An Initial Investigation of Choice Function Hyper-Heuristics for the Problem of Financial Forecasting

Michael Kampouridis
School of Computing, University of Kent
Medway, Chatham, ME4 4AG, UK
Email: m.kampouridis@kent.ac.uk

***Abstract*—Financial forecasting is a vital area in computational finance. This importance is reflected in the literature by the continuous development of new algorithms. EDDIE is well-established genetic programming financial forecasting tool, which has successfully been applied to a variety of international datasets. Recently, we introduced hyper-heuristics to EDDIE. This was the first time in the literature that hyper-heuristics were used for financial forecasting. Results showed that this introduction significantly benefited the performance of the algorithm. However, an issue was encountered in the way that low-level heuristics were selected during the search process, because it was considered to be a static way. To address this issue, in this paper we further improve our algorithm by introducing a Choice Function, which is a score based technique that offers a more dynamic selection of the low-level heuristics. This paper presents preliminary results, after having tested the Choice Function approach with 10 datasets. These results show that the introduction of the Choice Function is beneficial to EDDIE, thus making it a very promising tool for future investigation on financial forecasting problems.**

I. Introduction

Financial forecasting is an important area in computational finance [1]. There are numerous works that attempt to forecast the future price movements of a stock; several examples can be found in [2], [3]. Recently, we presented EDDIE 8-HH (ED8-HH) [4], [5], an extended version of a well-established genetic programming financial forecasting algorithm named EDDIE [6], [7], [8]. In this new version, *we were the first, to the best of our knowledge, to introduce hyper-heuristics into a financial forecasting problem*. Hyper-heuristics is a well-known method that has been used in a variety of search and optimization problems [9], such as transportation [10], scheduling [11], and timetabling [12], and has returned very promising results. The application of hyper-heuristics to EDDIE 8 was also a great success. Nevertheless, a limitation of our approach in [4], was in the way the low-level heuristics were selected, and particularly the weight updating scheme that was responsible for the selection of the heuristics. This was because the weights were updated by the same, predefined percentage, every time a heuristic was selected.

In this paper, a more *dynamic approach* is followed, which guides the selection of the low-level heuristics during the search process. We use the so-called Choice Function [13], [14], which not only rewards the performance of individual heuristics, but also the performance of heuristics as successors of previously invoked ones. In addition, the Choice Function has the advantage of being able to both intensify and diversify the search, based on certain conditions that will be explained later in this paper. As a result, the process of selecting different low-level heuristics is improved, leading to better exploration and exploitation of the search space, and therefore maximizing the advantages of the hyper-heuristics framework. In order to demonstrate the Choice Function's effectiveness, we present preliminary results for 10 different datasets and compare the results against the previous version of the EDDIE algorithm. Demonstrating the above is important, because of the continuous efforts of both the industrial and scientific society, for the continuous development of new and improved financial forecasting algorithms.

The rest of this paper is organized as follows: first of all, in order to make it clearer to the reader, we present the EDDIE 8 algorithm by itself (i.e. without hyper-heuristics) in Section II. Then, Section III presents the low-level heuristics that are going to be part of the hyper-heuristics framework, to which the Choice Function will be applied. Section IV presents the hyper-heuristics framework, and particularly describes the Choice Function and explains how it controls the selection of the low-level heuristics. Section V presents the experimental setup, and Section VI presents and discusses the experimental results. These compare EDDIE 8's performance in the following conditions:

- a hyper-heuristics framework that uses [4]'s static way of selecting the low-level heuristics, and
- a hyper-heuristics framework that uses the Choice Function as the selection method of the low-level heuristics.

Finally, Section VII concludes this paper and also discusses future work.

II. The EDDIE 8 algorithm

EDDIE 8 (ED8) is a Genetic Programming (GP) [15], [16] financial forecasting algorithm, which learns and extracts knowledge from a set of data. The kind of question ED8 tries to answer is 'will the price increase within the n following days by $r\%$'? The user first feeds the system with a set of past data; EDDIE then uses this data and through a GP process, it

produces and evolves Genetic Decision Trees (GDTs), which make recommendations of buy (1) or not-to-buy (0).

