A comparative study on four automated stock-trading agents

Hongwei Li[*] Menglin Chen[†] Jun Yang [‡]
The Hong Kong University of Science and Technology

Abstract

Trading algorithm is for long being a hot spot of computer science where people cast many energy and money onto. In this project report, we implemented three existing trading algorithm plus one of our improved ones and evaluated their performance under different patterns of stock price trends with returns and Sharpe ratio. However, the objective of this project is not about coming up with a champion trading strategy but aiming at how to compare these strategies through an analytical way, which benefits later research.

Keywords: trading algorithm, e-commerce, algorithm evaluation

1 Introduction

The advances in computer science magnificently change the way how ordinary equity trading works. Telephone and fax are soon replaced by the modernized computers, and the entire trading process is migrated into a so-called electronic fashion. This greatly accelerates the market efficiency and increases the volume of equity flowing around in every seconds. On the other hand, computers not only play an aide tool next to human traders, but sometimes, as many computer scientist hopes, can manage the trading itself, placing or withdrawing the orders and most importantly locking the profits. It is no doubt that with its computation power, computer can be smart enough to win over human traders if it is programmed with an algorithm, which realizes the principles of the market and predicts the future price precisely. The burden of looking for such a dreaming algorithm is shed onto our computer scientist, whether this algorithm or even a math model does exist, and if the algorithm is doable.

It has been a long time since the trading algorithm became the one of most active research areas in the computer science. One reason for its popularity undoubtedly is because this problem is the most rewarding one among all computer science problems. A competent algorithm is at fact a key to the door of an invaluable treasure. However people badly pore over this problem thousands of days and nights, and cast tons of cash upon it, still there is no really working algorithm which can prove itself in the real world.At least, to our best knowledge, there is not. Probably we are wrong, but surely if there is, the inventor must have not disclosed the algorithm and hence is able to enjoy the proceeds somewhere and somehow.

The algorithms published in the academic domain, though not tough enough, but are significant in the scope of research, for they have carried out the computation theories into real practice. Prior research includes a variety of approaches in the stock trading. Reinforcement learning has be studied in [Chan and Shelton 2001] and future employed in the automated trading algorithm in [Sherstov and Stone 2004]. Other approaches, like market-making [Sherstov and Stone 2004; Chan and Shelton 2001], reverse strategy and VWAP [Yu and Stone 2003; Feng et al. 2004] had proved their capability in simulating practise. An early brief overview of trading algorithms can be found in [Kearns 2004].

In this project, we have implemented three proposed algorithms, plus one of our improved one. Nevertheless, the objective of this project is not about the implementation detail and improvement of the current algorithms, but to conduct an empirical analysis of these algorithms by comparing their performance in the same environment. The adopted grading criteria is the Sharpe ratio, a reliable measure of ”the statistical significance of earnings and the trade-off between risk and return” [Kearns and Ortiz 2003]. It is computed by empirical daily average of returns divided by the standard deviation, a quantity most modern fund manager seek to maximize [Moody and Saffell 2001].

2 Overview

The subjects of our study contain four algorithms and the historical prices of stocks traded at the Shanghai Stock Exchange. We used minute-based series stock data, each of whose entry includes time (in the format of YY-MM-DD-HH-MM), price, and volume in that minute.

The reason for choosing the minute-based price data is we can focus on the short-term profit of the stock. As the time span gets larger, the more unknown and out-of-control influence is likely to walk in, like rumors, bad/good business performance of the company, which will largely determines the stock prices, instead of market trading activities. And the pattern of the long-term stock price tends to be monotonic increase/decrease, which benefits the most primitive buy-and-hold strategy, and turn other algorithms down. In summary, We do not think long-term evaluation is either simple or interesting to do in the scope of a course project.

We allow each simulator (the agent equipped with the trading algorithm) to perform three operations, buying, selling and shorting a particular stock with certain volume (constrained to an upper bound). To simplify the algorithm, the cash and stock holding balance of the simulator can be negative, that means, in the terms of purchase, simulator can borrow money without any interest during the simulation for purchasing any number of stocks it likes. This loan will be deduced from the total value of the simulator at the end of the simulation. Correspondingly, simulator can also borrow any number of stocks to sell as long as it returns the same amount of stock at the end. The current total value of the simulator can be described as following.

$$\text{total value} = \text{cash} + \text{holdings} * \text{current stock price}$$

We evaluate the performance of trading algorithms with Sharpe ratio by dividing the average return of each simulator with the standard deviation.

