# Optimizing Trading Strategies with MDPs

Brian A. Check* and Adam Ctverak †
*Stanford University, Stanford, CA, 94305*

**This paper presents a low-frequency algorithmic trading strategy designed to help firms navigate dynamic market conditions and maximize returns. The approach leverages historical weekly stock data to derive momentum-based indicators such as the Relative Strength Index (RSI) and frames the problem as a Markov Decision Process (MDP), to model state transitions and expected rewards for three possible actions: buy, sell, or hold. Transition and reward matrices are computed directly from limited historical data, allowing a model-based methodology that avoids the need for large datasets required by model-free methods. The resulting optimal policy, obtained through value iteration, is tested on multiple stocks and benchmarked against naive strategies. Aggregate analyses demonstrate that the optimal policy outperforms any fixed-action policy on average, particularly for high-volatility, momentum-driven stocks. The proposed framework identifies conditions under which specific actions yield greater expected returns, providing a data-driven systematic approach for firms trying to maximize their returns.**

## I. Nomenclature

| | | |
|---|---|---|
| $\gamma$ | = | Discount factor |
| $\mathcal{A}$ | = | Action space |
| $MDP$ | = | Markov Decision Process |
| $R(s,a)$ | = | Expected reward |
| $RS$ | = | Relative Strength |
| $RSI$ | = | Relative Strength Index |
| $\pi(s)$ | = | Policy |
| std() | = | Standard Deviation |
| $T(s,a,s')$ | = | Transition probability |
| $U(s)$ | = | Utility |

## II. Problem Statement

The motivation behind this project was to develop a low-frequency algorithmic trading strategy which helps investors make data-driven decisions to maximize returns in dynamic market conditions. This strategy is meaningful only to investment firms with considerable purchasing power as we assume that the actions taken in the stock market will affect the stock price, framing this challenge as a sequential decision making problem. The market agent operates with a fixed position size, focusing only on optimal action (buy, sell, or hold).

Our algorithm leverages weekly time-series price data available from Yahoo Finance and makes decisions based on derived market indicators, such as the Relative Strength Index (RSI), Expected Closing Price, and Volatility. To enhance usability, the program allows users to define the stock ticker of interest and the desired temporal scope for analysis.

After the key metrics are extracted from the data, the algorithm will model this problem as a Markov Decision Process to return optimal policy for state-action pairs. This involves analyzing the stock's historical performance to construct a transition model and a reward function. For real-time applicability, the most recent closing price is fetched to determine the current state of the stock and the program recommends an optimal action; whether to buy, sell, or hold.

*Graduate Student, Aeronautics and Astronautics.
†Graduate Student, Aeronautics and Astronautics.

## III. Approach

The goal of this project was to help firms determine the optimal strategy—buy, sell, or hold—for their stock. With substantial financial resources, these actions can significantly impact stock prices. By analyzing historical stock data, we sought to model how decisions made during one week affect price movements in the following week, enabling companies to adopt strategies that maximize returns.

We framed this problem as a Markov Decision Process, a framework that allowed us to model the relationship between actions and stock price movements in a probabilistic and reward-driven manner. The MDP consisted of three key components: states, which captured the stock's momentum using the Relative Strength Index (RSI); actions, which represented decisions to buy, sell, or hold; and rewards, defined as the percentage price change in the following week. Weekly data was chosen to reduce noise from daily fluctuations and to align with the typical decision-making timeframe for high volume corporate stock transactions.

The Markov Decision Process (MDP) framework is well-suited for this problem as it models the probabilistic nature of stock price movements while incorporating long-term decision-making. In this setup, actions such as buying or selling directly influence the stock's state and price, making action-based transitions integral to capturing the dynamics of the system.

Unlike many environments where model-free methods are preferred due to the complexity of calculating transition ($T$) and reward ($R$) matrices, stock trading operates in a setting with limited data. Even for daily stock data, we are constrained to at most 365 data points per year. This limited dataset makes it feasible to calculate $T$ and $R$ directly, leveraging a model-based approach rather than relying on model-free methods like Q-learning, which typically require much larger datasets to converge effectively.

