. Finally, we have prepared the forecast of cumulative returns of selected algorithms in Figure 13.

Figure 13: Enter Caption

Analyzing the cumulative return of the $epsilon$-greedy, UCB, and average portfolio methods, we have made several observations:

- The $epsilon$-greedy algorithm demonstrated the highest cumulative return, particularly by the end of the period, reflecting its ability to explore new opportunities while still exploiting high-performing actions. This algorithm's strategy of occasionally choosing actions at random with a small probability ($epsilon$) allowed it to discover profitable opportunities that more conservative approaches might miss. As a result, despite its slightly higher volatility, $epsilon$-greedy managed to accumulate the most rewards by the end of the period.

- The UCB algorithm closely followed $epsilon$-greedy in terms of cumulative return, especially up to the midpoint of the selected period, but it lagged behind at the end. This could be due to UCB's more structured exploration, which may have prevented it from adapting as quickly to new opportunities that the $epsilon$-greedy algorithm, with its inherent randomness, was able to capitalize on.
- On the other hand, the average portfolio method underperformed both $epsilon$-greedy and UCB in terms of cumulative return but demonstrated significantly lower volatility. This suggests that the average portfolio method focused on risk management and stability rather than aggressively pursuing maximum returns. This lower volatility made it a more stable option for investors who prioritize portfolio stability and consistency over high-risk, high-reward strategies. Based on our analysis this stability can be particularly beneficial in volatile market conditions and during sharp decline of the market, where minimizing losses is often more as maximizing gains.

Bibliography

[1] Huo, Xiaoguang & Fu, Feng. (2017). Risk-Aware Multi-Armed Bandit Problem with Application to Portfolio Selection. Royal Society Open Science. 4. 10.1098/rsos.171377.

[2] Yahoo Finance. (2024). Bitcoin USD (BTC-USD) Historical Data. (https://finance.yahoo.com/)

[3] UCB Algorithm. https://people.maths.bris.ac.uk/ maajg/teaching/stochopt/ucb.pdf