Combining Different Meta-heuristics to Improve the Predictability of a Financial Forecasting Algorithm

Babatunde Aluko
Centre for Computational Finance and Economic Agents, University of Essex, Wivenhoe Park, CO4 3SQ, UK

Dafni Smonou
Centre for Computational Finance and Economic Agents, University of Essex, Wivenhoe Park, CO4 3SQ, UK

Michael Kampouridis
School of Computing, University of Kent, Chatham, ME4 4AG, UK

Edward Tsang
Centre for Computational Finance and Economic Agents, University of Essex, Wivenhoe Park, CO4 3SQ, UK

***Abstract*— Hyper-heuristics have successfully been applied to a vast number of search and optimization problems. One of the novelties of hyper-heuristics is the fact that they manage and automate the meta-heuristic's selection process. In this paper, we implemented and analyzed a hyper-heuristic framework on three meta-heuristics namely Simulated Annealing, Tabu Search, and Guided Local Search, which had successfully been applied in the past to a Financial Forecasting algorithm called EDDIE. EDDIE uses Genetic Programming to extract and learn from historical data in order to predict future financial market movements. Results show that the algorithm's effectiveness has improved, thus making the combination of meta-heuristics under a hyper-heuristic framework an effective Financial Forecasting approach.**

Keywords—Genetic Programming; Financial Forecasting; Hyper-heuristics

I. INTRODUCTION

Financial forecasting is the process of predicting the future movements of financial markets (E.g. risk, volatility, and financial assets). It is an important aspect of finance and has led investors to develop several methods in order to accurately forecast the financial market movements. Over the years, several computational intelligent methods such as Genetic Algorithm [1], Fuzzy Logic [2], Genetic Network Programming [3], Learning Classifier Systems [4], Artificial Neural Network [5,6] have been used for financial forecasting.

In recent times, EDDIE [7,8,9], which is a financial forecasting algorithm that makes predictions by employing Genetic Programing [10,11] has been presented. The newest version is called EDDIE8, which when compared with its predecessor EDDIE7, it allows investors to specify a period range for its technical analysis indicators[1] instead of using pre-specified periods [12]. To be more specific, EDDIE7 allows the user to specify two pre-fixed period lengths for its indicators which is the norm observed in both the academic and industry literature. For instance, with EDDIE7 an indicator could have a pre-fixed period of 12 & 50 days. On the other hand, EDDIE8 uses a period from a parameterized range between 2 and 65. Hence, the algorithm can create new indicators, which have never been used in the past. This step was very significant, because as mentioned other works in the literature and also people in the industry have been using pre-specified indicators (e.g. 20 days Moving Average and 50 days Moving Average), without being able to justify why a 20 days Moving Average should always be preferrable to 25 days Moving Average. Results have shown that EDDIE8 could find new and better solutions compared to EDDIE7's non-dynamic framework [12,13]. However, EDDIE8 could not consistently outperform EDDIE7 as the search space was larger, and searching was ineffective which often led to local optima problem [13]. In order to address this stated problem, Smonou et al. [14] successfully applied three meta-heuristics named Simulated Annealing (SA), Tabu Search (TS), and Guided Local Search (GLS) to the period nodes of EDDIE's trees. As a result, both EDDIE's search effectiveness significantly improved, as well its predictive performance in terms of best results [14]. However, the meta-heuristics were only tested individually leaving room for further improvement and experiments.

To this end, in this work we will be combining the above meta-heuristics under a hyper-heuristics framework and evaluate the results. The merit of the hyper-heuristics lies on the fact that the advantages of the meta-heuristics are combined together into one algorithm and the meta-heuristic selection process is automated. Hence, an investor would not need to worry about which meta-heuristic is more appropriate on a given dataset. This is very important, because of the significance of the financial forecasting itself, which requires the continuous development of new and improved algorithms. The goal of our research is to show that as a result of the combination of the above meta-heuristics under a hyper-heuristic framework, EDDIE8's search effectiveness can be further improved, providing an even more reliable tool for the financial industry.

The rest of this paper is organized as follows: Section II presents EDDIE, Section III presents and discusses the hyper-heuristic method used in this research, Section IV covers the analysis of the experimental design, Section V presents and discusses the results, and finally, Section VI concludes this paper, and offers a recommendation for further development.

II. EDDIE

This section presents in detail the EDDIE process, the performance metrics used by the algorithm, and investigates the latest version of EDDIE called EDDIE8.

[1] A technical indicator is a tool used to measure and interpret market behavior. The list of technical indicators used by EDDIE includes the Moving Average (MA), Trade Break Out (TBR), Filter (FLR), Volatility (Vol), Momentum (Mom), and Momentum Moving Average (MomMA).

A. EDDIE Process

As mentioned in the introduction, EDDIE is a financial forecasting algorithm that uses Genetic Programming to extract data in order to predict the future price of a stock. EDDIE's goal is to answer the following question "Will the price increase by r% within the next n days?" [15]. The program receives a dataset which comprises of three inputs: historical data (daily closing prices of stocks or indices), a group of technical indicators, and binary target signals which are recommendations of buy (1) or not-to-buy (0). The user specifies the technical indicators which are relevant to the prediction along with the period range. EDDIE uses the inputted information through a GP process to construct Genetic Decision Trees (GDTs) needed to create the binary target signals of buy (1) and not-to-buy (0). Each individual GDT is then evaluated and evolved on a training dataset for a number of pre-defined generations. At the end of the process, the GDT with the highest fitness is applied to the testing data.