Value iteration is particularly advantageous in this context because it allows for an efficient computation of the optimal policy by iteratively refining value estimates for each state-action pair. By explicitly using the calculated $T$ and $R$ matrices, value iteration ensures a robust, data-driven solution that is computationally efficient and interpretable. This method leverages the available data fully, avoids the need for extensive exploration, and guarantees convergence to the optimal policy, making it a perfect fit for the structured and limited-data nature of stock trading problems.

### A. State Classification

To build the MDP, we used weekly stock data from 2000 to 2020 from *Yahoo Finance*, a period chosen for its diverse market conditions. This timeframe allowed the model to learn from both bull and bear markets while excluding anomalies caused by the COVID-19 pandemic.

We first classified each week into one of three states based on the stock's RSI, a momentum indicator widely used in financial markets to measure the magnitude of recent price changes. The RSI was calculated from daily stock price data obtained from *Yahoo Finance* and aggregated into weekly averages. Using the daily closing prices, we computed the price changes, separated the gains and losses into two series, and calculated their 14-day rolling averages. The Relative Strength (RS) was then derived as the ratio of average gains to average losses, and the RSI was calculated as:

$$\text{RSI} = 100 - \left(\frac{100}{1 + \text{RS}}\right), \quad \text{where RS} = \frac{\text{Average Gain}}{\text{Average Loss}}$$

The mean and standard deviation for the stock's RSI over the entire set of training data were then calculated, and the weekly RSI values were categorized into three states:

- **Oversold:** $\text{RSI} < \text{mean(RSI)} - \text{std(RSI)}$
- **Neutral:** $\text{mean(RSI)} - \text{std(RSI)} \leq \text{RSI} \leq \text{mean(RSI)} + \text{std(RSI)}$
- **Overbought:** $\text{RSI} > \text{mean(RSI)} + \text{std(RSI)}$

These classifications provided a clear representation of the stock's momentum state for each week.

### B. Action Classification

Since buying or selling a stock directly influences its price within the same week, actions were classified based on deviations in the change from the opening price to the closing price of the week. The expected closing price was calculated using the slope of the stock's price over the previous 12 weeks, creating a baseline trend model. This model accounted for historical trends while incorporating volatility through the variance of past prices. Based on these deviations, each week was labeled as:

- **Buy:** If the actual price increased by at least $+0.5 \times$ std above the expected next price.
- **Sell:** If the actual price decreased by at least $-0.5 \times$ std below the expected next price.

- **Hold:** If the price remained within $\pm 0.25 \times$ std of the expected next price.

Data points that did not fit into these categories were excluded to ensure a clear distinction between actions and to achieve balanced data distribution.

### C. Reward

The reward for each action was defined as the percentage price change in the following week. This backward-looking metric quantified the effectiveness of each action in achieving favorable price movements. Weekly price changes, computed from the historical data, provided a consistent and interpretable measure of rewards.

### D. Transition and Reward Matrix Calculation

Using the state and action classifications, we constructed transition (T) and reward (R) matrices for each stock:

- **T Matrix:** Captures the probabilities of transitioning from one state to another for each action. For example, the probability of transitioning from Oversold to Neutral after a Buy action was estimated by dividing the observed transitions of this type by the total number of Buy actions in the Oversold state. This matrix provided a probabilistic model of how the stock's state evolved over time depending on the action taken.
- **R Matrix:** Records the expected reward (percentage price change) for each state-action pair. For instance, the reward for taking a Buy action in the Oversold state was calculated as the average percentage price change observed in the data for similar occurrences.