The set of data used is composed of three parts: (i) daily closing price of a stock, (ii) a number of attributes, and (iii) signals. Stocks' daily closing prices can be obtained online on websites such as `http://finance.yahoo.com` and also from financial statistics databases like *Datastream*. The attributes are indicators commonly used in technical analysis [17]; which indicators to use depends on the user and his belief of their relevance to the prediction. The technical indicators that are used in this work are: Moving Average (MA), Trade Break Out (TBR), Filter (FLR), Volatility (Vol), Momentum (Mom), and Momentum Moving Average (MomMA).[1]

The signals are calculated by looking ahead of the closing price for a time horizon of *n* days, trying to detect if there is an increase of the price by *r%* [7]. For this set of experiments, *n* was set to 20 and *r* to 4%. In other words, the GP is trying to use some of the above indicators to forecast whether the daily closing price is going to increase by 4% within the following 20 days.

After feeding the data to the system, EDDIE creates and evolves a population of GDTs. Figure 1 presents the Backus Normal Form (BNF) [21] (grammar) of ED8. As we can see, the root of the tree is an If-Then-Else statement. The first branch is either a boolean (testing whether a technical indicator is greater than/less than/equal to a value), or a logic operator (and, or, not), which can hold multiple boolean conditions. The 'Then' and 'Else' branches can be a new GDT, or a decision, to buy or not-to-buy (denoted by 1 and 0).

<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, [MinP, MaxP]
Decision is an integer, Positive or Negative implemented
Threshold is a real number

Fig. 1. The Backus Normal Form of ED8

As we can observe from the grammar in Figure 1, there is a function called *VarConstructor*, which takes two children. The first one is the indicator, and the second one is the Period. Period is an integer within the parameterized range [MinP, MaxP] that the user specifies. The advantage of this approach is that ED8 is not constrained to pre-specified periods, as is usually the case in industry.[2] As a consequence, it is up to the GP and the evolutionary process to look for the optimal periods from the period range provided. For instance, if this range is 2 to 65 days, then ED8 can create Moving Averages with any of these periods, e.g., 12 days MA, 15 days MA, and so on. Furthermore, the periods are leaf nodes and are thus subject to genetic operators, such as crossover and mutation. A sample GDT of ED8 is presented in Figure 2. As we can see, if the 12 days MA is less than 6.4, then the user is advised to buy; otherwise, the user is advised to consult another GDT, which is located in the third branch ("else-branch") of the tree. As explained, the periods 12 and 50 of the figure's sample tree are leaf nodes; the advantage of this being that the GP can replace them with other, more effective periods, which might have come up during the evolutionary process.

Depending on the classification of the predictions, we can have four cases: True Positive (TP), False Positive (FP), True Negative (TN), and False Negative (FN). As a result, we can use the metrics presented in Equations 1, 2 and 3.

Rate of Correctness

$$RC = \frac{TP + TN}{TP + TN + FP + FN} \tag{1}$$

Rate of Missing Chances

$$RMC = \frac{FN}{FN + TP} \tag{2}$$

Rate of Failure

$$RF = \frac{FP}{FP + TP} \tag{3}$$

The above metrics combined give the following fitness function, presented in Equation 4:

$$ff = w_1 * RC - w_2 * RMC - w_3 * RF \tag{4}$$

where w_1, w_2 and w_3 are the weights for RC, RMC and RF respectively. These weights are given in order to reflect the preferences of investors. For instance, a conservative investor would want to avoid failure; thus a higher weight for RF should be used. For the experiments of this paper, the focus is on strategies that mainly target correctness and reduced failure. Thus these weights have been set to 0.6, 0.1 and 0.3 respectively.

The fitness function is a constrained one, which allows EDDIE to achieve lower RF. The effectiveness of this constrained fitness function has been discussed in [8], [22]. The constraint is denoted by R, which consists of two elements represented by a percentage, given by

$$R = [C_{min}, C_{max}],$$

[1]We use these indicators because they have been proved to be quite useful in developing GDTs in previous works like [18], [19] and [20]. Of course, there is no reason not to use other information like fundamentals or limit order book. However, the aim of this work is not to find the ultimate indicators for financial forecasting.

[2]In the literature, the users of similar algorithms pre-specify certain periods that they consider useful. For instance, **12** days MA, and **50** days MA. The indicators (e.g., MA) together with their respective period (e.g., 12) are treated by the GP as a single leaf node. Thus, the numbers 12 and 50 cannot change during the evolutionary process. In our previous work [6], we questioned this approach, because nobody can guarantee that, for instance, a 12 days MA is better than a 15 days MA. To address this issue, we created ED8, which is able to search in the space of technical indicators and their periods.