Additionally, the performance of the best GDT applied on the data is evaluated by comparing the level of prediction[2] against the reality[3]. The performance of the best GDT is measured through the 3 metrics presented in (1), (2) and (3). The TP, TN, FP and FN represent the True Positive, True Negative, False Positive and False Negative comparison results respectively.

Rate of Correctness (RC):

$$RC = \frac{TP+TN}{TP + TN + FP + FN} \quad (1)$$

Rate of Missing Chances (RMC):

$$RMC = \frac{FN}{FN + TP} \quad (2)$$

Rate of Failure (RF):

$$RF = \frac{FP}{FP + TP} \quad (3)$$

These three metrics were further combined by Li & Tsang to produce the fitness function presented in (4)

$$ff = w1 * RC - w2 * RMC - w3 * RF \quad (4)$$

where *w1*, *w2* and *w3* are the weights assigned to RC, RMC and RF respectively. These values were carefully selected in order to capture the different investor's preferences.

Further to this, Li & Tsang introduced a constraint function as described in (5), (6) and (7), in order to further improve the effectiveness of the fitness function.

$$R = [C_{min}, C_{Max}] \quad (5)$$

$$C_{min} = \frac{Pmin}{Tot\_No\_Td} X100\% \quad (6)$$

$$C_{Max} = \frac{Pmax}{Tot\_No\_Td} X100\% \quad (7)$$

In the above formulas, R denotes the effective constraint function, C_{min} and C_{max} represent the percentage of the minimum and maximum positive predictions required, Pmin and Pmax represents the total number of minimum and maximum positive predictions, and Tot_No_Td represents the total number of training data. It is important to state that the value of the weight of RC (*w1*) remained the same if the percentage of positive signal that the GDT produced falls within the [C_{min}, C_{max}] range, otherwise a value of zero is assigned to w1. More specifically, a constraint of R = [55,60] means the percentage of positive signals that a GDT predicts should fall within that pre-specified range. If this occurs, the *w1* remains the same; else, a value of zero is assigned to *w1*. This is useful because this contraint guide the search thus, makes it focus on important solutions. Research has shown that this method significantly reduces RF, while keeping RC at the same level. More information regarding the selection of these values can be found in [17,16].

B. Presentation of EDDIE8

EDDIE8, which is the current implementation of EDDIE, uses the GP to search the search space of technical indicators for solutions, which are represented in the form of GDTs. This as such was possible because of its extended grammar. Hence, an investor can specify indicators' periods in a parameterized range between 2 and 65. This merit gives EDDIE the ability to make the best combination of indicators and periods each time. Furthermore, this functionality provides EDDIE with a lot of flexibility as currently both academic and industrial applications use only pre-specified indicators. Fig. 1 illustrates the Backus Normal Form (BFN) grammar used by EDDIE8.

<Tree> ::= If-then-else <Condition> <Tree> <Tree> | Decision
<Condition> ::= <Condition> "And" <Condition> |
<Condition> "Or" <Condition> |
"Not" <Condition> |
VarConstructor <RelationOperation> Threshold
<VarConstructor> ::= MA period | TBR period | FLR period | Vol period | Mom period | MomMA period
<RelationOperation> ::= ">" | "<" | "="
Terminals:
MA, TBR, FLR, Vol, Mom, MomMA are function symbols
Period is an integer within a parameterized range, [Pmin, Pmax]
Decision is an integer, Positive or Negative implemented
Threshold is a real number

Fig. 1. The Backus Normal Form of ED8 [12].

As it can be observed from Fig. 1, the tree starts with an if-then-else as the root, continues with a Boolean or a logic operator as its first branch and finally a *VarConstructor* which takes as children the "technical indicator" and the "period" (denoted as [Pmin, Pmax]), which can be used to carry out genetic operations such as mutation and crossover [13]. The *VarConstructor* gives the program the ability to construct variables by combining various technical indicators with

[2] These are the predictions made by EDDIE.
[3] These are the actual the events that have already happened. The signals are calculated by looking ahead of the closing price for a time horizon of *n* days at the specified r% [27].

periods[4] such as 23 days Momentum, 19 days Trade Break Out, and so on. The advantage of this was that EDDIE was able to create new technical indicators, which had never been used before for financial forecasting.

However, the aforementioned advantage also gave birth to a drawback, which has to do with the significantly larger search space. For instance, if a given GDT has a maximum of k indicators and a period range between 2 and 65, then the permutations of exhausting the search space of 384 indicators (i.e. 6 indicators * 64 periods) would be 384^k. Whereas, in the previous version of EDDIE which was limited to 6 indicators and 2 periods, the permutation of exhausting the search space of 12 indicators (i.e. 6 indicator * 2 periods) would be 12^k. This extremely large search space made it much more difficult for EDDIE8 to consistently find better solutions than its predecessor. For this reason, we were interested in improving the local search of the algorithm at the level of the period nodes, since this was the main reason of the enlarged search space. This will be achieved by implementing a hyper-heuristic on three successful meta-heuristics to better explore the search space. The next section presents Smonou et al.'s previous work on meta-heuristics, which had the goal of improving the search effectiveness.