[*]e-mail: lihw@cse.ust.hk
[†]e-mail:menglin@cse.ust.hk
[‡]yjrobin@cse.ust.hk

3 The Trading Algorithms

The algorithms we are implemented and studied are Market-marking strategy, Reverse market-making strategy [Feng et al. 2004], Trend-following strategy based on regression-based price prediction [Sherstov and Stone 2004] and our Improved trending-following strategy.

3.1 Market-making strategy

The market-making algorithm exploits the volatility of the stock price. During a certain period, this algorithm assumes the stock price will fluctuate around a certain price axis. Without the predication of future price movement, it follows a simple strategy, putting a pair of buy and sell orders of the same volume on a single stock simultaneously. The price of buy order is likely lower than the current price, while the price of the sell order will be higher. Whenever the stock price goes up above the sell order's price or drops down below the buy order prices, the orders are executed and profits is locked as long as the balance of sell and buy orders are maintained.

The crucial problem of this strategy is how to define the put and call price. If the put and call price are far away from current stock price, it is unlikely that the orders will be matched and all efforts end up in vain. A natural choice is to add a little margin to the order price. For example, we can set the buy price 0.02 lower than the current price and symmetrically set the sell price 0.02 higher. Another issue is the volume of the order. To avoid the influence of the order on the market, we usually set an upper limit for the volume of order, namely 2000. Thus, intervene effect brought up by our virtual trading activity is small enough to be ignored and historical stock price is still reliable to be the estimation basis. Algorithm 1 describes the algorithm in a pseudo-code manner.

Algorithm 1 Market_Making_Strategy

```
while time permits do
    buyReferencePrice $\leftarrow$ getBuyOrderPrice(n) + 0.001;
    placeOrder(BUY, buyReferencePrice, volume);
    sellReferecePrice $\leftarrow$ getSellOrderPrice(n) - 0.001;
    placeOrder(SELL, sellReferencePrice, volume);
end while
```

The above naive algorithm can be improved future when taking the holding position into account. For example, when we are at large positive holding, we may lower the price of sell orders such that it can be more likely met than buy orders and thus alleviate our large holding problem. Algorithm 2 illustrates this idea.

$priceEncouragement$ and $volAlteration$ can be different from stock to stock. In our experiment, we set $priceEncouragement$ to be 0.0001, $volAlteration$ to be 0.001 and volume to be 100 shares. It is worthy to note that the placed order will not be executed at once. To address this problem, the program maintains an order buffer caching all not-yet-executed orders. Every time the simulator receives a stock price, it checks the buffer and execute all cached orders which meet the current price. The completed or starved (older than one day) orders will wiped out from buffer immediately once noticed.

3.2 The Reverse Strategy

The reverse strategy flips the strategy of Market-making algorithm, that is when stock prices goes up, it issues the sell order and when price goes down, it issues the buy order. The reason that reverse strategy makes profits, as the author argues, is because that "stock market prices are not constant and in fact undergo frequency changes in direction, rather than moving consistently in one direction". [Yu and Stone 2003]. The algorithm is nothing different but changes the position of two order actions in the Algorithm 2;see Algorithm 3, so we do not bother listing it here. A good point of the reverse strategy is it does not need to maintain an order buffer as the order placed by it can be executed at once.

Algorithm 2 Improved_Market_Making_Strategy

```
while time permits do
    buyReferencePrice $\leftarrow$ getBuyOrderPrice(n) + 0.001;
    sellReferecePrice $\leftarrow$ getSellOrderPrice(n) - 0.001;
    currentPosition $\leftarrow$ getAgentCurrentPosition()
    if currentPosition $< 0$ then
        buyReferencePrice $\leftarrow$ buyReferencePrice - currentPosition
        * priceEncouragement
    else if currentPosition $> 0$ then
        sellReferencePrice $\leftarrow$ sellReferencePrice - currentPosition
        * priceEncouragement
    end if
    buyVolume $\leftarrow$ volume * (1 - currentPosition * volAlteration)
    sellVolume $\leftarrow$ volume * (1 + currentPosition * volAlteration)
    placeOrder(BUY, buyReferencePrice, buyVolume);
    placeOrder(SELL, sellREferencePrice, sellVolume);
end while
```

Algorithm 3 Reverse_Market_Making_Strategy

```
while time permits do
    volume $\leftarrow$ 100;
    buyVolume $\leftarrow$ volume * (1-share * volAlteration);
    sellVolume $\leftarrow$ volume * (1+share * volAlteration);
    lastPrice $\leftarrow$ getLastPrice();
    currentPrice $\leftarrow$ getCurrentPrice();
    if currentPrice $>$ lastPrice then
        placeOrder(SELL, currentPrice, sellVolume);
    else
        placeOrder(BUY, currentPrice, buyVolume);
    end if
end while
```

3.3 The Trend-following Strategy

The Trending-following algorithm issues the buy and sell orders by observing the historical price trend. Basically, it puts the sell order if the price is falling and puts the buy order if the price is rising.

