International Journal on Advanced Science Engineering Information Technology

Vol.10 (2020) No. 1
ISSN: 2088-5334

Univariate Financial Time Series Prediction using Clonal Selection Algorithm

Ammar Azlan[#1], Yuhanis Yusof[#2], Mohamad Farhan Mohamad Mohsin[#3]

[#]School of Computing, Universiti Utara Malaysia, 06010 Sintok, Kedah, Malaysia
E-mail: [1]ammar_azlan1@ahsgs.uum.edu.my; [2]yuhanis@uum.edu.my; [3]farhan@uum.edu.my

***Abstract*— The ability to predict the financial market is beneficial not only to the individual but also to the organization and country. It is not only beneficial in terms of financial but also in terms of making a short-term and long-term decision. This paper presents an experimental study to perform univariate financial time series prediction using a clonal selection algorithm (CSA). CSA is an optimization algorithm that is based on clonal selection theory. It is a subset of the artificial immune system, a class of evolutionary algorithms inspired by the immune system of a vertebrate. Since CSA is an optimization algorithm, the univariate financial time series prediction problem was modeled into an optimization problem using a weighted regression model. CSA was used to search for the optimal set of weights for the regression model to generate prediction with the lowest error. Three data sets from the financial market were chosen for the experiments of this study namely S&P500 price, Gold price, and EUR-USD exchange rate. The performance of CSA is measured using RMSE. The value of RMSE for a problem is related to the maximum and minimum value of the data set. Therefore, the results were not compared to other data sets. Instead, it is compared to the range of values of the data sets. The result of the experiments shows that CSA can make decent predictions for financial time series despite being inferior to ARIMA. Hence, this finding implies that CSA can be implemented on a univariate financial time series prediction problem given that the problem is modeled as an optimization problem.**

***Keywords*—artificial immune system; clonal selection algorithm; financial time series; prediction; univariate.**

I. Introduction

The financial market is a marketplace where products under the financial sector are traded. These products include capital, commodity, money, derivatives, futures, insurance, and foreign exchange. Stock and bond are products traded under the capital market [1]. Some financial markets are small and involve few activities while some financial markets like the New York Stock Exchange (NYSE) deal with many transactions worth trillions of dollars daily [2]. The high profitability of the financial market has attracted many traders and investors to beat the market by forecasting its movement. Hence, many studies have come up with forecasting techniques and models to produce the highest forecasting accuracy with the intention to improve profit and reduce risk in investment.

Kwon et al. [3] conducted research to compare a machine learning model called a neural network with a linear regression model called ordinary least square. Their result using a t-test shows that machine learning performed significantly better than linear regression. This finding supports an earlier study by Qian [4] which compared the prediction precision of the traditional time series model with the machine learning model. Qian mentioned that the performance of traditional time series models like ARIMA and GARCH still requires much improvement despite being researched for a long time. On the other hand, Qian added that machine learning displays incredible performance in regression, classification, and prediction of financial time series despite being emerged recently. A result of an experiment using real data sets proved that the precision of machine learning far surpasses traditional models.

Machine learning is an application of artificial intelligence that enables systems to solve problems from a given set of rules and examples instead of explicitly programming the systems with a specific solution. It has been applied in many areas such as marketing, banking, healthcare, business, stock market, and weather [5]. Machine learning approaches are consistently among the top performers for financial market price prediction. One of the widely used approaches is time series forecasting, which applies a model to predict future values based on previously observed values [6]. However, better results were obtained from hybrid forecasting models [5]. Evolutionary algorithm is one of the approaches that has been successfully applied in financial market prediction.

The evolutionary algorithm (EA) is a subset of artificial intelligence that is inspired by biological evolution. It mimics the mechanism of biological evolution such as reproduction, mutation, recombination, and selection. Possible solutions to the optimization problem are represented as an element in a population while the quality of the solutions is represented as the fitness function. The population evolves through a series of mechanisms mentioned earlier. An example of a well-known EA is the genetic algorithm (GA). GA is an optimization method that is inspired by the process of natural selection from Darwin's theory of evolution. A solution, which is represented by a chromosome, evolves through a series of operation consists of selection, crossover, and mutation. Some studies implement GA on time series forecasting [7]–[9]. Despite its performance, GA suffers from complexity, given the many operations it must perform. A simpler alternative to GA is an artificial immune system.