C. Meta-heuristics and Hyper-heuristic Framework

In order to solve this presented challenge, three meta-heuristics were successfully applied by Smonou et al. [14] with the view to better explore the huge search space and find better solutions. These meta-heuristics were applied to the period of *VarConstructor* of the tree (therefore on the period leaf nodes of the tree) and each meta-heuristic followed a general process of selecting a tree based on a pre-specified probability, and subsequently randomly selecting any *VarConstructor* of the tree. In each case, we make marginal changes to the periods of each *VarConstructor* of the given tree, until the termination criteria is met. The periods that result to an improved tree fitness are then saved and they replace the old periods. Since in this paper we are presenting an extension of [14] by combining these 3 metaheuristics under a hyper-heuristics framework, it is essential to present a summary of the mechanism of each meta-heuristic. As the SA and the TS have been extensively presented in the literature [18,19], we will not go into much detail regarding their process. However, the GLS will be explained more thoroughly as it is considered a rather new meta-heuristic approach.

The Simulated Annealing [14] meta-heuristic improves the GDT's fitness by probabilistically selecting a tree and then indentifying and selecting all neighbors [5] surrounding the selected *VarConstructor's* period. The process is then iteratively exploring the search space of the neighbors of the tree until the terminal criteria are met. Additionally, worst solutions are probabilistically accepted and kept.

The Tabu Search [14] meta-heuristic follows a similar process like the Simulated Anealing with the addition of two parameters, which are the *aspiration criteria* and *tabu list*. The former keeps track of the memory size of neighbors that the solutions have currently visited without improvement, and the latter controls the memory size of the neighbors that contain promising solutions.

Last but not least, the Guided Local Search [14] meta-heuristic was added on a Hill Climbing local search in order to escape the local optima by guiding the search through the use of penalties and an augmented fitness function. GLS is activated when the search gets stuck in local optima, which occurs when the fitness has failed to improve (as a result of the local search in the tree's periods) a pre-specified number of times. As explained in [14], GLS uses an augmented fitness function as can be seen in Equation (8) to guide the Local Search out of the local optima:

$$h(s) = g(s) - \lambda \cdot \sum_{i=1}^{M} p_i \cdot I_i(s) \quad (8)$$

where, $g(s)$ is the fitness function, M the number of features, p_i is the penalty parameter for feature f_i and λ is a regularization parameter.

The advantage of using Equation (8) when GLS is used, is that the local optima which are confronted are with respect to the augmented function. Therefore, these may be different than the local optima with respect to the original fitness function (4). Before any penalties are applied (p_i), the original fitness function and the augmented function are the same, however as search progresses, the augmented function is being constantly updated. This allows the local search to escape from the local optima of the original fitness since GLS is altering the local optima status under the augmented fitness function using a penalty modification mechanism. For more information about the process of GLS the reader can refer to [20,14]

Results from this experiment in [14] were successful as it showed promising solutions as each meta-heuristic significantly improved the performance of EDDIE8 in terms of best results. However, the meta-heuristics applied had a few limitations. The first limitation was that the experimental parameters of each meta-heuristic were not fully investigated. Results could thus be affected by changing the values of the experimental parameters of the above meta-heuristics. Such parameters are for the SA, the *Temperature* and *kmax* (which is the maximum number of iterations), for TS the *aspiration criteria* (the memory size of neighbors the solutions has currently visited without improvement) and the *tabu list* (which is the memory size of the neighbors that contains promising solution) and finally for the GLS the *kmax* (which is maximum numbers of iterations). Furthermore, the second limitation was the fact that the meta-heuristics were only applied and tested independently. Hence, one algorithm (e.g. SA) might be good for some datasets, while another algorithm (e.g. TS) might be better for other datasets.

In order to tackle the above issues, we decided to improve each of the meta-heuristics presented above by tuning the experimental parameters and more importantly, to combine the best versions of these meta-heuristics under a hyper-heuristic framework. *Our purpose thus in this paper is to*

[4] The period represents an integer value in parameterized range between 2 and 65 that an indicator can take.

[5] A neighbor is the surrounding interval, which can be obtained by making a marginal change (k) to the selected period value, e.g. $5 \leq k \leq 5$. For example, if the selected period is 15 days of MA, then neighborhood will be any periods from 10 to 20 days.

automate the meta-heuristics selection process, while at the same time investigate the effect of the application of the hyper-heuristics framework on EDDIE8's performance.

III. METHODOLOGY

This section describes in detail the implementation of a hyper-heuristics framework called Choice Function on EDDIE. The reasons behind the adoption of the Choice Function, were firstly, its proven efficiency across various research fields [20, 21], as well as the fact that it fits well on our problem as it is a score based technique where different heuristics are selected based on the aggregation of three score measures.

A. Choice Function Hyper-heuristic

Choice Function hyper-heuristic was first introduced and presented in [21]. Detailed information about the Choice Function adopted for this research can be found in [22].

The Choice Function selects a heuristic from a set by assessing their performance based on three measures namely, the performance of each individual heuristic, how well a given heuristic has performed in relation to the previous heuristic used, and the heuristic that was last selected by the Choice Function. The meta-heuristics are scored based on the mixture of the measures described in (9), (10), (11) and (12).

