A Typified Greedy Dynamic Programming Model for the MetaTrader Platform

Article · September 2020

CITATIONS
0

READS
281

1 author:

David Oyemade
Federal University of Petroleum Resources
27 PUBLICATIONS **49** CITATIONS

SEE PROFILE

Journal, Advances in Mathematical & Computational Sciences
Vol. 8 No. 3, September, 2020
www.mathematics-computationaljournal.info

Journal of Advances in Mathematical & Computational Sciences
An International Pan-African Multidisciplinary Journal of the SMART Research Group
International Centre for IT & Development (ICITD) USA

https://www.isteams.net/socialinformaticsjournal
DOI: dx.doi.org/10.22624/AIMS/MATHS/ V8N3P5
CrossREF Member Listing - https://www.crossref.org/06members/50go-live.html

A Typified Greedy Dynamic Programming Model for the MetaTrader Platform

Oyemade, D. A.
Department of Computer Science
Federal University of Petroleum Resources
Effurun, Nigeria
E-mail: oyemade.david@fupre.edu.ng
Phones: +234-8039209152

ABSTRACT

Users of the MetaTrader platform and forex investors are usually perplexed and puzzled on the choice of the best values for take profit, stop loss, and pending order distance for trade orders and positions for profit optimization. By design, the MetaTrader platform contains so many constraints with diverse effects that mystify the traders and investors. For example, increasing the pending order distance automatically reduces the take profit while the effect of such a decision on the stop loss is uncertain. Furthermore, trading with a very high stop loss combined with low or moderate take profit values usually produces very nice looking profits for a period but results in great losses in the long run because of the swooping losses that accompany high stop losses when hit by the forex market price. I propose a typified greedy dynamic programming model for the MetaTrader platform for the optimization of the take profit, stop loss, and pending order distance values to guide investors' decisions for optimal profitability. The model is a hybrid of a proposed typified greedy algorithm and dynamic programming algorithm. The proposed model makes a distinction between take profit, stop loss and pending order distance for the long orders, short orders and positions to clarify the effects of changes in the values of these data on an expert adviser's profit. Research results show that my proposed model out-performed the conventional dynamic programming algorithm for profit optimization.

Keywords: Greedy Algorithm, Dynamic Programming, Optimization, Forex, MetaTrader Platform, Profitability.

Oyemade, D.A. (2020): A Typified Greedy Dynamic Programming Model for the MetaTrader Platform. Journal of Advances in Mathematical & Computational Sc. Vol.8, No. 3. Pp 49-60. DOI: dx.doi.org/10.22624/AIMS/MATHS/ V8N3P5. Available online at www.isteams.net/mathematics-computationaljournal.

1. INTRODUCTION

The MetaTrader platform is a programmable integrated development environment available for traders, programmers and software developers for the automation of forex exchange (forex) trading decisions. Two modern programming language paradigms, imperative and object-oriented paradigms can be applied to the MetaTrader platform and implemented with MetaQuote language 4 (MQL4) and MetaQuote language 5 (MQL5) respectively [1, 2].

MQL4 resembles C programming language in structure while MQL5 adopts object-oriented programming style [3, 4]. Scripting is another recent programming language paradigm and domain applied to forex. ZuluScript is an example of the application of scripting style for the automation of forex trading decisions [5]. In forex, the automated system either places a bearish position or a bullish position for a currency pair, an example of which is GBP/USD. GBP stands for Great Britain pounds sterling while USD represents the United States dollar. In a clearer term, a bullish position means a buy or long position; a bearish position means a sell or short position. A position can be opened by either a physical manual trader or an automated trading system, on the MetaTrader platform. However, this study focuses on trading automation.

Once a position is opened, the decision cannot be reversed and it will eventually end in either a profit or a loss. Every currency pair is associated with a market price, which is dynamic, for buy or sell position. Opening a buy position means that beginning from the current market price, the trading system predicts that the price will increase to a certain value. Several and diverse factors are taken into consideration for the prediction or forecasting of the market price.

The failure of this prediction means a loss; a favorable prediction results in a profit. Similarly, opening a sell position implies that the trader or trading system predicts that the price will decrease to a certain value. The Take Profit is the value set for profit retrieval in case the market price moves in favor of the trading decision.

A Stop Loss is the value that you set to minimize your loss in case the market price moves against your prediction, since the direction of the market price is not certain. Two methods are available for opening a position: the market order method and the pending order method. The position opening method described above is known as the market order. In the pending order method, the trading system sets values for bearish or bullish movements and only opens a position if the market price reaches the value. The value set here is known as the Pending Order Distance. If the price does not move in the direction of the pending order distance, the pending order can be canceled without any loss but if the price reaches the pending order distance, a trading position is automatically opened.