The artificial immune system (AIS) is a computational intelligence approach that is inspired by the immune system of a vertebrate. It can self-organize, learn, memorize, and adapt besides having the characteristics of robustness and scalability [10]. It can be categorized into five types which are Clonal Selection Algorithm (CSA), Negative Selection Algorithm (NSA), Immune Network Algorithm (INA), Danger Theory Algorithm (DTA), and Dendritic Cell Algorithm (DCA) [11]. These algorithms were inspired by theories from immunology namely clonal selection theory, negative selection theory, danger theory, and artificial immune network theory. It gained popularity by efficiently solving numerous optimization problems [12] and has been applied in numerous domains such as computer security [13], optimization [14], disease diagnosis [15] and bankruptcy prediction [15]. Most of its achievements were contributed by CSA [16].

CSA or sometimes referred to as CLONALG (CLONal selection ALGorithm) is a model inspired by clonal selection theory from the immunology domain. This theory indicates that when a foreign substance called antigen is detected, it induces an immune response in the body called an antibody. The best match between an antigen and an antibody which is called affinity will cause the antibody to multiply in order to counteract the foreign invasion rapidly. This multiplication is performed through a process called cloning. Next, the cloned antibodies hypermutate in order to increase their receptor population [17]. CSA consists of four components which are selection, clonal expansion, hypermutation, and re-selection [18]. Fig. 1 illustrates the flow chart of CSA [19].

Like other EA, CSA starts its operation with an initial population. A set of possible solutions is generated randomly to create an initial population. This initial population is represented as a population of antibodies. Then the population is evaluated against the antigen. The degree of matching between an antibody and an antigen is called affinity. Based on the affinity, a subset of the initial population is selected and cloned to a specific cloning rate. This will generate another population called the clone population. Next, the clone population will maturate through the process of hypermutation, where some elements in the population will change its value according to a pre-set probability. Then, the mature clone population is re-evaluated and re-selected according to the affinity. The process is iterated until the goal is achieved. Usually the goal can be a minimum objective value, a convergence of the objective value (no or little change over certain period), or a pre-set number of iterations.

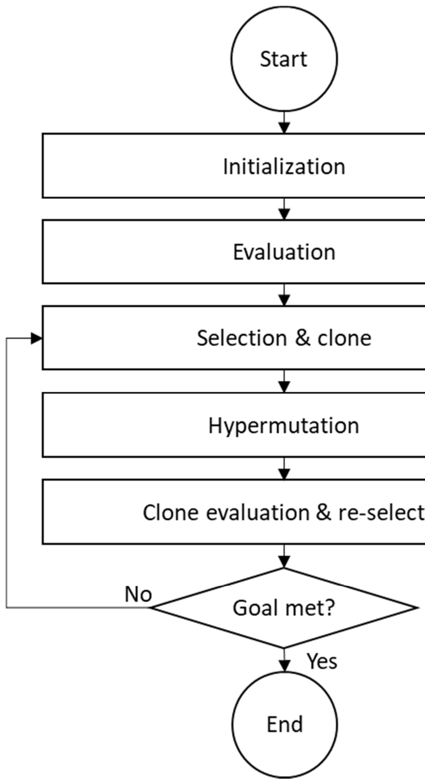

Fig. 1 Flow Chart of CSA

In this study, an evolutionary algorithm inspired by the clonal selection theory called CSA is proposed to solve the problem of univariate financial time series prediction. This algorithm is chosen because based on a survey by Luo and Lin [20], CSA has been applied to many optimization problems, including constrained optimization [21], combinatorial optimization [22], and nonlinear optimization [23]. Furthermore, according to Hu, Sun, Nie, Li, and Liu [24], CSA is less complicated than GA because it does not have crossover operation which reduces the computational cost. They also added that CSA is better than Ant Colony Optimization for continuous problems.