The first measure $f_1(h_j)$ evaluates the performance of each individual heuristic $h_1, h_2, h_3..hj$ as described in (9):

$$f_1(h_j) = \sum_{n}^{i} \alpha^{n-1} \frac{I_n(h_j)}{T_n(h_j)} \quad (9)$$

where $I_n(h_j)$ is the change in the fitness function with regards to the nth time that h_j was used, $T_n(h_j)$ is the time taken in milliseconds from the time the heuristic was used in the *nth* last time until when it returned a solution to the Choice Function and α is a random value between (0,1), interval with a decreasing geometric behavior assigned to the past performance values of h_j.

The second measure evaluates how well a given heuristic has performed in relation to the previous heuristic used. Suppose a heuristic h_j is called, which is based on the previous heuristic h_k, then the value of the function f_2, is calculated using the formula in (10):

$$f_2(h_j,h_k) = \sum_{n}^{i} \beta^{n-1} \frac{I_n(h_j,h_k)}{T_n(h_j,h_k)} \quad (10)$$

where $I_n(h_j,h_k)$ is the change in the fitness function with regards to the nth time that the pair (h_j,h_k) was used, $T_n(h_j,h_k)$ is the time taken in milliseconds from the time the pairs (h_j,h_k) were used in the *nth* last time until it returned a solution to the Choice Function and β is a random value between (0,1) interval with a decreasing geometric behavior assigned to the past performance values of the pairs (h_j,h_k).

The third measure evaluates the elapsed time since a given meta-heuristic was last used by the Choice Function and $\tau(h_j)$ is set to zero each time h_j is used. The function (f_3) is presented in (11):

$$f_3(h_j) = \tau(h_j) \quad (11)$$

At any stage of the search, the meta-heuristics are accessible to the Choice Function and the score of each measure is computed using (12):

$$f(h_j) = \alpha f_1(h_j) + \beta f_2(h_j,h_k) + \delta f_3(h_j) \quad (12)$$

where α and β are weights that are used to intensify the search and δ is a weight used to diversify the search. These factors increase the efficiency by which the Choice Function explores the search space.

B. Choice Function on EDDIE8

Based on the general description of the Choice Function as presented in the previous sub-section, the Choice Function was implemented on EDDIE as presented in Fig 2.

While k < = PopulationSize
    Select a candidate from the population with a probability of P.
    ForEach (Meta-heuristics)
        Evaluate the performance measure $f_1(h_j)$, $f_2(h_j,h_k)$, $f_3(h_j)$ and calculate the choice function score F.
    Select the meta-heuristics with the highest F score
    Apply the meta-heuristic to the period of the tree.
End While.

Fig. 2 Choice Function Pseudocode

As it can be seen from Fig 2, the process begins by randomly selecting a tree from the population based on a specified probability. When this probability condition is met, the Choice Function value $f(h_j)$ is evaluated and calculated for each heuristic as described in Section III-A. In addition, the heuristic with the highest $f(h_j)$ is selected. The meta-heuristic selected is then applied to the selected tree. As a rule, these meta-heuristics explore the search space of the periods of *VarConstructor* of the selected tree to search for better fitness value until the termination criteria are met. Every time the meta-heuristic is applied, the values of the Choice Function measure score $f_1(h_j)$, $f_2(h_j,h_k)$, $f_3(h_j)$ are updated accordingly. This process continues until the population size is exhausted. The merit of this approach lies on the fact that it enables the different meta-heuristics to be applied on the period of the tree. These above factors, which effectively control the intensification and diversification of the search, increase EDDIE's exploration and exploitation abilities of the huge search space. Hence, enhances the chance of improving the performance of EDDIE8 algorithm.

IV. EXPERIMENTAL DESIGN

This section presents the data and experimental parameter values that have been used for the adjusted meta-heuristics and the Choice Function hyper-heuristic. The data used for this experiment can be collected from *"DataStream"* or *www.finance.yahoo.com*.

In this research, 9 datasets were used. They consist of 7 FTSE 100 stocks (Aggreko, Easyjet, Hammerson, HSI, Imperial Tobacco "denoted as Imp", Marks_Spencer, Next) along with 2 indices (MDAX, NASDAQ). The reason behind using these stocks is because of their previously observed good performance [12,13,23].

Furthermore, the data fed into EDDIE for our experiment had a time horizon of 1300 days, where 1000 days were used as the training days and 300 days as the testing days. In addition, the experiment parameter values presented in Table I, and Table II were used for the experimental design. The parameters used for the experiment remained the same as used by Smonou et al. [14]. The reason behind using these values is to maintain consistency with the previous research for the result analysis. In Table I, the GP parameters used by EDDIE8 are illustrated. Specifically, the program generates 500 GDTs with a maximum initial depth of 6, maximum tree depth of 8 having 0.1, 0.01, 0.9 probabilities of reproduction, mutation and crossover respectively. Those are then evolved for 50 generations over a 1000 days training period. At end of each generation, the best GDT (which is the one with the highest fitness (4)) is applied on a testing period.

Table I
GP PARAMETERS

GP Parameters	Values
Max initial Depth	6
Max Depth	8
Generation	50
Population Size	500
Tournament Size	2
Reproduction probability	0.1
Mutation probability	0.01
Crossover probability	0.9

Table II
EDDIE PARAMETERS

EDDIE Parameters	Values
n	20
EDDIE8 Period	[2, 65]
Weight *w1*	0.6
Weight *w2*	0.1
Weight *w3*	0.3

Furthermore, Table II presents the EDDIE's parameters used in the experiments. More specifically, the *n* represents the time horizon, *EDDIE8 period* is the technical indicator's parameterized period range, and the *w1*, *w2* and *w3*, are weights assigned to RC, RMC and RF respectively.