### E. Value Iteration

With the $T$ and $R$ matrices constructed, we applied value iteration to determine the optimal policy for each stock. Value iteration is an iterative algorithm that optimizes long-term rewards by selecting actions that maximize expected utility for each state. The optimal action $a^*(s)$ for a state $s$ was determined as:

$$a^*(s) = \text{argmax}_a \left[ R(s,a) + \gamma \sum_{s'} T(s,a,s') U(s') \right]$$

where $\gamma$, the discount factor (set to 0.9), balances immediate and future rewards, and $U(s')$ is the utility of transitioning to state $s'$. The result was an optimal policy for each stock, specifying the best action to take in each of the three states—Oversold, Neutral, and Overbought. Across these three states, there are $3^3 = 27$ possible policies, and value iteration identified the one that maximized expected returns.

This comprehensive approach, combining technical analysis (RSI), statistical modeling (expected next price), and optimization (value iteration), provided a robust framework for identifying strategies that maximize stock returns. By leveraging historical data and probabilistic models, the methodology is well-suited for companies seeking to influence their stock prices strategically.

## IV. Analysis and Results

Our analysis focused on evaluating the effectiveness of the optimal policy derived from value iteration compared to all 27 naive policies across individual stocks and in aggregate for the 20 largest companies by market capitalization. A naive policy refers to a strategy that consistently applies the same action—buy, sell, or hold—for all three market states (oversold, neutral, overbought) without regard to state-specific dynamics or historical data; in other words, a naive policy is the same policy regardless of the stock. By comparing the optimal policy against these naive strategies, we aimed to assess whether our data-driven approach provides a significant advantage. This section outlines the methodology and key findings.

### A. Evaluating Policy Effectiveness

The primary goal of our analysis was to determine how well the optimal policy performed relative to naive strategies. For each stock, we analyzed the returns from Jan. 1, 2023 to Dec. 1, 2024 to assess the impact of our approach.

*1. Policy Return Calculation*

To evaluate the returns for each policy, we calculated the average price change associated with every state-action pair in the testing data. The total return for a policy was determined using the formula:

$$\text{Total Return for Policy} = \sum_{s} R(s, \pi(s)) \times \text{Frequency of } s$$

where $R(s, \pi(s))$ represents the reward (average price change) for taking action $\pi(s)$ in state $s$, and Frequency of $s$ denotes the number of occurrences of state $s$ during the testing period. Summing over all states provided the overall expected return for each policy.

*2. Individual Stock Results*

For example, for NVIDIA (NVDA), our algorithm identified the optimal policy as Sell in the Oversold state, Hold in the Neutral state, and Buy in the Overbought state. The total return achieved by this policy was the highest among all 27 strategies, as shown in Figure 1. Conversely, for Johnson & Johnson (JNJ), a relatively stable stock, the optimal policy underperformed many naive strategies, even resulting in a negative return (see Figure 2).