Since CSA is an optimization algorithm, the prediction problem will be modeled as an optimization problem. This is achieved by optimizing the weights of a regression model. First, the actual time series is plotted. Next, the predicted time series generated by the regression model with initial weight is plotted. Then, CSA will search for the optimal set of weights that returns the minimal objective value which is RMSE. Finally, the model with an optimized set of weight is tested on a new data set that has not been exposed to the model during the training phase. The performance of CSA on univariate financial time series prediction problem is evaluated by plotting line graphs and calculating RMSE.

The rest of this paper is organized as follows: First, Section II discusses materials and methods. It covers the preparation of the data sets used in the experiments from data collection to data pre-processing. It also covers the parameter setting of CSA and method to evaluate the experiments. The section ends with the specification of the machine used to perform the experiments. Second, Section III presents the result and discussion. It includes all results from all data sets. Then, the section ends with a discussion based on the findings. Third, Section IV presents the conclusion. It is a short section that concludes the findings of this study, provides the implications of the findings, and suggestion for possible future works.

II. Materials and Method

A. Data Sets Preparation

1) Data Collection: To perform this experiment, historical data from three financial markets were retrieved from Yahoo Finance namely S&P500 price index, Gold price, and EUR-USD exchange rate. The data sets consist of seven variables which are labeled as Date for transaction date, Open for opening price on the Date, High for the highest price on the Date, Low for the lowest price on the Date, Close for the closing price on the Date, Adj Close for adjusted closing price on the Date, and Volume for the transaction volume on the Date. The data sets also consist of 2,516 observations for each record on the working day of a week from 31 December 2008 to 28 December 2018. The sample of the raw data sets from S&P500 is shown in Table I.

TABLE I
Sample of Raw Data Set from S&P500

Date	Open	High	Low	Close	Adj Close	Volume
2008-12-31	890.59	910.32	889.67	903.25	903.25	-122027296
2009-01-02	902.99	934.73	899.35	931.80	931.80	-246697296
2009-01-05	929.17	936.63	919.53	927.45	927.45	1118942704
2018-12-27	2442.50	2489.10	2397.94	2488.83	2488.83	-198357296
2018-12-28	2498.77	2520.27	2472.89	2485.74	2485.74	-592347296

2) Data Pre-processing: The first pre-processing is dimension reduction. The acquired data sets consist of seven variables namely Date, Open, High, Low, Close, Adj Close, and Volume as shown in Table I. Since this study focuses on univariate time series prediction which involves only a single variable, the dimension of the data was reduced to two variables namely Date and Close. The date is kept to maintain the sequence of the time series. The main variable to calculate the time series prediction is Close. The sample of the final data set from S&P500 that was fed into the algorithm is shown in Table II.

The next pre-processing is data splitting where the data set was split into a training data set and testing data set. This study used three training-to-testing (training: testing) ratio setting which is 70:30, 80:20, and 90:10. The training data set was used to train the algorithm where the optimal set of weight was discovered throughout the iterations of the algorithm. On the other hand, the testing data set was used to test the prediction formula with the discovered optimal weight during training on a new set of data. By testing on a new data set that is not exposed during training, the performance of the algorithm could be differentiated from an overfitting case where the algorithm fit too well to the training data but performed poorly on new data. The isolation of testing data set from a training data set could also ensure generalization.

TABLE II
Sample of Final Data Set from S&P500

Date	Close
2008-12-31	903.25
2009-01-02	931.80
2009-01-05	927.45
2018-12-27	2488.83
2018-12-28	2485.74