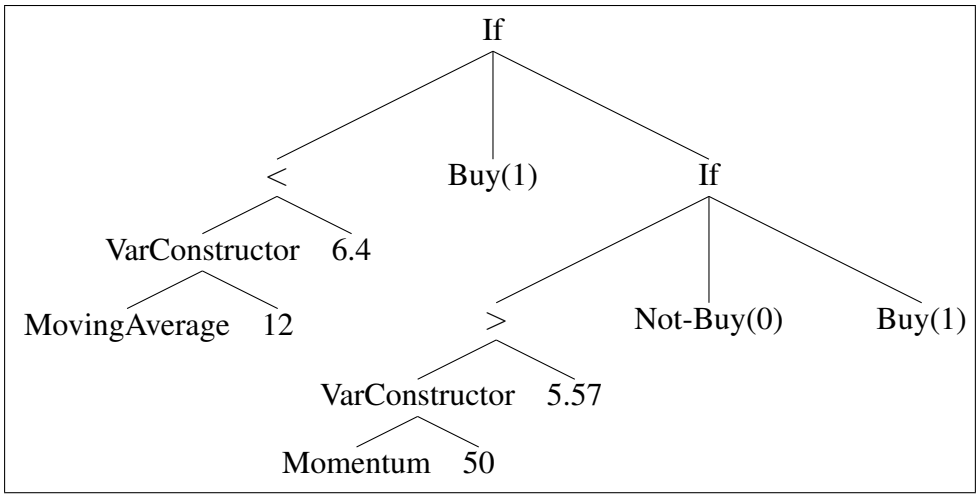

Fig. 2. Sample GDT generated by ED8.

where $C_{min} = \frac{P_{min}}{N_{tr}} \times 100\%$, $C_{max} = \frac{P_{max}}{N_{tr}} \times 100\%$, and $0 \leq C_{min} \leq C_{max} \leq 100\%$. N_{tr} is the total number of training data cases, P_{min} is the minimum number of positive position predictions required, and P_{max} is the maximum number of positive position predictions required.

Therefore, a constraint of $R = [50, 65]$ means that the percentage of positive signals that a GDT predicts[3] should fall into this range. When this happens, then w_1 remains as it is (i.e. 0.6 in the experiments of this paper). Otherwise, w_1 takes the value of zero.

This concludes this short presentation of ED8. The next section presents the low-level heuristics that are applied to ED8 as part of the hyper-heuristics framework.

III. Heuristics

Although ED8 was characterized as an effective financial forecasting algorithm [23], [6], an issue that arose, as a result of its extended grammar, was that ED8's search space was significantly bigger.[4] As a consequence, ED8 could sometimes miss good solutions due to ineffective search. To address this, hyper-heuristics frameworks were introduced in [4], [5], which consisted of different low-level heuristics. The rest of this section describes such heuristics, which are going to be used for the experiments of this paper.

[3]As already mentioned, each GDT makes recommendations of buy (1) or not-to-buy (0). The former denotes a positive signal and the latter a negative. Thus, within the range of the training period, which is t days, a GDT will have returned a number of positive signals.

[4]In order to understand the kind of search space ED8 is dealing with, let us give an example: since ED8 is using 6 technical indicators, with periods ranging from 2 to 65 days, the total number of the potential indicators is $6 \times 64 = 384$. If a given GP tree can take up to a maximum of k indicators (this depends on the depth of the particular tree), then the permutations of the available indicators are 384^k. To put this in perspective, let us also calculate the permutations of a GP that would not use a range of periods, as ED8 does. Thus, if for instance the GP was using 6 technical indicators with 2 periods each (this is a quite common approach [27], [28]), then the number of permutations would be $(2 \times 6)^k = 12^k$. We can thus see that the number of permutations for ED8 is significantly higher, thus making it extremely difficult to thoroughly search this search space.

In EDDIE 8, each GP individual represents a possible GDT whose basic component is the variable constructor. As explained above, an issue that ED8 was facing was its big search space, which was a direct result of the increased number of period choices. Therefore, the objective was to design heuristics that improve a given GDT by exploring the space of its periods.