Additionally, the experiment design is separated into two parts. In the first part, we examined the experimental parameter values of the previous meta-heuristics [14] (which will be denoted as ED8_SA, ED8_TS, ED8_GLS respectively) and tested if calibrating these meta-heuristic parameters would produce further improvement.

Firstly, two parameters of the SA called *temperature* and *kmax* were examined. The former affects the probability of acceptance of SA, which has a significant role to determine if a solution has been accepted. The latter controls the maximum number of iteration allowed to explore the search space of the periods of the VarConstructor per time. We chose different values of *temperature* and the *kmax* using Smonou et al.'s values (*temperature*: 0.9, *kmax*: 8) as benchmarks. The values used are 0.5, 0.7, 1, 1.2 for the *temperature*, and 7, 8, 10, 12, 15, 20 for *kmax*.

Furthermore, the *tabu list* and *aspiration criteria* of TS were examined. The former keeps track of the memory size of neighbors that the solutions have currently visited without improving and the latter controls the memory size of the neighbors that contain promising solution. We chose higher and lower values of *tabu list* and the *aspiration criteria* using Smonou et al.'s value (*tabu list*: [-2, +2], *aspiration criteria*: [-1, +1]) as a benchmark. We examined the *tabu list* parameter values with neighborhood size of (-3, +3), (-4, +4), (-5, +5) and *aspiration criteria* parameter values with neighborhood size of (-1, +1), (-2, +2), (-3, +3), (-4, +4).

Moreover, the *kmax* of GLS, which controls the maximum number of iteration, was examined. Higher and lower values of *kmax* were chosen using Smonou et al. value (*kmax*: 10) as a benchmark. The values varied from 4, 8, and 15. These were selected in order to investigate their influence on the algorithm's performance, taking into consideration the computational time.

Finally, the experiment parameter tuning showed that for the SA the combination of *temperature* value of 1 and *kmax* of 15 gave the best results. Regarding the TS algorithm, the combination of *tabu list* of (-4, +4) and *aspiration criteria* of (-1, +1) gave the best results. Lastly, the GLS with *kmax* of 8 gave the best results in terms of the average ranking of all metrics. Thus the above values were selected to be used in our experiments.

For the second and the main part of the experiments, we examined if the introduction of the Choice Function would bring further improvement to the best-performing version of the meta-heuristics. We experimented with higher and lower values of α, β, and δ, which are the weights assigned to the measure score of the Choice Function (*f1, f2, f3* respectively) where α and β intensify the search and δ diversifies the search as mentioned in Section IIIA. We used values ranging between 0.03 to 1 for α, β, and δ respectively. The reason we chose these values for α, β and δ was because of our preference to balance exploitation with exploration. For instance, even if a particular meta-heuristic is performing well, we still want the program to occasionally explore other meta-heuristics. From our experiments we have concluded that parameters of $\alpha = 0.5$, $\beta = 0.3$ and $\delta = 1.0$ produced the best results in our preliminary experiments; thus, we have decided to give more emphasis on diversification (since δ diversifies the search). The merit of having this Choice function parameter setup prevents the search from getting stuck to only well-performing meta-heuristics.

V. RESULTS

This chapter presents and discusses the results from our experiments on 9 datasets over 50 runs. For the purpose of this paper, we considered the best [6] and average for each of the performance measures (Fitness, RC, RMC, and RF).

Tables III and IV illustrate the average and best results of the application of the hyper-heuristic to EDDIE8, in comparison to the versions of EDDIE8 with each parameter-amended meta-heuristic. We have chosen to compare the selected hyper-heuristic with the individual meta-heuristics that were picked during our experiments with different parameters as mentioned in Section IV.

Table III
AVERAGE RESULTS HYPER-HEURISTIC

Dataset	Heuristics	Fitness	RC	RMC	RF
Easyjet	ED8_SA	**0.1220**	**0.4600**	**0.6550**	**0.2960**
	ED8_TS	**0.1300**	**0.4680**	**0.6380**	0.2890
	ED8_GLS	0.1510	0.4890	0.5910	0.2790
	Hyper-heuristic	0.1416	0.4820	0.5930	0.2940
First	ED8_SA	**0.1480**	0.4900	0.5400	**0.3060**
	ED8_TS	0.1560	0.4980	0.5310	**0.3000**
	ED8_GLS	0.1510	0.4940	0.5180	**0.3110**
	Hyper-heuristic	0.1482	0.4860	0.5570	0.2920
Hammerson	ED8_SA	**0.1190**	**0.5010**	**0.4940**	**0.4400**
	ED8_TS	**0.1270**	**0.5070**	**0.4590**	**0.4370**
	ED8_GLS	0.1520	0.5310	0.4240	0.4160
	Hyper-heuristic	0.1480	0.5290	0.4380	0.4180
HSI	ED8_SA	**0.2650**	**0.6260**	0.1990	**0.3030**
	ED8_TS	**0.2510**	**0.6100**	**0.2290**	**0.3060**
	ED8_GLS	**0.2600**	**0.6190**	**0.2220**	0.2990
	Hyper-heuristic	0.2674	0.6290	0.1990	0.3010
Imp	ED8_SA	0.2090	0.5690	**0.4820**	0.2820
	ED8_TS	0.2100	0.5720	0.4610	0.2900
	ED8_GLS	**0.1970**	**0.5610**	0.4520	**0.3140**
	Hyper-heuristic	0.2029	0.5650	0.4700	0.2960
Marks_Spencer	ED8_SA	0.1390	0.5020	0.4840	0.3790
	ED8_TS	0.1300	0.4950	0.4780	0.3950
	ED8_GLS	0.1280	0.4920	0.4910	0.3920
	Hyper-heuristic	0.1223	0.4860	0.4950	0.4000
MDAX	ED8_SA	**0.1260**	**0.4970**	0.1940	0.5080
	ED8_TS	0.1340	0.5050	0.1820	0.5030
	ED8_GLS	**0.1210**	**0.4960**	**0.2330**	0.5100
	Hyper-heuristic	0.1266	0.4990	0.2010	0.5080
NASDAQ	ED8_SA	**0.1780**	**0.5300**	**0.4390**	**0.3210**
	ED8_TS	**0.1690**	**0.5200**	**0.4590**	**0.3250**
	ED8_GLS	**0.1730**	**0.5250**	**0.4400**	**0.3260**
	Hyper-heuristic	0.1874	0.5410	0.4120	0.3200
Next	ED8_SA	**0.1140**	**0.4650**	**0.5110**	**0.3820**
	ED8_TS	0.1240	0.4760	0.4930	**0.3750**
	ED8_GLS	0.1420	0.4950	0.4810	0.3580
	Hyper-heuristic	0.1238	0.4760	0.4990	0.3730