B. Algorithm Design

1) Clonal Selection Algorithm: This study adapted the standard CSA as proposed by de Castro and Von Zuben [25] to solve the univariate financial time series problems. The algorithm receives six inputs namely antibodies Ab, the number of generations Ngen, population size N, selected population size n, random population size d, problem size L, and cloning rate β. The output of the algorithm is antibodies Ab and fitness value f. The algorithm starts by generating Ab which is a population of initial solutions with the size of N times L. The solutions in Ab are generated randomly. Then for every Ngen, the Ab is decoded to generate f. Based on f and n, a subset of Ab which is Ab_n is selected. Next, Ab_n is cloned according to β and f to generate the cloned population C. C is hypermutated based on f to generate a matured population C*. The fitness value of C* is generated by decoding C*. Based on C* and f*, n number of solutions are selected and inserted into Ab. Another population is randomly generated based on d and L to produce Ab_d. Ab will be replaced with Ab_d based on f i.e. Ab_d with better f will replace Ab. Finally, Ab is decoded to generate f and both Ab and f are presented as output. The pseudo-code for the algorithm is shown in Fig. 2.

```
Input : Ab,Ngen,N,n,d,L,β
Output      : Ab,f

Ab := rand(N,L);

for t = 1 to Ngen,
  f   := decode(Ab);
  $Ab_n$ := select(Ab,f,n);
  C   := clone($Ab_n$,β,f);
  C*  := hypermut(C,f);
  f*  := decode(C*);
  $Ab_n$ := select(C*,f*,n);
  Ab  := insert(Ab,$Ab_n$);
  $Ab_d$ := generate(d,L);
  Ab  := replace(Ab,$Ab_d$,f);
end;

f := decode(Ab);
```

Fig. 2 Pseudo Code of CSA

2) CSA Parameter Setting: The first parameter is the problem size which is based on the problem. CSA is an optimization algorithm. Therefore, the problem for this study which is the univariate financial time series prediction needs to be modeled into an optimization problem using a regression model [26] as shown in equation 1.

$$\hat{y}_t = x_1 y_{t-1} + x_2 y_{t-2} + x_3 \quad (1)$$

The predicted value, $\hat{y}_t$ is the summation of two-time lags, y_{t-1} and y_{t-2} with three weights, x_1, x_2, and x_3. This study used CSA to find the optimal set of weight, (x_1, x_2, x_3) that will return the closest predicted value, $\hat{y}_t$ from the actual value, y. Therefore, the problem size for this experiment is 3, which represents the three weights.

The next parameter is the population size, which is set to 100. Population size is the size of possible solutions at all times which means throughout the cloning and hypermutation; there are some elements added and some elements removed from the population. The number of elements to be selected for cloning is determined by the selection size which is set to 10. The cloning rate is set to 20 which means, each selected element will be cloned up to 20 times. The mutation rate, which is set to 0.2, on the other hand, is the rate of the hypermutation operator on the cloned population. This is followed by a random cell number which is set to 20.

The objective value for this algorithm is the RMSE which is explained in Section C. The objective of the optimization algorithm, which in this study is CSA, is to find the minimum value of RMSE, which is 0. This means that the CSA able to predict with zero highly unlikely error prediction. Therefore, this study also set another control parameter which is the stopping condition to 1000. This parameter is to ensure that the algorithm will not iterate infinitely. Instead, it will stop after 1000 iterations even if it does not reach the minimum objective value. The parameter setting for this experiment is summarized in Table III.

TABLE III
PARAMETER SETTING

Parameter	Value
problem_size	3
population_size	100
selection_size	10
clone_rate	20
random_cells_num	20
mutation_rate	0.2
stop_codition	1000

C. Evaluation

The performance of CSA is measured by the difference between the predicted value from the actual value. Many metrics could calculate this measurement like MAE, MAPE, and RMSPE. This study deployed Root Mean Square Error (RMSE) to evaluate the performance of CSA. The equation for RMSE is shown in (2).

$$RMSE = \sqrt{\frac{\sum_{t=1}^{n}(\hat{y}_t - y_t)^2}{n}} \quad (2)$$

Based on (2), the difference between the predicted value at time t, $\hat{y}_t$ and actual value at time t, y_t is squared for every time point from $t = 1$ to $t = n$. Then, the values for the whole time series are added and divided by the number of time points in the time series, n to get the average error. Lastly, the square root of the average error is calculated to produce the value of RMSE for the prediction.