Based on [4], it was found that a very effective hyper-heuristic framework is one that includes a combination of random mutators and hill climbers. These can take one of the following three approaches:

- A random mutation: this approach makes a random change either to the indicators or the periods of the current GDT, resulting in new GDT(s). It compares the new GDT(s) with the original GDT, and returns the best one.
- An iterative hill climbing: this is a local search procedure that iteratively searches the local space (i.e. neighborhood) of the current GDT. A neighbor can be obtained from the current solution by making a small change to its structure. The procedure starts from an initial solution, and then iteratively moves to a better neighbor. The search stops when none of the neighbors yields an improvement to the current tree, returning a local optimum GDT.
- A single-step hill climbing: this approach is similar to the previous hill climber, however, the search examines only the neighbourhood of the initial GDT. It stops once a better neighbor is found, or the neighborhood is examined completely without improvement.

Based on the above approaches, the following period-based heuristics are devised:

1) Period-based Mutation ($PMut$). This heuristic mutates the current GDT by trying two new periods for a randomly picked variable constructor. The two periods are obtained by adding/subtracting a pre-set value (k) from the current period, resulting in two new GDTs. The hyper-heuristics framework used in this paper utilizes

3 values of k: 11 ($PMut11$), 13 ($PMut13$), and 15 ($PMut15$).[5]

2) Period-based hill climbing (PHC). This is an iterative hill climbing procedure. In PHC, the neighbourhood includes any GDT that can be obtained by modifying the period of a variable constructor in the current tree. Here, this modification is defined as a marginal change (k) to the value of period, such that $-10 \leq k \leq 10$.
3) Period-based single-step hill climbing ($sPHC$). This is the single-step hill climbing version of PHC.

Hence, the hyper-heuristics framework that is going to be used in this paper consists of the above 5 low-level heuristics, i.e., $PMut11$, $PMut13$, $PMut15$, PHC, and $sPHC$.

IV. Hyper-heuristics Framework

This section presents the hyper-heuristics frameworks that are going to be tested in this paper. First, we present the details of the framework that was introduced in [4], and explain how the low-level heuristics are selected. This framework will act as a benchmark for the experiments. Then we present the Choice Function, which is the newest addition to the ED8 algorithm, and the main contribution of this paper. We start by presenting the details of the Choice Function, and then we continue by explaining how it is incorporated into the hyper-heuristics framework, and thus how it controls the selection of the low-level heuristics.

A. Hyper-heuristics Framework 1 (ED8-HH)

In this simple framework, which from now on will be referred to as ED8-HH, all low-level heuristics are used simultaneously. Inspired by the Population Based Incremental Learning algorithm [24] and the similar Estimation of Distribution Algorithms [25], [26], all low level heuristics are initially given a weight w of being selected, where $w = \frac{1}{\#heuristics}$. Thus, if a framework consists of 5 heuristics, then the initial weight of each heuristic is $\frac{1}{5}$. Then depending on the result on the performance of a tree after the implication of a heuristic, the following cases can occur: increase in performance, no change in performance, decrease in performance. Depending on the case, there is a different reward/punishment for the respective heuristic. This is denoted by r. Thus, the weight w is updated as follows:

1) Increase in performance
   $w = w_0 + r$
2) No change in performance
   $w = w_0 - r/5$
3) Decrease in performance
   $w = w_0 - r$

The highest reward is offered when the selected heuristic has offered an increase in the performance of the respective tree. In the case of no change in performance, the selected heuristic is slightly penalized by a decrease in its weight by $r/5$. Lastly, there is a punishment of r in the case of decrease in the performance.

As explained already, the above mentioned way of updating the weights can be considered rather static, because these updates are always related to the same, pre-defined r. Thus, it was important to look for a more dynamic way of selecting the low-level heuristics. The suggested solution to this is the Choice Function, the details of which are presented next.

B. The Choice Function

Choice Function heuristic selection methods were first introduced in [13]. These methods guid the selection of low-level heuristics during the search process, by introducing score based techniques. A detailed explanation of the implementation of the Choice Function in a hyper-heuristics framework is presented in [14]. The implementation in this paper is based on this work ([14]).