For the results presented in both Tables III and IV, an improvement between the Hyper-heuristic and the rest of the algorithms' results is denoted in **bold**. For example, if the Hyper-heuristic has provided better solution than the ED8_SA algorithm, then the ED8_SA solution will be denoted in **bold**. This can be seen in Table III for instance, in the Easyjet dataset, the Hyper-heuristic has improved the Fitness comparing to both ED8_SA and ED8_TS algorithms.

Furthermore, as we can see from Table III, the average results of the hyper-heuristics have done quite well, improving the meta-heuristics average results in a total of 55 instances (denoted in bold). In addition, some improvements are quite important, for instance the RMC figures of Easyjet's ED8_SA, and ED8_TS, Hammerson's ED8_SA, MDAX's ED8_GLS and NASDAQ's ED8_TS, where the improvements are in the scale of 3 - 6%.

Table IV
BEST RESULTS HYPER-HEURISTIC

Dataset	Heuristics	Fitness	RC	RMC	RF
Easyjet	ED8_SA	**0.1530**	**0.4830**	**0.6550**	0.2530
	ED8_TS	0.2210	0.5700	**0.4480**	0.2390
	ED8_GLS	**0.0695**	**0.4070**	**0.7590**	**0.3290**
	Hyper-heuristic	0.1890	0.5400	0.4090	0.3140
First	ED8_SA	0.2520	0.6100	0.2510	**0.3780**
	ED8_TS	**0.0663**	**0.4070**	**0.6430**	0.2950
	ED8_GLS	0.1490	0.4930	0.4730	0.3310
	Hyper-heuristic	0.1090	0.4500	0.5650	0.3480
Hammerson	ED8_SA	**0.0898**	**0.4700**	**0.5210**	0.4180
	ED8_TS	**0.1100**	0.4970	**0.6210**	**0.4670**
	ED8_GLS	0.2130	0.5930	0.3020	0.3760
	Hyper-heuristic	0.1320	0.5030	0.3550	0.4490
HSI	ED8_SA	**0.2750**	**0.6370**	**0.1070**	0.3080
	ED8_TS	**0.1850**	**0.5330**	**0.4290**	**0.3220**
	ED8_GLS	0.3140	0.6830	**0.0537**	0.3020
	Hyper-heuristic	0.3150	0.6830	0.0049	0.3150
Imp	ED8_SA	**0.2250**	**0.5930**	**0.4000**	0.3020
	ED8_TS	**0.2280**	**0.5970**	**0.3890**	0.3020
	ED8_GLS	**0.1610**	**0.5270**	**0.4650**	**0.3610**
	Hyper-heuristic	0.2430	0.6130	0.2050	0.3470
Marks_Spencer	ED8_SA	0.1760	0.5400	0.3440	0.3710
	ED8_TS	0.1590	0.5230	**0.4350**	0.3780
	ED8_GLS	**0.1240**	**0.4870**	**0.4680**	0.4040
	Hyper-heuristic	0.1430	0.5030	0.3820	0.4040
MDAX	ED8_SA	0.1960	0.5800	0.1630	0.4690
	ED8_TS	0.1990	0.5670	0.0068	0.4530
	ED8_GLS	**-0.0005**	**0.3930**	**0.5510**	**0.6050**
	Hyper-heuristic	0.1610	0.5470	0.2450	0.4740
NASDAQ	ED8_SA	**0.1570**	**0.5070**	**0.4680**	**0.3140**
	ED8_TS	**0.1930**	**0.5470**	**0.4030**	**0.3350**
	ED8_GLS	0.2200	0.5770	0.3530	**0.3010**
	Hyper-heuristic	0.2180	0.5730	0.3830	0.2910
Next	ED8_SA	0.1940	0.5500	0.4040	**0.3350**
	ED8_TS	**0.1570**	**0.5100**	0.4800	0.3180
	ED8_GLS	0.2180	0.5730	0.4290	0.2760
	Hyper-heuristic	0.1710	0.5230	0.4800	0.3180