In addition to using metric, the performance of CSA is also evaluated by comparing it with a benchmark model. This paper used the AutoRegressive Integrated Moving Average (ARIMA) as the benchmark. ARIMA is a popular and widely used statistical model for time series prediction. It contains three parameters called p, d, and q in the setting of ARIMA(p,d,q). The ARIMA setting that is used in this paper is ARIMA(3,1,0).

D. Experiment setup

The algorithm for this paper was developed in Python 3 programming language on Jupyter Notebook IDE 5.7.6. The experiments were executed on a desktop computer with Intel® Core™ i7-4770 CPU @ 3.40GHz processor, 8.00GB RAM, Windows 10 Professional 64-bit operating system, AMD Radeon™ HD 8490 graphic card, and 500GB HDD storage. The algorithm was trained and tested on three data sets namely S&P500, Gold, and EUR-USD.

III. RESULTS AND DISCUSSION

This study run experiments to determine the performance of CSA on univariate financial time series using data from S&P500 index price, Gold price, and EUR-USD exchange rate. The performance of the algorithm was measured using RMSE. The actual and predicted time series for each data set were plotted on three separate line graphs; one graph for each data set as shown in Fig. 3, Fig. 4, and Fig. 5 for S&P500, Gold, and EUR-USD, respectively.

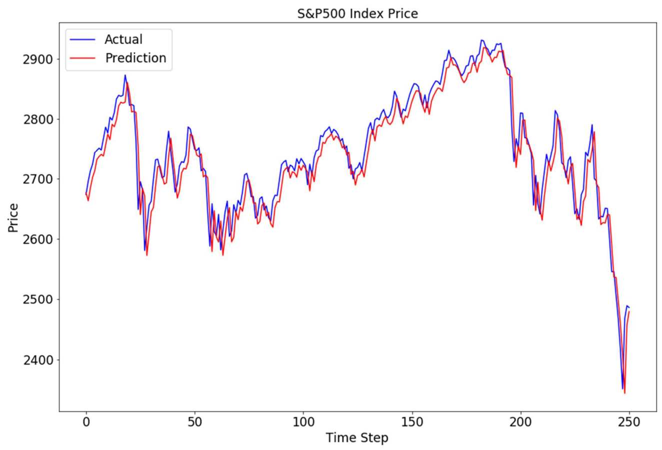

Fig. 3 Line graph of actual and predicted time series for S&P500

Fig. 3 illustrates the line graph for the S&P500 price time series. The actual time series is plotted using the blue line while the predicted time series is plotted using the red line. The RMSE for prediction of S&P500 is 22.439 which means in average, the predicted price of the S&P500 index produced by CSA has a variance of USD22.44 from the actual price.

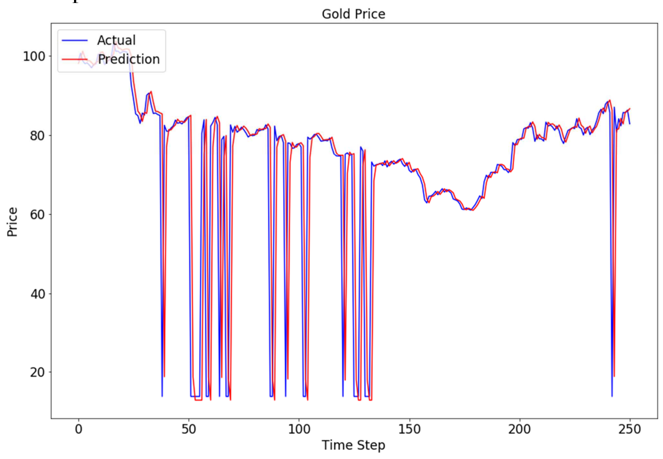

Fig. 4 Line graph of actual and predicted time series for Gold

Fig. 4 illustrates the line graph for the Gold price time series. The actual time series is plotted using the blue line while the predicted time series is plotted using red line. The RMSE for prediction of the Gold price is 16.890 which means in average, the predicted price of Gold produced by CSA has a variance of USD16.89 from the actual price.