More specifically, the Choice Function focuses on 3 aspects:

1) How well a given heuristic has performed individually (Factor f_1 - Equation 5)
2) How well a given heuristic has performed as a successor of a previously invoked heuristic (Factor f_2 - Equation 6)
3) The elapsed time since a given heuristic was called (Factor f_3 - Equation 7)

The Choice Function thus ranks the low-level heuristics with respect to a combined score of the above aspects. It should also be mentioned that the first two aspects intensify recent performance, while the last one offers diversification in the search.

1) Calculating the Factors in the Choice Function:

a) Factor f_1: First of all, the Choice Function is interested in the performance of each low-level heuristic. Thus, a measure of the past performance of a low-level heuristic H_j is equal to:

$$f_1(H_j) = \sum_{n=1}^{l} \alpha^n \left(\frac{I_n(H_j)}{T_n(H_j)}\right) \tag{5}$$

In the above equation, $I_n(H_j)$ is the change in the fitness function the n^{th} last time H_j was used, and l refers to the first time that H_j was selected. Furthermore, $T_n(H_j)$ is the amount of CPU time in milliseconds from the time H_j was used the n^{th} last time, until the moment it returned a solution to the controller. The parameter α takes a value in the interval $[0, 1]$ and it assigns a decreasing geometric sequence of weights to the past performance measures of H_j.

The idea above is that if a certain low-level heuristic has recently been effective, it is possible that it will continue being effective in the near future.

b) Factor f_2: This factor takes into account the performance of the current heuristic as a function of the heuristic that was used immediately before. This is because, the performance of a low-level heuristic may be affected by the low-level heuristic that was used immediately before it. Suppose that H_k was used at the last iteration and the use of H_j next is

[5] In our previous work, other k values were tested, too. However, the ones presented here were found very useful and were included in the framework we are presenting.

being considered. Then the measure of the past performance of the pair (H_k, H_j) is calculated using:

$$f_2(H_k, H_j) = \sum_{n=1}^{l} \beta^n \left(\frac{I_n(H_k, H_j)}{T_n(H_k, H_j)}\right) \quad (6)$$

Here $I_n(H_k, H_j)$ is the change in the cost function the n^{th} last time the pair (H_k, H_j) was used, and l refers to the first time in the search that H_j was used immediately after H_k. $T_n(H_k, H_j)$ is the amount of CPU time in milliseconds from the time the pair (H_k, H_j) was used the n^{th} last time, until the time when a solution was returned to the controller. The parameter β takes a value in the interval $[0, 1]$ and it also assigns a decreasing geometric sequence of weights to the past performance measures of the pair (H_k, H_j).

c) Factor f_3: So far the factors f_1 and f_2 were presented. These two factors are important because they intensify the search on low-level heuristics which have performed well in the past. However, it is also important to have a factor that offers diversification. The third factor f_3 thus diversifies the search by considering low-level heuristics that may not have been used for some time. This is beneficial in situations where the search is stuck at a local optimum. Thus, it might be worth using another heuristic that has not been chosen recently during the search. This leads to a diversified search process. The value of f_3 is calculated for each low-level heuristic H_j using:

$$f_3(H_j) = \delta \times \tau(H_j) \quad (7)$$

Here $\tau(H_j)$ is the amount of CPU time in milliseconds since the low-level heuristic H_j was last used and each time H_j is used $\tau(H_j)$ is reset to 0. Parameter δ is the weight of f_3 and receives values between 0 and 1.

C. Incorporating the Choice Function into the Hyper-heuristics framework (ED8-CF)

Based on the above, the Choice Function is incorporated into the hyper-heuristics framework, which is from now on going to be referred to as ED8-CF. Thus, instead of using the selection method presented in Section IV-A, the low-level heuristics are selected based on the Choice Function F. The value of F is calculated by the sum of the 3 factors presented above:

$$F(H_j) = f_1(H_j) + f_2(H_k, H_j) + f_3(H_j) \quad (8)$$

The process that is followed is:

- With a probability p, select a tree from the current population
- For a number of k iterations, do:
  - Calculate the value of F for each low-level heuristic
  - Select the low-level heuristic with the highest Choice Function F score (Equation 8)
  - Apply the low-level heuristic to a period of the selected tree
  - Update the factors f_1, f_2, and f_3, as described above, in Equations 5 - 7

The above process allows a number of different heuristics to be applied on the period branches of the tree, resulting to an improved fitness. As already mentioned, this approach has the advantage that the low-level heuristics are selected based on different factors, which are related to both intensified and diversified search criteria. This thus increases the chances of a better exploration and exploitation of the huge search space of EDDIE 8. Hence, it also increases the chances of improving the algorithm's performance.