The heart of the Trend-following agent is about the predication of the price trend. It envisions the future price based on the observation of historical prices. The first and second derivatives, P' and P'', about the price in a certain period are computed as the basic measurement of the price movement. In [Sherstov and Stone 2004], the author chose 3600 and 400 seconds to be the interval for computing P' and P'' respectively. The pseudo-code is given in Algorithm 4.

3.4 Our trading algorithm

The key idea of previous Trend-following algorithm is based on equal-weighted linear regression, but this equal-weighted algorithm is not suitable for stock market. Intuitively, the older the price is, the less influence it imposes on today's price. It motivates us to come up with a more reasonable algorithm of linear regression used in trend-following strategy. Here we proposed three methods, All of which emphasize the impact of recent prices.

Algorithm 4 Trend_Following

```
Require: sell-price ← max{last-price, predicated-last-price}
Require: buy-price ← min{last-price, predicated-last-price}
  if $P' > 0$ and $P'' > 0$ then
    return "Buy 75 shares at buy-price"
  else if $P' < 0$ and $P'' < 0$ then
    return "SELL 75 shares at sell-price"
  end if
```

The first method uses linear-weight; e.g., $\omega = i$, where i is the number of days counting from the first day. Obviously, the latest price will certainly have the largest weight; the second method uses quadratic weight, e.g., i^2; the third method uses exponential weight, e.g., e^i.

4 Comparison and Analysis

4.1 Individual comparison

In this comparison study, we deliberately use a set of representative minute-based daily price data to test out the performance of the four algorithms. The representative data, we believe, have covered all the price trend patterns in the reality, the monotonic increase/decrease, the obvious fluctuation, the zig-zag and mixtures of aforementioned patterns. Figure 1 showcases the miniatures of these price data with labels annotating different patterns. Thus, by examining the daily returns of algorithms given different pattern of data, we can learn the efficiency of these algorithms under different conditions and their overall efficiencies, which can be a reasonable estimation about how the strategy will work in the practice.

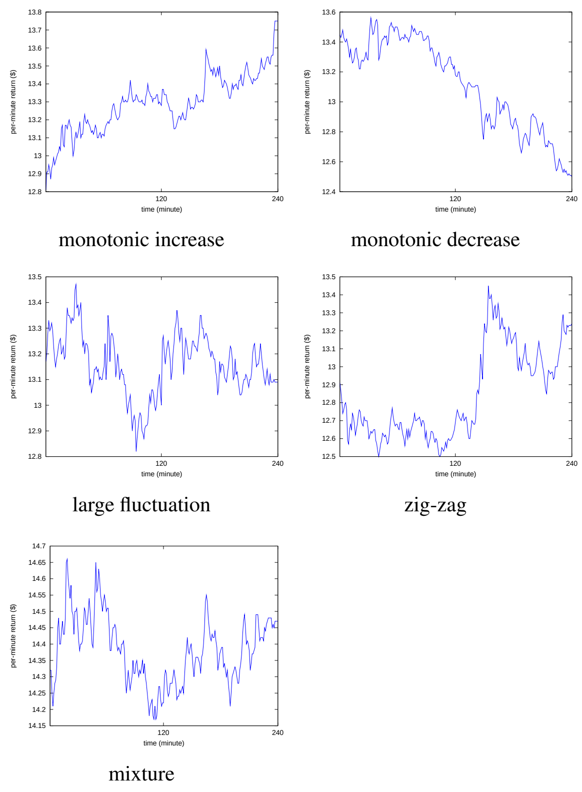


Figure 1: Different price trend patterns.

Table 1 reports the resulting profit/loss of market-making, reverse market-making , trend following and our strategies in the individual simulations on different price trend patterns. Each row in the table gives out the final return, the average the per-half-an-hour return, the standard deviation of per-half-an-hour return and Sharpe ratio.