Furthermore, the best results in Table IV show further significant improvements. More specifically, we can observe that, with the exception of the Next dataset, *hyper-heuristics have managed to improve at least three metrics for at least one meta-heuristic, for each dataset.* This is very important, because it implies that the hyper-heuristics framework can take advantage of the benefits of the different meta-heuristics

[6] The best GDT means selecting the best tree in terms of training data (out of all 50 individual runs), and reporting its performance (fitness, RC, RMC, RF) in the testing data. This has practical value, because in real-life a trader would not have access to the test data (unseen data); thus we pick the best tree from training and use it. We use that tree and check how well it performs in the unseen data.

and *be applicable to a wide range of datasets*. Furthermore, the best results were impressively improved for certain datasets; for instance, in *NASDAQ the hyper-heuristic managed to improve all metrics of the SA and TS*. In addition, hyper-heuristics have introduced several significant improvements in terms of the RMC best results. The most notable improvements are in the HSI dataset by 42% for ED8_TS, 10% for ED8_SA, and 5% for the ED8_GLS, in the Imp by 20% for ED8_SA, 18% for ED8_TS, and 26% for ED8_GLS and in the Easyjet by 25% for ED8_SA, 4% for ED8_TS and by 35% for ED8_GLS. Therefore, we can argue that the addition of the hyper-heuristic was proven quite beneficial for EDDIE8's best results especially in terms of the RMC. The immediate implication of having low RMC is that the algorithm will be able to discover a greater amount of buy opportunities, thus increase an investor's profit chances.

Moreover, the Friedman non-parametric test presents the ranking results of all metrics, for Best and Average results, in Tables V, VI, VII, and VIII. As we can observe, hyper-heuristics rank first [7] in the majority of the Best results (Fitness, RC, RMC), and also rank first in terms of Average RF. This is very important, because it confirms the beneficial performance of the framework, which we reported earlier in this section.

Subsequent analysis on the Holm post-doc test [24] [25] did not show a statistical significance in terms of the above results. However, this should not alarm us, because the fact remains that the hyper-heuristic framework was ranked first across the majority of the Best Results tests. As we mentioned earlier, it is very important for an algorithm to be performing well in terms of Best Results, because of its real-life value: a trader in real-life would run the forecasting algorithm multiple times and then use the best-performing tree (trading strategy). Thus, having better best trees can lead to *an increase to the trader's profit margin*.

The above discussion allows us to argue that the introduction of the hyper-heuristics has made EDDIE8 a robust algorithm. In addition, EDDIE8 would be expected to produce even better results if more meta-heuristics were included in the framework, and this is something we intend to further investigate. Lastly, as we have seen from the results (especially in terms of Best), the automated selection process of the meta-heuristics led to a broader applicability of the EDDIE algorithm, as it was able to introduce multiple improvements in the performance metrics, without being affected by the dataset that was used.

Table V
Friedman Ranking – Best and Average Results Fitness

Algorithm	Ranking Best Results	Ranking Average Results
ED8_SA	2.556	2.778
ED8_TS	2.667	**2.333**
ED8_GLS	2.556	2.444
Hyper-heuristic	**2.222**	2.444

[7] Results with lower ranking denote a better overall performance.

Table VI
Friedman Ranking – Best and Average Results RC

Algorithm	Ranking Best Results	Ranking Average Results
ED8_SA	2.444	2.778
ED8_TS	2.778	**2.333**
ED8_GLS	2.500	2.444
Hyper-heuristic	**2.278**	2.444

Table VII
Friedman Ranking – Best and Average Results RMC

Algorithm	Ranking Best Results	Ranking Average Results
ED8_SA	2.333	3.000
ED8_TS	2.944	2.444
ED8_GLS	2.667	**2.000**
Hyper-heuristic	**2.056**	2.556

Table VIII
Friedman Ranking – Best and Average Results RF

Algorithm	Ranking Best Results	Ranking Average Results
ED8_SA	**2.333**	2.778
ED8_TS	2.444	2.556
ED8_GLS	2.444	2.444
Hyper-heuristic	2.778	**2.222**

VI. CONCLUSION

This paper presented work on the application of a hyper-heuristic framework to 3 meta-heuristics previously applied on a Genetic Programming Financial Forecasting algorithm called EDDIE8. EDDIE8 allows the GP to search in the space of technical indicators for solutions, instead of using pre-specified ones, as happens in other works in the literature and also in the industry. However, a consequence of this is that EDDIE8's search area is quite large, leading to occasionally missed solutions due to ineffective search.

In order to address this issue, we applied a Choice Function hyper-heuristic to Simulated Annealing, Tabu Search and Guided Local Search, which were meta-heuristics that in the past had individually been applied to the period nodes of EDDIE8's trees [14]. Results showed that the algorithm's performance was improved in terms of best results with the most impressive being the RMC, thus *proving that the combination of Genetic Programming and hyper-heuristics is valuable for Financial Forecasting*. In terms of the best results, the improvement was very important, due to the fact that an investor, who would use the best tree of those

experiments, *could experience an outstanding boost of his profit*. Additionally, the enhancement of the RMC results is proof that our algorithm is more competitive in terms of identifying more trading opportunities; therefore it would be preferred by an investor who would like to *ensure that most buy opportunities are captured*.