The problem can be stated as follows: the take profit, stop loss and pending order distance were made dependent by the designer of the MetaTrader platform such that an increase or decrease in the value of any of the variables affects the others, and these effects on bearish or bullish order or positions are still made to be different. The effect of changes in the value of these variables on the profit that can be generated by a trading system appears to mystify traders.

Figure 1 shows the effect of Take Profit and Stop Loss values on the fluctuations of investment balances for a period of three months for an investment of 10,000 USD on a MetaTrader platform, using back-testing simulation. In the simulation, the value of only one variable was changed among a set of variables during the period and this produced diverse effects on the profit balances as shown in Figure 1.

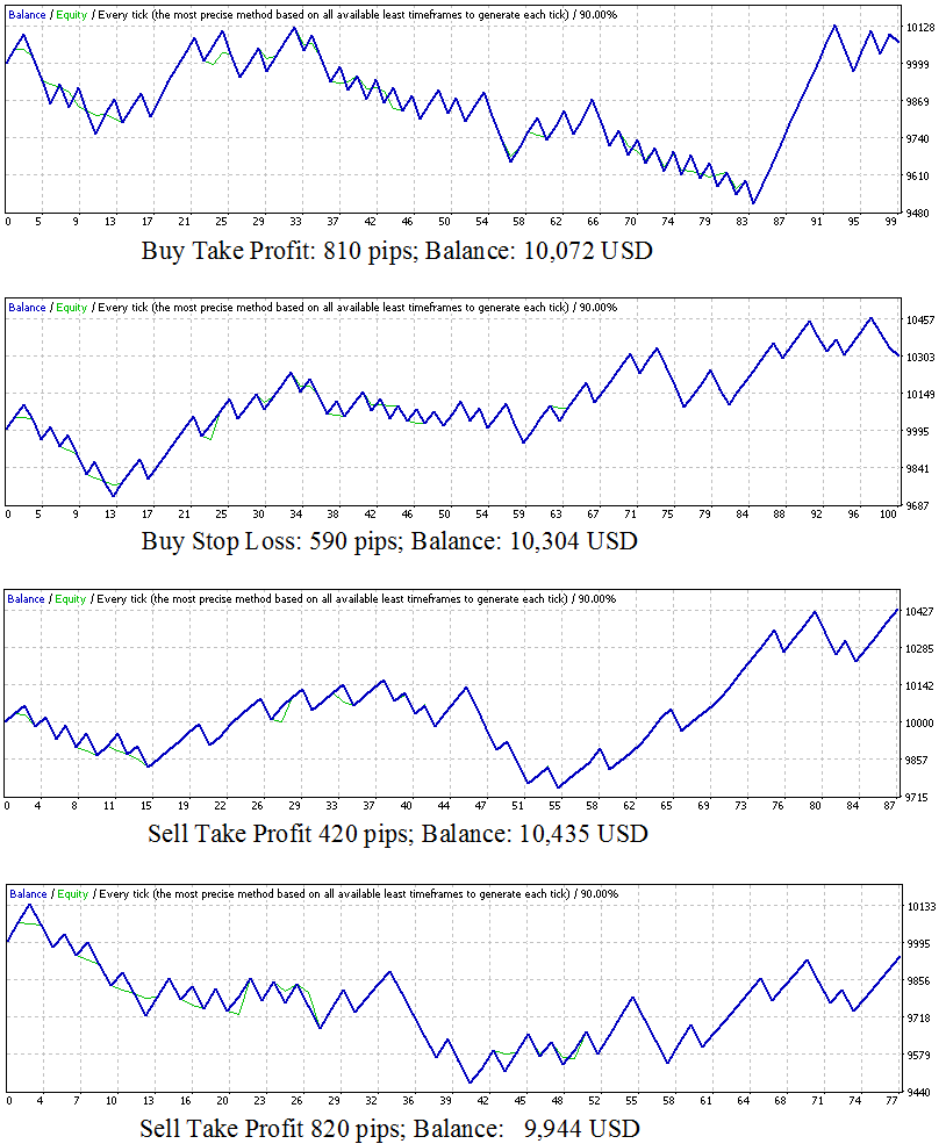

Figure 1: Effect of Take Profit and Stop Loss Values on Balances

The effect of changing the values of pending order distances on the investor's trading balance is shown in Figure 2. While other variables such as take profit, stop loss for bearish and bullish positions were made constant, buy pending order of 40 and 100 pips produced a balance of 10,069 USD and 10,195 USD respectively.