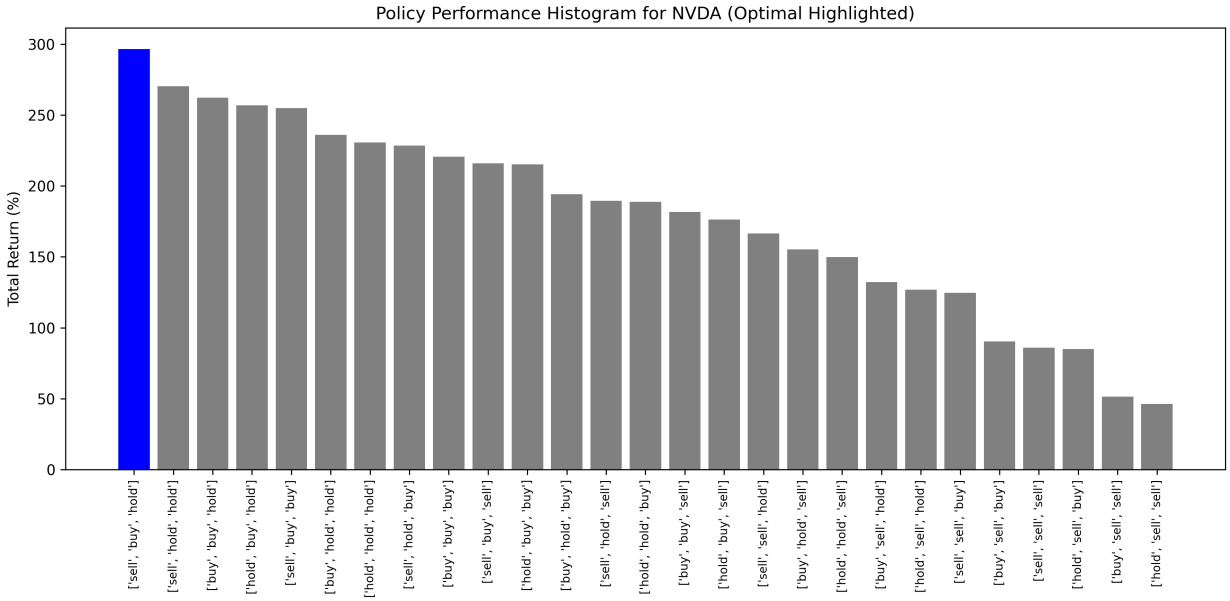


**Fig. 1 Histogram of policy returns for NVIDIA (Optimal Colored Blue)**

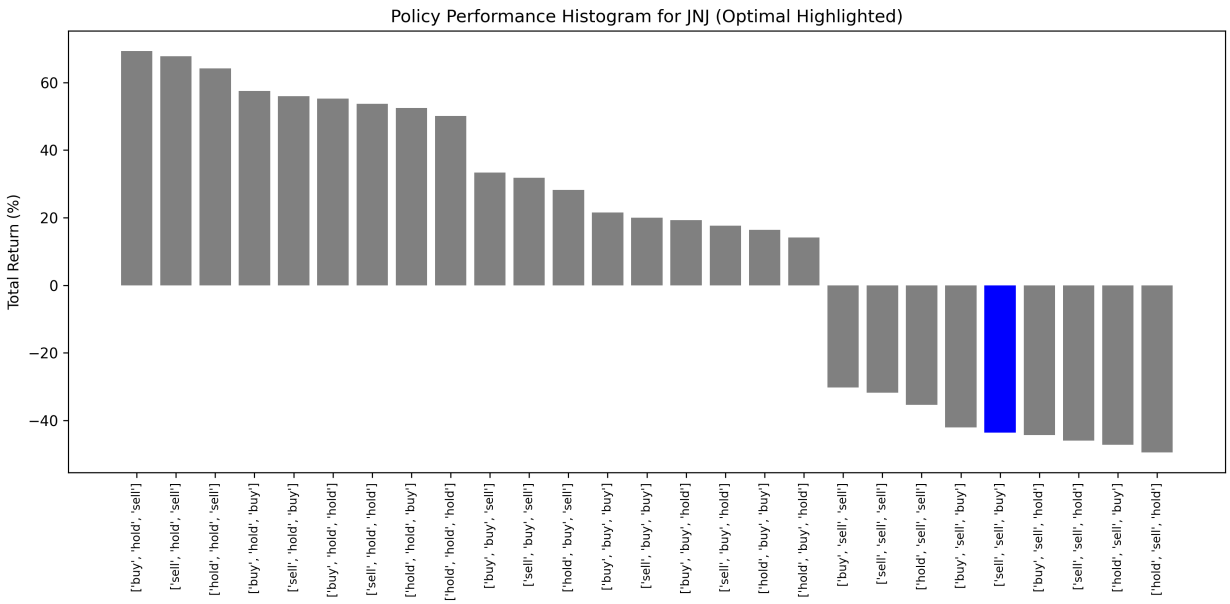


**Fig. 2 Histogram of policy returns for Johnson & Johnson (Optimal Colored Blue)**

### B. Key Observations

- **Performance Variability Across Stocks:** The results highlighted that our policy was highly effective for certain stocks, particularly those characterized by high trading volume and volatility, such as NVIDIA and Tesla. These stocks are more reactive to short-term momentum, making the RSI-based strategy particularly relevant. Conversely, for more stable and low-volatility stocks, such as Johnson & Johnson, the momentum-based approach was less effective, as their prices are less sensitive to short-term speculative actions and more reflective of fundamental earnings.
- **Momentum-Driven Predictability:** Stocks with momentum-driven behavior—often influenced by public sentiment, speculative trading, and future growth expectations—tended to perform well under our strategy. For example, Tesla and NVIDIA consistently ranked among the best-performing stocks under the optimal policy, reinforcing the suitability of RSI-based models for these types of stocks.