V. Experimental Setup

We run tests for 10 datasets. These datasets consist of daily closing prices from 8 stocks from FTSE 100, and 2 international indices. The 8 FTSE 100 stocks are: Aggreko, Amlin, British American Tobacco (BAT), Carnival, Centrica, Easyjet, Next, and Xstrata. The 2 indices are: Athens Stock Exchange (Greece), and MDAX (Germany). The training period is 1000 days and the testing period 300.

We are interested in investigating the advantages of introducing the Choice Function to a hyper-heuristics framework. For this reason, we compare the Choice Function algorithm introduced in this paper (ED8-CF) with the previously developed hyper-heuristics framework [4] (ED8-HH). The goal is to report the differences in the performance of ED8 under each framework and comment on whether the introduction of the Choice Function is beneficial to EDDIE 8. This will take place in the next section, Section VI.

The GP parameters are presented in Table I. For statistical purposes, the GP is run 50 times. Thus, the process is as follows. We create a population of 500 GDTs, which are evolved for 50 generations, over a training period of 1000 days. At the last generation, the best performing GDT in terms of fitness is saved and applied to the testing period. As we have already said, this procedure is done for 50 individual runs.[6]

TABLE I
GP Parameters.

GP Parameters	
Max Initial Depth	6
Max Depth	8
Generations	50
Population size	500
Tournament size	2
Reproduction probability	0.1
Crossover probability	0.9
Mutation probability	0.01
Weight w_1	0.6
Weight w_2	0.1
Weight w_3	0.3
Period (ED8)	[2,65]

Furthermore, Table II presents the parameters of the Choice Function hyper-heuristics framework. The probability p of

[6]We do not argue that these are the optimal GP parameters. Nevertheless, experience from previous EDDIE experiments has shown that the above GP parameters return effective results.

applying hyper-heuristics to a certain tree is set for this work at 35%. Thus, 35% of the trees' periods can be updated through hyper-heuristics at each generation. It should be mentioned at this point that we experimented with higher and lower values of p and found that $p = 35\%$ was offering the best results. Similarly, the number of maximum iterations k that the low-level heuristics are applied to a tree is set to 30. We also experimented with $k = 15$ and $k = 50$, but found that $k = 30$ was offering the best results. Lastly, we have set the Choice Function parameters α, β and δ to 0.8, 0.5, and 1 respectively. We have set them in a way that reflects the aspects of the Choice Function we consider the most important during the search; for instance, experience has shown that the algorithm tends to get stuck in specific well-performing low-level heuristics. For this reason, we have decided to give emphasis on diversification and thus gave δ a value of 1.[7]

TABLE II
CHOICE FUNCTION HYPER-HEURISTIC PARAMETERS.

CF-HH Parameters	
Choice Function Probability p	35%
Maximum Iterations k per Tree	30
α	0.8
β	0.5
δ	1

VI. RESULTS

This section presents the experimental results. Results are presented in Tables III(a) and III(b). The former presents the average results, over the 50 individual runs, for the 10 datasets tested in this paper, under the performance metrics of Fitness, RC, RMC and RF. The first line for each dataset denotes results returned by ED8-HH, while the second line presents results returned by ED8-CF. Table III(b) presents the best[8] results, over the 50 runs. For example, the first entry in this table, denotes that the highest Fitness value that ED8-HH obtained, out of 50 individual runs for the dataset of Aggreko, was 0.342. We should emphasize at this point that *we are particularly interested in the best results*, because if an investor was using EDDIE to assist him with his investments, he would run the algorithm many times and then select the best tree that was produced. Thus, we are very interested in any improvements that might come up in terms of best results, because they have practical advantages in the real world.

A. Average results

Table III(a) presents the average results. We are interested in examining if ED8-CF has significantly improved any of the following: Fitness, RC, RMC, and RF. In order to determine if a difference is significant, we run a two tailed t-test at 5% significance level, under the null hypothesis that the two distributions have equal means and equal but unknown variances; the alternative hypothesis is that the means are not equal. When there is a significant improvement introduced by ED8-CF, this is denoted by setting the respective value in bold fonts; when on the other hand ED8-CF introduced a significant diminution to a certain metric, the respective value is underlined. As one can observe from Table III(a), there are no values in bold, and only one value is underlined (Aggreko's RF). This thus shows that ED8-CF performs as well as its predecessor, ED8-HH.