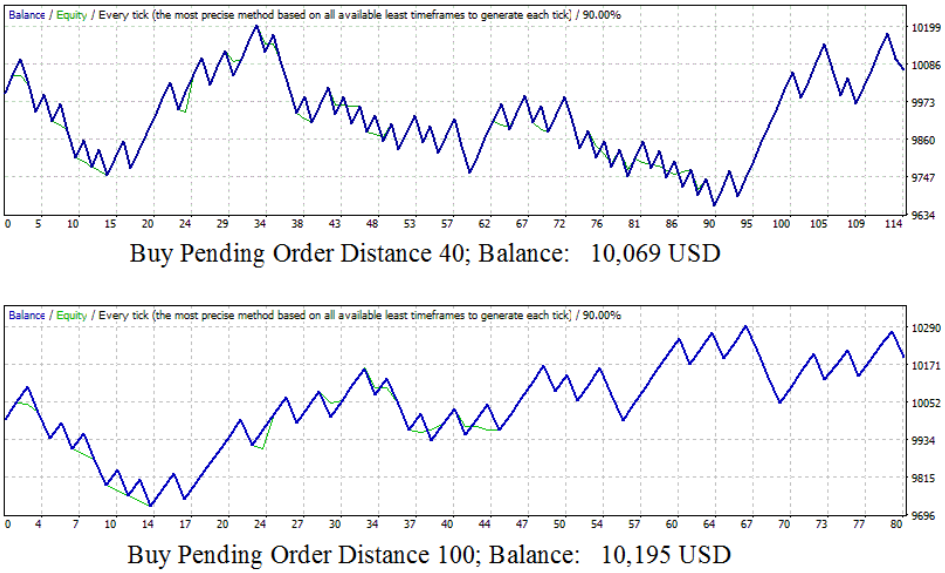

Figure 2: Effect of Pending Order Distance Values on Balances

The constraints and imbalance in take profit, stop loss and pending order distance are embedded in the MetaTrader platform by its designers and developers to make sustained profitability in the forex market difficult or almost impossible, as it seems. I propose a typified greedy dynamic programming model for the profit optimization of the MetaTrader platform.

The remaining parts of this paper are structured as follows. I describe the background of the study in section 2 and related works in section 3. The materials and methods are covered in section 4 while section 5 gives the dynamic programming problem formulation. The typified greedy dynamic programming model is presented in section 6. In section 7, the tests and results are presented. Finally, the conclusion is given in section 8.

1.1 Background of the Study

Dynamic programming algorithms typically divide a problem into sub-problems and solve it recursively. However, an important component of dynamic programming is a table or array that stores the solutions of the sub-problem for reuse during computation to avoid computing the solution from the start each time in the process of solving the problem [7]. The essential use of the table makes dynamic programming unique and distinguishes it from conventional computer programming, which employs programming languages as tools. Dynamic programming algorithms are used for tackling optimization problems. It has found wide application both in theoretical computing and in professional solutions to real-life problems. Assembly schedule in a factory, for example, a car assembly factory, has been used to illustrate and explain dynamic programming algorithms for the determination of the fastest route through the assembly factory.

Matrix chain multiplication, longest common substring and rod-cutting problems are some other commonly used examples and illustrations of the effectiveness of dynamic programming for solving optimization problems [8]. Apart from the use of tables, the properties of optimal substructure and the application of recurrence are other essential components of dynamic programming algorithm.

The property of optimal substructure exists if the solution to sub-problems can result in the solution to the entire problem. Recursion is used in computing to repeatedly solve the sub-problem to arrive at the optimal solution to the entire problem. In dynamic programming, the results of previous computations are stored in a table and reused to avoid repetitive computations and reduce calculation overhead. This act of storing the data on a table is similar to the documentation of data on a memo. It is therefore often referred to as memoization.

Also employed for solving the optimization problem is the greedy algorithm. Greedy algorithm requires ordering of the input data and the application of greedy property choice such that when an optimal value is selected without looking back [8], the value of the remaining choices can be computed and the remaining alternatives can be applied [10]. The problem should be divisible into smaller sub-problems so that repeated application of greedy decisions can result in an optimal overall solution. This is why it is applicable to fractional knapsack problem; if two tins of milk are selected from a carton, the weight of the two tins and the monetary worth can be calculated and the remaining weight that the knapsack can take can be determined. However, you must start by ordering the items with items of the highest values having weights less than the Total weight of the knapsack.