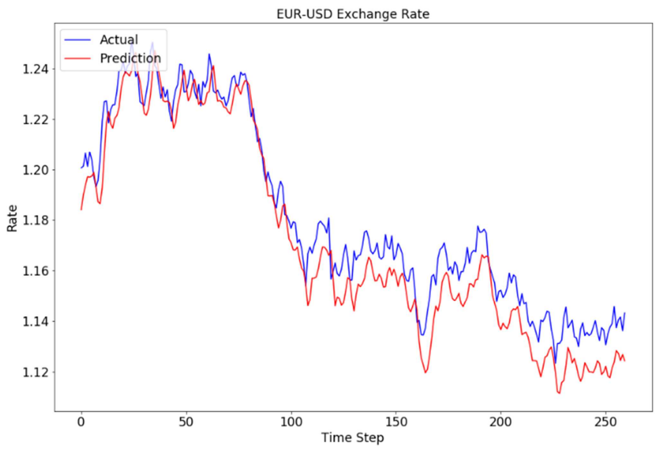

Fig. 5 Line graph of actual and predicted time series for EUR-USD

Fig. 5 illustrates the line graph for the EUR-USD exchange rate time series. The actual time series is plotted using the blue line while the predicted time series is plotted using the red line. The RMSE for prediction of EUR-USD exchange rate is 0.0102 which means in average, the predicted exchange rate of EUR-USD produced by CSA has a variance of USD0.01 from the actual exchange rate.

Based on the obtained results, it can be concluded that these experiments had achieved the objective of this study which is to perform univariate financial time series prediction using CSA. By comparing the pattern of the predicted time series (red line) with the original time series (blue line) plotted in the line graphs, as illustrated in Fig. 3, Fig. 4, and Fig. 5, it is learned that the predicted values generated by CSA carefully follow the actual values. This finding is an indication that CSA manages to capture the movement of the time series hence able to make a good prediction. Moreover, CSA even able to follow the rapid movement of Gold price around time steps 30 to 140 and 240. The maximum value, minimum value, range of value, and RMSE for each of the data sets for the experiments are summarized in Table IV.

TABLE IV
EXPERIMENTAL RESULTS OF UNIVARIATE FINANCIAL TIME SERIES PREDICTION USING CLONAL SELECTION ALGORITHM ON S&P500, GOLD, AND EUR-USD EXCHANGE RATE

Data Set	Max	Min	Range	RMSE
S&P500	2930.75	676.53	2254.22	22.439
Gold	126.88	13.12	113.76	16.890
EUR-USD	1.512791	1.039047	0.473744	0.0102

Based on Table IV, it is learned that the range of RMSE is relative to the range of the value of the data set. RMSE for S&P500 is the highest since its range of values is also the highest. Similarly, RMSE for EUR-USD is the smallest because its range of values is also the smallest. Therefore, a comparison between different data sets cannot be performed with RMSE. However, by comparing RMSE with the range of each data set, it is learned that the errors are small with the highest is not more than 20% for Gold and not more than 5% for S&P500 and EUR-USD. Overall, CSA successfully generates a decent prediction model.

To further evaluate the performance of CSA, this study ran several experiments with different training-to-testing ratio consists of 70:30, 80:20, and 90:10. This study also ran an experiment using ARIMA to be a benchmark for CSA. The data sets for ARIMA were also split into different training-to-testing ratio like the CSA setting to have a fair comparison. The results of these experiments are recorded in Table V with the best result for each model and each data set being bolded.

TABLE V
COMPARISON OF RMSE BETWEEN CSA AND ARIMA WITH DIFFERENT DATA SET AND TRAINING-TO-TESTING RATIO

Data Set	RMSE					
	CSA			ARIMA		
	70:30	80:20	90:10	70:30	80:20	90:10
S&P500	23.852	**22.439**	30.070	**20.188**	21.695	28.900
Gold	**12.891**	16.890	20.170	**11.224**	13.623	19.205
EUR-USD	0.0302	**0.0102**	0.0103	0.0054	**0.0053**	**0.0053**

Based on the result in Table V, it is learned that the ARIMA model dominates overall performance. However, the performance of the CSA model surpasses the performance of the ARIMA model when compared against the difference ratio. This differences in performance between ratio indicate that the ratio of training-to-testing data set influence the performance of a model. Therefore, it is essential to find the best ratio to optimize the performance of a model and to compare different models with the same ratio.