	Final	Mean	STD	Sharpe ratio
Market-making				
Monotonic increase	-223.04	-27.88	96.36	0.00073
Monotonic decrease	-218	-27.25	80.03	0.00087
Large fluctuation	310	38.75	94.68	0.00074
Zig-zag	193.04	24.13	33.71	0.00208
Mixture	207.04	25.88	24.28	0.00288
Reverse market-making				
Monotonic increase	-205.04	-25.63	84.43	0.00083
Monotonic decrease	310	38.75	65.8	0.00106
Large fluctuation	440	55	169.17	0.00041
Zig-zag	348	43.5	26.08	0.00268
Mixture	245.04	30.63	23.8	0.00294
Trend-following				
Monotonic increase	351.04	43.88	124.76	0.00056
Monotonic decrease	159.04	19.88	253.25	0.00028
Large fluctuation	38.00	4.75	315.77	0.00022
Zig-zag	-1839.04	-229.88	302.79	0.00023
Mixture	-616.00	-77.00	191.95	0.00036
Improved trend-following with $\omega = i$				
Monotonic increase	620	77.50	158.21	0.00046
Monotonic decrease	686	85.75	212.46	0.00033
Large fluctuation	-653.04	-81.63	261.22	0.00027
Zig-zag	-1573.04	-196.63	209.53	0.00033
Mixture	-471.04	-58.88	148.71	0.00047
Improved trend-following with $\omega = i^2$				
Monotonic increase	606.00	75.75	153.42	0.00046
Monotonic decrease	578.00	72.25	210.62	0.00033
Large fluctuation	-676.00	-84.50	252.96	0.00028
Zig-zag	-679.04	-84.88	230.28	0.00030
Mixture	-226.00	-28.25	79.12	0.00088
Improved trend-following with $\omega = e^i$				
Monotonic increase	118	14.75	66.96	0.00105
Monotonic decrease	251.04	31.38	76.98	0.00091
Large fluctuation	14	1.75	102.59	0.00068
Zig-zag	33.04	4.13	144.91	0.00048
Mixture	-237.04	-29.63	31.60	0.00222

Table 1: The performance of each strategy under different price trend patterns.

Market-making strategy In our experiment, we found that Market-making strategy is more suitable for large fluctuation pattern than any other patterns in terms of Sharpe ratio. This indicates Market-making strategy can guarantee a more stable income than other strategies.

Reverse market-making strategy The mean profit value of this method is always the highest, except when price is in monotonic increase or monotonic decrease. It is because of the nature of the algorithm itself. Basically, it assumes the stock price will bounce back when it decreases for a certain while and it will drop down when it keeps increasing. Correspondingly, strategy buys the stock when the price is going down and sells in the opposite case, hoping the stock price will turn around after a certain time. Usually market is like this, and thus this strategy makes money. But when the price is in monotonic increase or decrease, this method will fail to achieve profiting and it will lose more and more if the trend continues.

Trend-following strategy In monotonic increase/decrease pattern, it beats the other strategies(Our algorithm is a kind of Trend-following strategy) thanks to its insensitivity to small price fluctuation. In other scenarios, as Trend-following strategy is based on the linear regression, which can not quickly find the peak/valley of the price, especially after the price is increasing for a short period of time, it definitely loses money, as shown in the zig-zag and large fluctuation patterns.

Our strategy In fact, the three algorithms using different method to assign the weight are reflecting the sensitivity to the price of the strategy. One interesting phenomena we observed from the experiment results, as the weight goes larger, e.g, e^i, the Improved trend-following algorithm tends to be more insensitivity to large fluctuations, as shown in the last three rows of Table 1, though on the other hand, it performed worse than other Trend-following with smaller weights. And no matter which algorithm we use to do the linear regression, trend following is always good at the monotonic increasing scenario.

4.2 Joint comparison

Comparing to the individual comparison, this joint comparison was using another 10 days of price data but without knowing the specific price pattern. We were focusing on the overall profit of each strategy after ten days of simulation by computing their Sharpe ratios, rather than individual daily return. This is because, in the real trading world, a consistent albeit small profit earning strategy is more desirable than the one with large variability, so in the this test, the winner is the strategy with the largest Sharpe ratio.

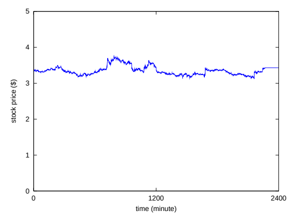
Table 2 shows the results of per-half-an-hour returns of each strategy during 10 days and Figure 2 plots the profit/loss.