Therefore, what is needed is breaking all the items into the smallest units orders by price and weight, in descending order of price, with the weight indicated. In the knapsack problem, the thief will simply need to glance through the list, first pick the item with the highest monetary value that the bag can take, subtract the weight from the maximum weight the bag can accommodate to know the remaining weight that the bag can still accommodate. The thief will never go back to the beginning of the list but continue to still check the next item with the highest value that the bag can still accommodate, select item, note the cumulative amount and calculate the remaining weight that the bag can still accommodate. The process will be repeated until the bag cannot accommodate any other item.

2. RELATED WORKS

Previous studies have applied dynamic programming algorithm to a variety of computational problems. Fachini and Armentano [20] proposed an exact and heuristic dynamic programming algorithm for tackling the traveling salesman problem. Two alternative extensions were proposed to solve the problem. Sachan and Zhou [6] proposed a probabilistic dynamic programming model for obtaining the optimal cost-effective maintenance policy for power cables. Ozolins [19] proposed a dynamic programming algorithm for solving open shop scheduling problem. Sampurno et al [10] compared the application of dynamic programming and greedy algorithm on integer knapsack problem to freight transportation. Ji et al [21] introduced functional analysis to dynamic programming algorithm for the optimal scheduling of reservoir electric power generation.

Xiao and Konak [14] proposed a hybrid genetic algorithm and dynamic programming model for the reduction of CO_2 emission in the routes of green vehicles. Furini et al [17] proposed a new dynamic programming algorithm with a pseudo-polynomial time complexity for the knapsack problem while Rong and Figueira [18] focused on bi-objective integer knapsack problem. Maleki et al. [7] introduced a method for parallelizing dynamic algorithm. This additional feature to the dynamic programming algorithm allows the computation of multiple stages in spite of their dependencies. Haugh and Kogan [9] described the application of duality theory and approximate dynamic programming to option pricing in finance engineering and posited that the problem of portfolio optimization cannot be solved directly. Casperson [11] presented a recursive memoization method for converting recursive algorithms into dynamic programming algorithms and claimed that this approach is easier and more flexible.

Fragkou et al. [12] employed dynamic programming algorithm for the segmentation of linear text through the minimization of a segmentation cost function. Li et al. [13] formulate the minimum fragment removal problem in terms of integer linear programming problem and developed a dynamic programming algorithm to solve the problem. Tan el al. [15] demonstrated a scheme to exploit fine-grain parallelism of a dynamic programming algorithm with non-uniform data dependence on a multi-core architecture. Kopylov et al. [16] applied the concept of parametric dynamic programming to the problem of low-level signal processing.

Previous study failed to apply dynamic programming algorithm to the optimization of the MetaTrader platform, to my knowledge. A major contribution and novelty of this work is the proposal of a hybrid greedy and dynamic programming algorithms for the profit optimization of the take profit, stop loss and pending order distance, which are prominent intrinsic properties of the MetaTrader platform.

Some computer science methods previously applied in literature in the efforts toward profit optimization in the financial market include: machine learning [22, 31], adaptive fuzzy time series [23], support vector machine and genetic algorithm [24]; reinforcement learning [25], text mining of news [26], neural network and particle swam optimization [27, 30]; recurrent neural network [28] and deep learning [29].

Apart from the work of Haugh and Kogan [9] which applied duality theory and approximate dynamic programming to option pricing, the application of typified greedy and dynamic programming to the financial market is missing in the open literature. My proposed model contributes to knowledge filling this vacuum.

3. MATERIALS AND METHODS

The methods adopted in this research includes a dynamic programming algorithm with definite inputs combined with a modified greedy algorithm which uses the output of the dynamic programming algorithm as its input. The output of the dynamic programming algorithm is an array of best profit emanating from different choices which can be broadly classified into two types, buy or sell.

The dynamic programming algorithm obtained its own input from the profit generated through back-testing trade simulation on the MetaTrader platform using definite values of take profit, stop loss and pending order distance. The generated profits were captured in a table and used for the problem formulation in terms of dynamic programming.

The experiments were carried out on AAAFX US as the Forex broker and on MetaTrader platform [32], deployed on Commercial Network Services Virtual Private Server [33].

3.1 Dynamic Programming Problem Formulation

The problem of imbalanced effect of take profit, stop loss and pending order distance is formulated as a dynamic programming algorithm problem as shown in Table 1 which displays different values of profit generated with different take profit, stop loss and pending order distance values.