Another analysis that could be learned from Table V is that the performance of each model for each ratio does not return the best result for every data set. For the CSA model, the best ratio for S&P500 and EUR-USD data sets is 80:20 while for Gold data set is 70:30. For the ARIMA model, the best ratio for S&P500 and Gold data sets is 70:30 while for EUR-USD data set is 80:20 and 90:10. These findings concluded that the data set also affects the performance of a model.

IV. Conclusion

This paper presents an experimental study on univariate financial time series forecasting using clonal selection algorithm. Three data sets from the financial market were chosen for the experiments namely S&P500 index price, Gold price, and EUR-USD exchange rate to represent the financial market. Based on the result of the experiments which is illustrated in line graph and quantified using RMSE, it is learned that CSA can make decent predictions for financial time series. This finding implies that the proposed algorithm which is CSA can be implemented on a univariate financial time series prediction problem given that the problem is modeled as an optimization problem. However, CSA could not surpass the performance of widely used statistical models like ARIMA. Further research can be explored from the finding of this study such as running experiments using different experiment settings like number of iterations, number of initial populations, and size of the window to improve the performance of CSA. Furthermore, the proposed algorithm can be tested on data sets from other domains such as medical, geographical, and astronomy. Also, more research on related studies that implemented CSA would be a good review of its current state of the art. Finally, since CSA results are not better than ARIMA, there is a need to integrate it with machine learning techniques which have shown success in forecasting.

Acknowledgment

The Ministry of Education Malaysia has funded this study under the Fundamental Research Grant Scheme (S/O Code: 13820).

References

[1] R. B. Singh, “Financial Markets,” in *The DBS Handbook of Finance*, 1st ed., DBS Imprints, 2014, p. 159.

[2] W. Kenton, “New York Stock Exchange - NYSE,” 2018. .

[3] O. Kwon, S. Rahmatian, A. Iriberri, and Z. Wu, “Comparison of Neural Network and Ordinary Least Squares Models in Forecasting Chinese Stock Prices,” *Int. J. Bus. Econ.*, vol. 1, no. 1, pp. 1–17, Mar. 2018.

[4] X.-Y. Qian, “Financial Series Prediction: Comparison Between Precision of Time Series Models and Machine Learning Methods,” pp. 1–9, 2017.

[5] G. Mahalakshmi, S. Sridevi, and S. Rajaram, “A survey on forecasting of time series data,” in *2016 International Conference on Computing Technologies and Intelligent Data Engineering, ICCTIDE 2016*, 2016, pp. 1–8.

[6] L. A. Laboissiere, R. A. S. Fernandes, and G. G. Lage, “Maximum and minimum stock price forecasting of Brazilian power distribution companies based on artificial neural networks,” *Appl. Soft Comput.*, vol. 35, pp. 66–74, 2015.

[7] S. Bouktif *et al.*, “Optimal Deep Learning LSTM Model for Electric Load Forecasting using Feature Selection and Genetic Algorithm: Comparison with Machine Learning Approaches,” *Energies*, vol. 11, no. 7, p. 1636, Jun. 2018.

[8] Y. Al-Douri, H. Hamodi, and J. Lundberg, “Time Series Forecasting Using a Two-Level Multi-Objective Genetic Algorithm: A Case Study of Maintenance Cost Data for Tunnel Fans,” *Algorithms*, vol. 11, no. 8, p. 123, Aug. 2018.

[9] V. Manahov and H. Zhang, “Forecasting Financial Markets Using High-Frequency Trading Data: Examination with Strongly Typed Genetic Programming,” *Int. J. Electron. Commer.*, vol. 23, no. 1, pp. 12–32, Jan. 2019.