### C. Aggregated Results Across the Largest Companies

To examine the generality of our approach, we extended the analysis across the 20 largest companies by market capitalization. For each stock, returns for each policy were normalized by dividing by the maximum return for that stock, enabling cross-stock comparisons on a relative scale. The normalized returns for each policy were then averaged across all 20 stocks.

The aggregate results are presented in Figure 3. The optimal policy outperformed every single naive policy on average, validating the robustness of our approach in the aggregate.

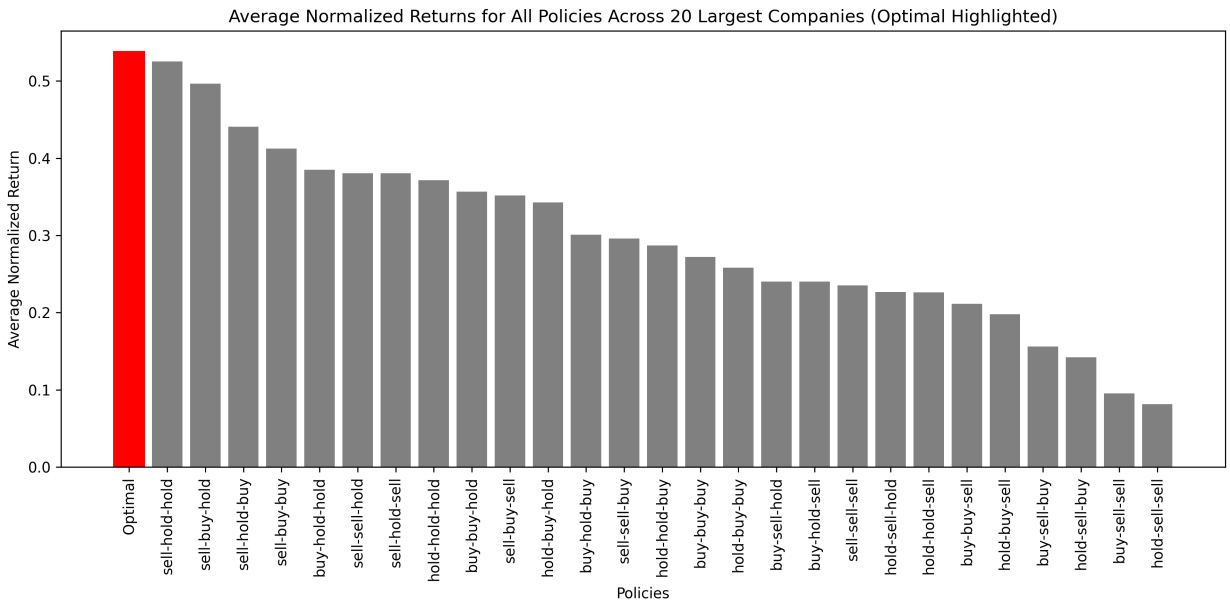


**Fig. 3 Average normalized returns for optimal policy vs. all naive policies across the 20 largest companies**

### D. Implications of the Results

- **Strength of the Optimal Policy:** The results demonstrate that, in aggregate, the optimal policy derived from value iteration provides higher returns than any naive strategy. This finding underscores the utility of combining technical analysis (RSI), statistical modeling (expected next price), and reinforcement learning (value iteration) to derive actionable insights.
- **Limitations for Stable Stocks:** Despite the overall success, the strategy's limitations for stable, low-volatility stocks highlight the importance of tailoring models to stock characteristics. Incorporating additional features, such as fundamental metrics or industry-specific factors, may enhance performance for such stocks.