Table 1: The Effect of Different Values of TP, SL and POD on Profit

Choice1	Buy TP 410	Sell TP 420	Buy SL 590	Sell SL 600	POD Buy 40	POD Sell 40
Choice2	Buy TP 810	Sell TP 820	Buy SL 990	Sell SL 1000	POD Buy 100	POD Sell 100
Choice1 Profit	96	435	304	426	69	155
Choice2 Profit	72	-56	433	379	195	177

Table 1 contains two sets of values for take profit (TP), stop loss (SL) and pending order distance (POD), grouped as Choice1 and Choice2. Choice1 values are low values of TP, SL and POD while Choice2 values are high values of TP, SL and POD. The profits generated by Choice1 values during the back-testing simulation are shown in Table 1 as Choice1 Profit while the profit generated by Choice2 values are shown as Choice2 Profit. The profit balance generated by these sets of values, simply referred to as the existing model, is graphically illustrated in Figure 3. Table 1 can be referred to as the base table. For any column in the base table, either Choice1 or Choice2 must be selected at any instance. The dynamic programming algorithm seeks to answer the question: Which values of TP, SL and POD should we choose from the base table for profit optimization?

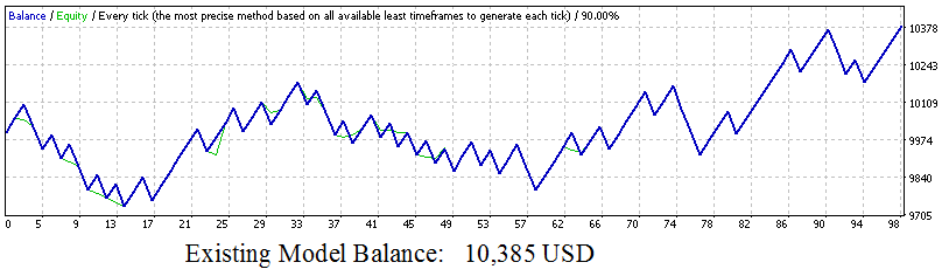

Figure 3: Existing Model with Initial Values

3.2 Typified Greedy Dynamic Programming Model

The procedure BestProfitForAnExpertAdvisor(w) represents the dynamic programming algorithm.
In the procedure,

b stands for best profit, which can be from Choice 1 set or Choice 2 set;
w stands for the weight of the choice, which can be from the Choice 1 set, in the first row of the table or from Choice 2 set in the second row of the table;
s stands for the set from where the choice is made based on the dynamic algorithm criteria. The set can be either from row 1 or row 2.

```
BestProfitForAnExpertAdvisor(w)
1.  b[1,1] ← w[1,1]
2.  b[2,1] ← w[2,1]
3.  for j ← 2 to n
4.    if b[1,j-1]+w[1,j] > b[2,j-1]+w[1,j]
5.          b[1,j] ← b[1,j-1]+w[1,j]
6.          s[1,j] ← (1,j-1)
7.    else
8.          b[1,j] ← b[2,j-1]+w[1,j]
9.          s[1,j] ← (2,j-1)

10.   if b[2,j-1]+w[2,j] > b[1,j-1]+w[2,j]
11.         b[2,j] ← b[2,j-1]+w[2,j]
12.         s[2,j] ← (2,j-1)
13.   else
14.         b[2,j] ← b[1,j-1]+w[2,j]
15.         s[2,j] ← (1,j-1)

16. if b[1,n] > b[2,n]
17.   s[n+1] ← (1,n)
18.   return b[1,n]
19. else
20.   s[n+1] ← (2,n)
21.   return b[2,n]
```

The principle of optimal substructure which is one of the hallmark of dynamic programming applicability can be seen in line 5, line 8, line 11 and line 14 which define the best profit for an expert advisor. The result of applying the dynamic programming algorithm is shown in Table 2 which displays Profit1(j) and Profit2(j).

Table 2: Dynamic Programing Memoization Table

J	1	2	3	4	5	6
Profit1(j)	96	531	835	1390	1495	1740
Profit2(j)	72	40	964	1343	1585	1762

Table 3 shows the output of the dynamic programming. The best choices of TP, SL and POD that would result in the optimal profit are coloured in red in Table 3.