Method	Final	Mean	STD	Sharpe ratio
Market-making	604.8	7.56	56.07	0.00125
Reverse market-making	1348.8	16.86	64.77	0.00108
Trend-following	-1552.80	-19.41	113.81	0.00062
Advanced TF,w=i	-1216.20	-15.21	115.87	0.00060
Advanced TF,w=i*i	-406.40	-5.08	115.66	0.00061
Advanced TF,w=exp(i)	-344.80	-4.31	37.87	0.00185

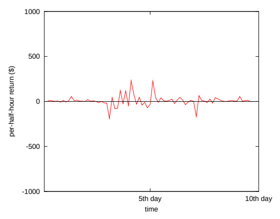
Table 2: The overall performance of each strategy during 10 days.

In conclusion, Market-making is the strategy with highest Sharpe ratio in this experiment; the runner-up is Reverse market-making strategy; then follows Advanced Trend-following strategy with weight as $\omega = e^i$. By taking the average return into account, we think that Reverse market-making is the best trading algorithm due to our experiment result.

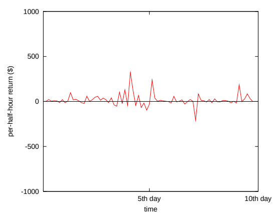
The reason that the Reverse market-making strategy outperforms other algorithms in the join comparison, is that the algorithm is truly tailor for the usual behavior of the practical markets. Though simple, it is elegant yet effective. For future works, we are consider to shoot up more rigorous and careful evaluation for Reverser market-making to help identifying (there must be some) some weakness of it and see if we can fill the hole.



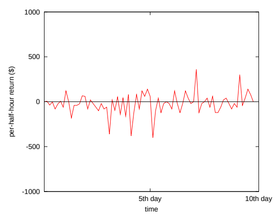
Stock price over 10 days



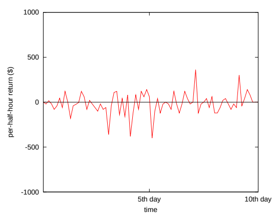
Market-making



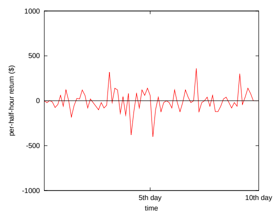
Reverse market-making



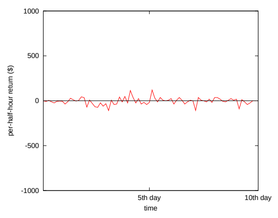
Trend following



Our method($\omega = 1$)



Our method($\omega = i^2$)



Our method($\omega = e^i$)

Figure 2: The profit/loss during 10 days of each strategy.

Acknowledgments

Here we would like to thank all authors of cited papers to motivate us to start this project; thanks Prof. Skiena for carefully proofreading and useful comments. For other unacknowledged helpers, our gratefulness knows no bounds.

References

CHAN, N. T., AND SHELTON, C. 2001. An electronic market-maker. Tech. rep., 04.

DAS, S., AND DS, S., 2003. Intelligent market-making in artificial financial markets.

FENG, Y., YU, R., AND STONE, P. 2004. Two stock-trading agents: Market making and technical analysis. In *Volume 3048 of Lecture Notes in Artificial Intelligence*, Springer Verlag, 18–36.

2008. Google Finance. WWW page. `http://finance.google.com/finance`.

KEARNS, M., AND ORTIZ, L. 2003. The penn-lehman automated trading project. *IEEE Intelligent Systems 18*, 22–31.

KEARNS, M., 2004. WWW: Common stock-traidng strategies. WWW page. `http://www.cis.upenn.edu/~mkearns/projects/strategies.html`.

MOODY, J., AND SAFFELL, M. 2001. Learning to trade via direct reinforcement. *Neural Networks, IEEE Transactions on 12*, 4 (Jul), 875–889.

SHERSTOV, E. A., AND STONE, P. 2004. Three automated stock-trading agents: A comparative study. In *AAMAS 2004 Workshop on Agent Mediated Electronic Commerce VI*.

YU, R., AND STONE, P. 2003. Performance analysis of a counter-intuitive automated stock-trading agent. In *ICEC '03: Proceedings of the 5th international conference on Electronic commerce*, ACM, New York, NY, USA, 40–46.