This comprehensive analysis demonstrates that the RSI-based strategy is particularly effective for high-volatility, momentum-driven stocks. Additionally, the aggregate results validate the robustness of our approach for large-scale decision-making, providing a promising framework for companies to strategically influence their share prices.

## V. Related Works

Hult and Kiessling (2010) also use Markov chains to represent the state and evolution of financial instruments to evaluate optimal strategies for buying and selling. In their analysis, they implement a value-iteration algorithm to numerically find an optimal strategy for high-frequency trading in the foreign exchange (forex) market [1]. The central focus of their paper is the limit order book, a list of all buy and sell offers, represented by price and quantity. They model the limit order book as a high-dimensional Markov chain where each state reflects the current volumes of buy and sell orders at discrete price levels. Transitions happen due to events like new orders, cancellations, and trade executions. To contrast, our approach simplifies the state space significantly, only focusing on derived market indicators as opposed to modeling the entire order book. While Hult and Kiessling address milli-second decision-making, we built our program for swing trading on a weekly basis. Lastly, while their action space is $\mathcal{A}$ = [limit buy order, limit sell order, market buy order, market sell order, cancel], our action space only considers market orders.

Rundo et al. (2019) propose an innovative approach based on the combined usage of Long Short-Term Memory (LSTM) networks with Markov models to perform careful stock price prediction for high-frequency trading [2]. The authors claim that traditional statistical methods and soft computing techniques are limited due to their inflexibility in dynamic variability. Modern machine learning approaches, such as the LSTM model, are shown to be more accurate than other machine learning models. Their methodology integrates two LSTM pipelines: one for forecasting mid- to long-term market trends and another for predicting daily stock closing prices, corrected using Markov property-based

assumptions. Their experiments use historical data from high-volatility stocks, demonstrating significant improvements in prediction reliability and trading profitability compared to standalone LSTM approaches. To contrast their approach, we use simpler methods for trend prediction and focus only on price regression. The action space described in the paper by Rundo et al. includes operations such as LONG, SHORT, and null trades, which correlates with our simplified approach of $\mathcal{A}$ = [buy, sell, hold].

## VI. Conclusion

In this final project, we developed and evaluated a data-driven trading strategy formulated as an MDP, using RSI-based states and action-dependent transitions to identify optimal policies for dynamic market conditions. The value iteration approach used historical data, outperformed naive strategies overall and showed particular strength on volatile, momentum-sensitive stocks. While the method proved less effective for stable, low-volatility equities, the results highlight the potential of combining technical analysis, statistical modeling, and reinforcement learning to inform trading decisions under uncertainty. Future work may include other fundamental metrics, alternative modeling pipelines, or more frequent data updates to improve performance for a broader range of market scenarios.

## VII. Division of Labor

- Brian Check: Contributed to the development of the optimal policy selection code and wrote the policy evaluation code. Authored the "Approach" and "Analysis and Results" sections of the paper.
- Adam Ctverak: Coded the value iteration pipeline and added the optimal policy based on the current state. His contribution to the paper includes the Problem Statement, Literature Review, and Conclusion.

## Appendix (Code)

Code is contained in the following github repository: https://github.com/b9check/stockmdp


## Acknowledgments

The authors would like to thank professor Mykel J. Kochenderfer and the teaching assistants of AA 228 for their guidance and passion for teaching.


## References


[1] Hult, H., and Kiessling, J., "Algorithmic trading with Markov chains," 2010.

[2] Rundo, F., Trenta, F., Di Stallo, A. L., and Battiato, S., "Advanced Markov-based machine learning framework for making adaptive trading system," *Computation*, Vol. 7, No. 1, 2019, p. 4.