Table 3: Dynamic Programming Selection Output

Buy TP 410	**Sell TP 420**	Buy SL 590	**Sell SL 600**	POD Buy 40	POD Sel 40
Buy TP 810	Sell TP 820	**Buy SL 990**	Sell SL 1000	**POD Buy 100**	**POD Sell 100**

The procedure for Typified Dynamic Programming algorithm is as follows:

TypifiedDynamicProgrammingSelector(DPChoice, DPChoiceValue, DPChoiceType)
Call BestProfitForAnExpertAdvisor(w)
Create a 3-Dimensional Array TypifieaArray(x,y,z) // where x,y,z represent the sizes of DPChoice, DPChoiceValue and DPChoiceType
Update the 3-Dimensional array

Table 4 shows the optimal selection using the typified-greedy dynamic algorithm procedure. For optimization, only the two selected items of unique type will replace the equivalent in the existing or base table. That is, only Sell TP of 420 and buy SL of 990 will be selected and used to update Table 1 for optimality. In the conventional greedy algorithm, the next greedy choice would have been considered using a loop until the termination condition of the greedy property choice is satisfied. I posit that this will be unnecessary since the design constraint of the MetaTrader platform centered on bearish and bullish positions only.

Table 4: Typified-Greedy Dynamic Algorithm Selection

DP Choice	Sell TP 420	Buy SL 990	Sell SL 600	POD Buy 100	POD SELL100	Buy TP 410
DP Choice Value	435	433	426	195	177	96
DP Choice Type	SELL	BUY	SELL	BUY	SELL	SELL

4. TESTS AND RESULTS

For the purpose of testing, three expert advisors were deployed on automated live trading without human intervention for a period of seven weeks. The first expert advisor used the TP, SL and POD from Table 1, the base table. This is referred to as the existing model. The second expert advisor employed the TP, SL and POD from Table 3, the output of conventional dynamic programming algorithm. The third expert advisor used the TP, SL and POD from the proposed typified greedy dynamic programming algorithm.

The existing method and conventional dynamic programming model produced negative results with losses of -153 and -121 pips respectively at the end of the live trading testing period. In contrast, typified greedy dynamic programming model produced a profit of 78 pips at the end of the same period. The results are shown graphically in Figure 4. Furthermore, the existing method and conventional dynamic programming model descended to lowest losses of -560 and -420 respectively, translating into draw-down percentage of 38% and 41% respectively, before improving to the undesirable negative values at the end of the experiments. On the contrary, the proposed typified greedy dynamic algorithm model experienced a draw-down of only 30%, which is preferable.