The above result should not surprise us. Even if an algorithm that is applied to EDDIE 8 is able to offer improvements to ED8's performance metrics, it would be extremely hard to consistently do this. This is because of the huge search space that ED8 is dealing with. Algorithms applied to ED8 might be able to offer improvements in certain runs, but on average, it is not likely to offer significantly improved average results.[9] Thus, one should focus on investigating the performance of the new algorithms (i.e., the Choice Function in this paper) in terms of the best results. This is done next.

B. Best results

The best results are presented in Table III(b). The formatting of the table follows the same philosophy as above, i.e., an improvement is denoted by bold fonts, and a diminution is denoted by underlined fonts.[10]

The first observation one can make is that we are dealing with a completely different picture than the one we discussed above in the average results. There are now quite a few improvements and the number of these improvements, caused by ED8-CF, is higher than the number of diminutions. This allows us to argue that *the Choice Function has been very beneficial to the hyper-heuristics framework, and to the ED-DIE algorithm as a whole*. We can observe that the Choice Function has offered 15 improvements to the HH framework. These improvements are: Aggreko (RF), Amlin (RC), Athens (RMC and RF), BAT (RC and RMC), Centrica (RF), Easyjet (Fitness, RC, and RMC), MDAX (Fitness, RC, and RF), Next (RF), and Xstrata (RF). It should also be noted that *some of these improvements are quite significant*, e.g., MDAX's Fitness increase from 0.1691 to 0.2798 (11% increase). In addition, it is also important to note that the number of diminutions caused by ED8-CF is low (only 4); this strengthens the argument that the Choice Function has improved the hyper-heuristics framework, because it not only has improved the values of the performance metrics, but it has also kept any negative effects to a low range.

[7]Nevertheless, we do not argue that these values are the optimal ones. We have left the investigation of searching for the optimal values of these parameters as a future work. More details about this can be found at the Conclusion section of this paper.

[8]Since fitness and RC are maximization problems, the 'best' result between two values is the maximum value. On the other hand, RMC and RF are minimization problems, so the 'best' result between two values is the minimum value.

[9]The same behaviour in terms of average results was also observed in [4].

[10]Since we are now dealing only with a single value (best), instead of distributions, we cannot run a formal statistical test to determine the best algorithm. We therefore consider an improvement to be significant when the value of the metric has improved by at least 1%.

TABLE III
AVERAGE [III(A)] AND BEST [III(B)] RESULTS FOR ED8-HH VS ED8-CF.

(a) Average Results

Dataset	Heuristic	Fitness	RC	RMC	RF
Aggreko	ED8-HH	0.252	0.5941	0.3392	0.2523
	ED8-CF	0.2508	0.5949	0.3023	0.2702
Amlin	ED8-HH	0.1756	0.5423	0.3663	0.3869
	ED8-CF	0.1569	0.5273	0.3835	0.4126
Athens	ED8-HH	0.1176	0.5049	0.4253	0.476
	ED8-CF	0.1303	0.5127	0.436	0.4576
BAT	ED8-HH	0.2192	0.5431	0.4606	0.2173
	ED8-CF	0.2348	0.5557	0.4272	0.2193
Carnival	ED8-HH	0.2085	0.5788	0.278	0.3811
	ED8-CF	0.193	0.5645	0.3201	0.3894
Centrica	ED8-HH	0.1888	0.4869	0.5353	0.1661
	ED8-CF	0.2048	0.4995	0.5186	0.1597
Easyjet	ED8-HH	0.0862	0.4137	0.7609	0.2972
	ED8-CF	0.1098	0.4335	0.7236	0.2796
MDAX	ED8-HH	0.0701	0.4759	0.3117	0.613
	ED8-CF	0.0784	0.4793	0.3073	0.5938
Next	ED8-HH	0.1258	0.4772	0.5276	0.3591
	ED8-CF	0.1454	0.4892	0.5037	0.3553
Xstrata	ED8-HH	0.2474	0.5903	0.2868	0.2602
	ED8-CF	0.2462	0.5841	0.3181	0.2416