[10] J. Timmis, A. Hone, T. Stibor, and E. Clark, “Theoretical advances in artificial immune systems,” *Theor. Comput. Sci.*, vol. 403, no. 1, pp. 11–32, Aug. 2008.

[11] D. Dasgupta, S. Yu, and F. Nino, “Recent advances in artificial immune systems: Models and applications,” *Appl. Soft Comput. J.*, vol. 11, no. 2, pp. 1574–1587, Mar. 2011.

[12] R. Syahputra and I. Soesanti, “An artificial immune system algorithm approach for reconfiguring distribution network,” in *AIP Conference Proceedings*, 2017, vol. 1867, p. 20019.

[13] G. Cheng, “Unattended remote attestation delegation for grid computing,” 2009.

[14] H. S. Bernardino and H. J. C. Barbosa, “Artificial Immune Systems for Optimization,” Springer, Berlin, Heidelberg, 2009, pp. 389–411.

[15] C. Liang and L. Peng, “An Automated Diagnosis System of Liver Disease using Artificial Immune and Genetic Algorithms,” *J. Med. Syst.*, vol. 37, no. 2, p. 9932, Apr. 2013.

[16] J. Timmis, “Artificial immune systems—today and tomorrow,” *Nat. Comput.*, vol. 6, no. 1, pp. 1–18, Feb. 2007.

[17] Y. Peng and B.-L. Lu, “Hybrid learning clonal selection algorithm,” *Inf. Sci. (Ny).*, vol. 296, pp. 128–146, Mar. 2015.

[18] W. Pang, K. Wang, Y. Wang, G. Ou, H. Li, and L. Huang, “Clonal Selection Algorithm for Solving Permutation Optimisation Problems: A Case Study of Travelling Salesman Problem,” in *Proceedings of the International Conference on Logistics, Engineering, Management and Computer Science*, 2015.

[19] N. Xu, Y. Ding, L. Ren, and K. Hao, “Degeneration Recognizing Clonal Selection Algorithm for Multimodal Optimization,” *IEEE Trans. Cybern.*, vol. 48, no. 3, pp. 848–861, Mar. 2018.

[20] W. Luo and X. Lin, “Recent advances in clonal selection algorithms and applications,” in *2017 IEEE Symposium Series on Computational Intelligence (SSCI)*, 2017, pp. 1–8.

[21] Y.-C. Chou, Y.-H. Fan, M. Nakajima, and Y.-L. Liao, “Constrained design optimization of active magnetic bearings through an artificial immune system,” *Eng. Comput.*, vol. 33, no. 8, pp. 2395–2420, Nov. 2016.

[22] G. Pan, K. Li, A. Ouyang, and K. Li, “Hybrid immune algorithm based on greedy algorithm and delete-cross operator for solving TSP,” *Soft Comput.*, vol. 20, no. 2, pp. 555–566, Feb. 2016.

[23] Z. Li, X. Yan, Y. Fan, and K. Tang, “Improved Clonal Selection Algorithm for Solving AVO Elastic Parameter Inversion Problem,” in *Qiao J. et al. (eds) Bio-inspired Computing: Theories and Applications. BIC-TA 2018. Communications in Computer and Information Science*, Springer, Singapore, 2018, pp. 60–69.

[24] Y. Hu, X. Sun, X. Nie, Y. Li, and L. Liu, “An Enhanced LSTM for Trend Following of Time Series,” *IEEE Access*, pp. 1–1, 2019.

[25] L. N. de Castro and F. J. Von Zuben, “Learning and optimization using the clonal selection principle,” *IEEE Trans. Evol. Comput.*, vol. 6, no. 3, pp. 239–251, Jun. 2002.

[26] E. Hadavandi, A. Ghanbari, and S. Abbasian-Naghneh, “Developing a Time Series Model Based on Particle Swarm Optimization for Gold Price Forecasting,” in *2010 Third International Conference on Business Intelligence and Financial Engineering*, 2010, pp. 337–340.