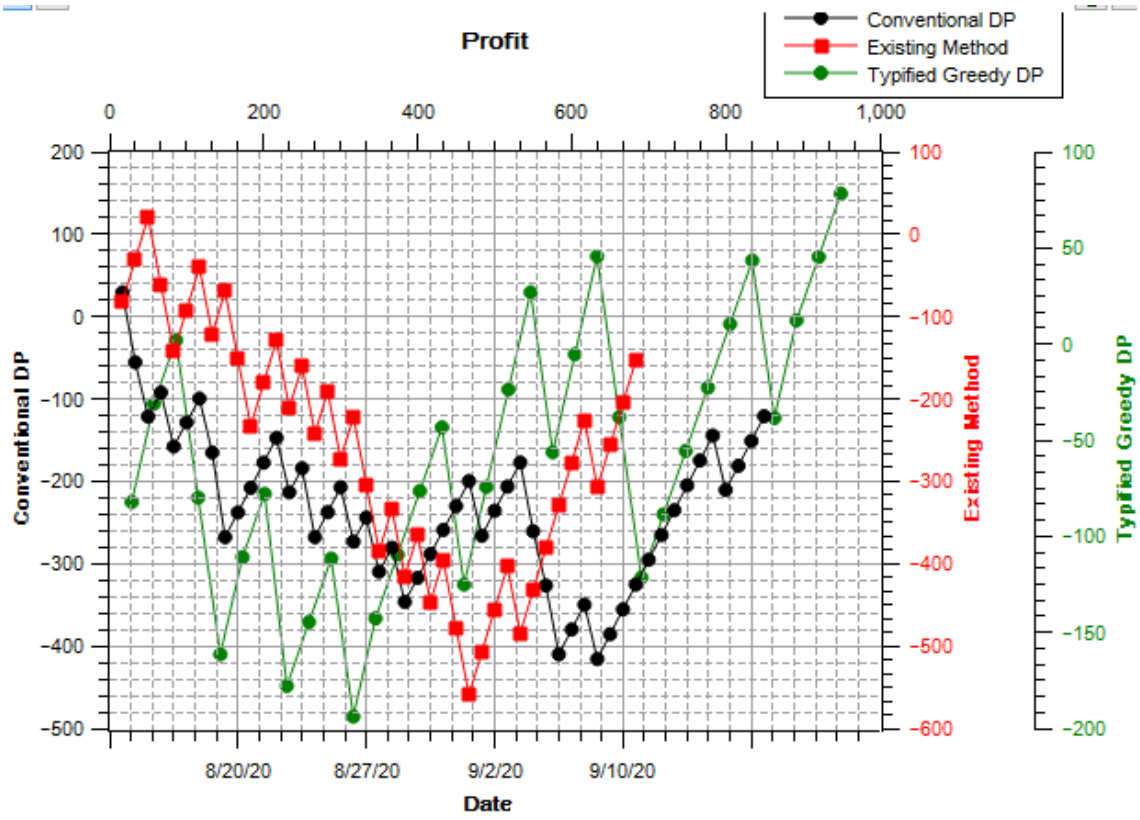

Figure 4: Performance Evaluation Results

5. CONCLUSION

In the design of the MetaTrader platform, the operational values of take profit, stop loss and pending order distance were made to depend on one another, producing a mystifying effect on the overall profit achievable by a trading system such as an expert advisor. Increase or decrease in the value of one affects the other and often produces negative effects on the overall profit achievable. This paper confirms the imbalance in the effects of these properties and their varied impacts on the profits of automated trading systems. A typified greedy dynamic programming model has been proposed for the MetaTrader platform for profit optimization to address this problem. The results showed that in the profits generated, the proposed typified greedy dynamic programming model outperformed the existing model designed without dynamic programming as well as the model designed with conventional dynamic programming algorithm which were all tested on live trading MetaTrader platforms. As part of the significance of this study, the results emphasize that for a trading system to be optimally profitable on the MetaTrader platform, the values of take profit, stop loss and pending order distance should not be assigned arbitrarily but followed a tested method and model. Future works shall investigate the effects of other intrinsic properties of the MetatTrader platform on the profitability of automated trading systems and the mutual effects of the properties on one another.

REFERENCES

[1] M. L. Scott, “Programming language pragmatics (3RD EDITION)”. Elsevier, 2009.
[2] R. W. Sebesta, “Concepts of Programming Languages”, Eleventh Edition, Pearson, 2016.
[3] https://www.metaquotes.net/en/metatrader5/algorithmic-trading/mql5. Accessed: 30th November 2020.
[4] https://www.mql5.com/en/docs/migration
[5] https://www.zulutrade.com/zuluscript-guide. Accessed: 30th November 2020.
[6] S. Sachan, C. Zhou, “Probabilistic dynamic programming algorithm: a solution for optimal maintenance policy for power cables”, Life Cycle Reliability and Safety Engineering, 8:117–127, 2019.
[7] S. Maleki, M. Musuvathi; T Mytkowicz, “Low-Rank Methods for Parallelizing Dynamic Programming Algorithms”, ACM Transactions on Parallel Computing, Vol. 2, No. 4, Article 26, pp. 26:1–26:32, 2016
[8] T. H. Cormen, C. E. Leiserson, R L. Rivest, C. Stein, “Introduction to Algorithms, *Third Edition*”, The MIT Press Cambridge, Massachusetts, 2009
[9] M.B. Haugh, L.Kogan, “Duality Theory and Approximate Dynamic Programming for Pricing American Options and Portfolio Optimization”, J.R. Birge and V. Linetsky (Eds.), Handbooks in OR & MS, Vol. 15, pp. 925-948, Elsevier, 2008.
[10] G. I. Sampurno, E. Sugiharti, Alamsyah, “Comparison of Dynamic Programming Algorithm and Greedy Algorithm on Integer Knapsack Problem in Freight Transportation”, Scientific Journal of Informatics, Vol. 5, No. 1, pp. 40-49, 2018.
[11] D. Casperson, “Towards a Better Presentation of Dynamic Programming”, WCCCE’14, May 2-3, 2014, Richmond, British Columbia, Canada, ACM, 2014.
[12] P. Fragkou, V. Petridis, A. Kehagias, “A Dynamic Programming Algorithm for Linear Text Segmentation”, Journal of Intelligent Information Systems, 23:2, 179–197, Kluwer Academic Publishers, 2004.
[13] Z. Li, L. Wu, Y. Zhao, X. Zhang, “A Dynamic Programming Algorithm for the *k*-Haplotyping Problem”, Acta Mathematicae Applicatae Sinica, English Series Vol. 22, No. 3, 405–412, 2006.
[14] Y. Xiao, A. Konak, “A genetic algorithm with exact dynamic programming for the green vehicle routing and scheduling problem”, Journal of Cleaner Production 167 (2017), pp. 1450-1463, 2017.
[15] G. Tan, N. Sun, G. R. Gao, “A Parallel Dynamic Programming Algorithm on a Multi-core Architecture”, SPAA’07, June 9–11, San Diego, California, USA, pp. 135-144, 2007.
[16] A. Kopylov, O. Krasotkina, O. Pryimak, and V. Mott, “A Signal Processing Algorithm Based on Parametric Dynamic Programming”, ICISP 2010, LNCS 6134, pp. 280–286, Springer, 2010.
[17] F. Furini, I. Ljubi, M. Sinnl, “An effective dynamic programming algorithm for the minimum-cost maximal knapsack packing problem”, European Journal of Operational Research 262, 438–448, Elsevier, 2017.
[18] A. Rong, J. R. Figueira, “Dynamic programming algorithms for the bi-objective integer knapsack problem”, European Journal of Operational Research 236, pp. 85–99, 2013.
[19] A. Ozolins, “Dynamic programming approach for solving the open shop problem”, Central European Journal of Operations Research, Springer, 2019.
[20] R. F. Fachini, V. A. Armentano, “Exact and heuristic dynamic programming algorithms for the traveling salesman problem with flexible time”, Optimization Letters, 14:579–609, Springer, 2020.
[21] C. Ji H Wu, C. Li, “Optimal scheduling system for reservoir electric power generation based on functional dynamic programming algorithm”, Journal of Interdisciplinary Mathematics, Vol. 21, No. 5, pp. 1267–1272, Taylor and Francis, 2018.
[22] E. A. Gerlein , M. McGinnity , A. Belatreche, S. Coleman, “Evaluating machine learning classification for financial trading: An empirical approach”, Expert Systems With Applications 54, pp. 193–207, Elsevier, 2016.