Furthermore, this approach has *automated the meta-heuristic selection process*, as the investor has no concerns about which meta-heuristics to use every time. As we saw, this led to a broader applicability of the EDDIE algorithm, as the improvements that were introduced were not dependent on the given dataset.

The fact that the hyper-heuristic tested in this paper has improved EDDIE's performance is very promising. Our future research objectives are to investigate the possibility of dynamically updating the values of the Choice Function parameters, to introduce a re-enforcement learning scheme to the Choice Function parameters, to apply the same framework in more datasets, and last but not least to experiment with other hyper-heuristics framework implementation**.**

REFERENCES

[1] F Allen and R Karjalainen, "Using genetic algorithms to find technical trading rules," *Journal of Financial*, vol. 51, pp. 245–271, 1999.

[2] A Kablan, "Adaptive neuro fuzzy inference systems for high frequency," , Sliema, 2009, pp. 105-110.

[3] Y Chen, S Muba, K Hirasawa, and J Hu, "Genetic network programming with sarsa learning and its application to creating stock trading rules," , Singapore, 2007, pp. 220–237.

[4] S Schulenburg and P Ross, "Explorations in LCS Models of Stock Trading," in *Advances in Learning Classifier Systems,4th International Workshop, IWLCS 2001*, P.L. Lanzi, W. Stolzmann, and S.W. Wilson, Eds. Berlin: Springer Berlin Heidelberg, 2002, vol. 2321, pp. 150-179.

[5] AP Refenes, *Neural Networks in the Capital Markets*. New York: John Wiley & Sons, Inc., 1994.

[6] N Baba and M kozaki, "An intelligent forecasting system of stock price using neural networks," , vol. 1, Baltimore, MD, 1992, pp. 371 - 377.

[7] E Tsang, P Yung, and J Li, "EDDIE-Automation, a decision support tool for financial forecasting," *Journal of Decision Support Systems*, vol. 37, no. 4, pp. 559– 565, 2004.

[8] J Li and E Tsang, "Improving technical analysis predictions: An application of genetic programming," , USA, 1999, pp. 108–112.

[9] E Tsang and J Li, "EDDIE for financial forecasting," in *Genetic Algorithms and Genetic Programming in Computational Finance*, New York, 2002, pp. 161–174.

[10] J Koza, *Genetic Programming II: Automatic Discovery of Reusable Programs*. Cambridge: MIT, 1994.

[11] J Koza, *Genetic Programming: On the programming of computers by means of natural selection*. Cambridge: MIT Press, 1992.

[12] M Kampouridis, "Computational Intelligence in Financial Forecasting and Agent-Based Modeling: Applications of Genetic Programming and Self-Organizing Maps," *PhD Thesis*, 2011.

[13] M Kampouridis and E Tsang, "EDDIE for Investment Opportunities Forecasting: Extending the Search Space of the GP," *Proceedings of the IEEE Congress on Evolutionary Computation*, pp. 2019-2026, 2010.

[14] D Smonou, M Kampouridis, and E Tsang, "Metaheuristics Application on a Financial Forecasting Problem," in *IEEE Congress on Evolutionary Computation*, Cancun, Mexico, 2013, pp. 1021-1028.

[15] E Tsang et al., "EDDIE in financial decision making," *Journal of Management and economics*, p. 4, 2000.

[16] J Li and E Tsang, "Reducing failures in investment recommendations using genetic programming," , Barcelona, 2000.

[17] E Tsang, S Markose, and H Er, "Chance discovery in stock index option and future arbitrage," *New Mathematics and Natural Computation*, vol. 1, no. 3, pp. 435-447, 2005.

[18] F Glover and E Taillard, "A user's guide to tabu search," *Annals of Operations Research*, vol. 41, no. 1, pp. 1-28, 1993.

[19] L Davis, *Genetic algorithms and simulated annealing*.: Pitman, 1987.

[20] C Voudouris and E Tsang, "Guided Local Search and its application to the Travelling Salesman problem"," *European Journal of Operational Research* , vol. 133, pp. 469-499 , 1999.

[21] P Cowling, G Kendall, and E Soubeiga, "Hyperheuristic: A tool for rapid prototyping in scheduling an optimisation," *Second European Conference on Evolutionary Computing for Combinatorial Optimisation (EvoCop)*, pp. 1–10, 2000.

[22] P Rattadilok, "An Investigation and Extension of a Hyper-heuristic Framework," *Informatica: An International Journal of Computing and Informatics*, vol. 34, no. 4, pp. 523-534, 2010.

[23] M Kampouridis, A Alsheddy, and E Tsang, "On the investigation of hyper- heuristics on a financial forecasting problem," *Annals of Mathematics and Artificial Intelligence*, 2012.

[24] S Garcia and F Herrera, "An Extension on Statistical Comparisons of Classifiers over Multiple Data Sets for all Pairwise Comparisons," *Journal of Machine Learning Research*, vol. 9, no. 66, pp. 2677-2694, 2008.

[25] J Demšar, "Statistical comparisons of classifiers over multiple data sets," *The Journal of Machine Learning Research*, vol. 7, pp. 1-30, 2006.

[26] M Friedman, "A Comparison of Alternative Tests of Significance for the Problem of m Rankings," *The Annals of Mathematical Statistics*, vol. 11, no. 1, pp. 86-92, 1940.

[27] E Tsang, J Li, and J Butler, "EDDIE beats the bookies," *International Journal of Software*, pp. 1033–1043, 1998.