[23] M. Bahrepour, R. Mohammad. T. Akbarzadeh, M. Yaghoobi, B.Mohammad. S.Naghibi, “An adaptive ordered fuzzy time series with application to FOREX”, Expert Systems with Applications 38, pp. 475–485, Elsevier, 2011.

[24] B. J. de Almeida, R. F Neves, N. HortaInstituto “Combining Support Vector Machine with Genetic Algorithms to optimize investments in Forex markets with high leverage”, Applied Soft Computing 64, pp. 596–613, Elsevier, 2018.

[25] João Carapuço, Rui Neves and Nuno Horta, “Reinforcement learning applied to Forex trading”, Applied Soft Computing Journal 73, Elsevier RV, pp. 783–794, 2018

[26] A. K. Nassirtoussi, S. Aghabozorgi, T. Y. Waha, D. C. L. Ngo, “Text mining of news-headlines for FOREX market prediction: A Multi-layer Dimension Reduction Algorithm with semantics and sentiment”, Expert Systems with Applications 42, pp. 306–324, Elsevier, 2015.

[27] Georgios Sermpinis, Konstantinos Theofilatos et al, "Forecasting foreign exchange rates with adaptive neural networks using radial-basis functions and Particle Swarm Optimization", European Journal of Operational Research, Elsevier, vol. 225(3), pp. 528-540, 2013.

[28] Mehreen Rehman, Gul Muhammad Khan and S. A. Mahmud, “Foreign Currency Exchange Rates Prediction using CGP and Recurrent Neural Network”, 2014 International Conference on Future Information Engineering, ScienceDirect, IERI Procedia 10, pp. 239 – 244.

[29] Lina Ni and Yujie Li et al, “Forecasting of Forex Time Series Data Based on Deep Learning”, International Conference on Identification, Information and Knowledge in the Internet of Things, ScienceDircct, Procedia Computer Science 147 (2019) pp. 647–652, 2019.

[30] Emam Ahmed, “Optimal artificial neural network topology for foreign exchange forecasting”, ACM-SE 46: Proceedings of the 46th Annual Southeast Regional Conference on XXMarch 2008, pp. 63-68.

[31] E. A. Gerlein, M. McGinnity, A. Belatreche, S. Coleman, “Evaluating machine learning classification for financial trading: An empirical approach” Expert Systems With Applications 54 () 193–207, 2016

[32] https://www.aaafx.com

[33] www.commercialnetworkservices.com