(b) Best Results

Dataset	Heuristic	Fitness	RC	RMC	RF
Aggreko	ED8-HH	0.342	0.7133	0.0	0.0845
	ED8-CF	0.342	0.7133	0.0	**0.0**
Amlin	ED8-HH	0.2707	0.6467	0.0	0.0
	ED8-CF	0.2798	**0.66**	0.0	0.3287
Athens	ED8-HH	0.2362	0.61	0.0124	0.3764
	ED8-CF	0.2222	0.6067	**0.0**	**0.3167**
BAT	ED8-HH	0.3598	0.7233	0.0628	0.0658
	ED8-CF	0.369	**0.7433**	**0.0**	0.0833
Carnival	ED8-HH	0.2652	0.6467	0.0	0.2598
	ED8-CF	0.2745	0.6533	0.0	0.2685
Centrica	ED8-HH	0.4357	0.8167	0.0	0.0688
	ED8-CF	0.4371	0.8167	0.0	**0.0**
Easyjet	ED8-HH	0.3045	0.6733	0.0985	0.0
	ED8-CF	**0.3194**	**0.69**	**0.0**	0.0968
MDAX	ED8-HH	0.1691	0.6233	0.0	0.5301
	ED8-CF	**0.2798**	**0.64**	0.0	**0.0**
Next	ED8-HH	0.294	0.66	0.0	0.2476
	ED8-CF	0.294	0.66	0.0	**0.2308**
Xstrata	ED8-HH	0.3835	0.76	0.0	0.056
	ED8-CF	0.3896	0.7667	0.0	**0.0**

Another observation that can be made is that the improvements in the best values took place for 9 out of the 10 datasets. This is also very important, because it demonstrates that the Choice Function has the ability to improve the metrics of the majority of the datasets that is given. It thus seems that *the Choice Function can successfully be applied to a wide variety of datasets*.[11]

VII. CONCLUSION

To conclude, from the above preliminary results we can argue that the introduction of the Choice Function has been advantageous. As we have seen, the Choice Function is competitive in terms of average results, since it performs as well as its predecessor. We should again mention that because of the huge search space that EDDIE 8 is dealing with, it can be very hard to constantly locate extremely good solutions. Thus, it might not be that easy for any new EDDIE version to be consistently better in terms of average results.

On the other hand, the Choice Function has shown that it can perform very well in terms of best results. It seems that, if given enough individual runs, it is able to locate an extremely good solution, at least once. *This is all that an investor who uses EDDIE wants and needs*. In real life an investor is not necessarily interested in robust average results. If an investor was using EDDIE to assist him with his investments, he would run the algorithm many times and then select the best tree that was produced. Thus, we are very interested in any improvements can be achieved in terms of best results, because they have practical advantages in the real world. The Choice Function seems to be able to offer this, allowing us to *consider it as a beneficial extension to the EDDIE 8 series*. In addition to this, we should re-iterate that the applicability of the Choice Function to a wide range of different datasets is considered as an important advantage of the algorithm.

To conclude, the Choice Function seems to have maximized the advantages of the hyper-heuristics framework used in this paper. It is worth investigating whether this could generalize to other hyper-heuristics frameworks. We thus plan to further investigate this in future work. Plans about other types of future work are presented next.

There are several paths that can be followed as a future work. First of all, we could look into the possibility of dynamically updating the Choice Function values of α, β and δ. In the current framework these values are fixed. [14] has already done some work in this direction, by extending the work of [13], and introducing a learning scheme that dynamically updates the values of α, β and δ. This is certainly something that deserves further investigation. Finally, applying the Choice Function algorithm to more datasets could help with the generalisation of the results of this paper.

[11] Of course, this requires further investigation using a wider range of datasets. Nevertheless, the fact that there were improvements in 9 out of the 10 datasets tested is very encouraging.

REFERENCES

[1] E. Tsang and S. Martinez-Jaramillo, "Computational finance," *IEEE Computational Intelligence Society Newsletter*, pp. 3–8, 2004.

[2] S.-H. Chen, *Genetic Algorithms and Genetic Programming in Computational Finance*. Springer-Verlag New York, LLC, 2002.

[3] J. Binner, G. Kendall, and S.-H. Chen, Eds., *Applications of Artificial Intelligence in Finance and Economics*, ser. Advances in Econometrics. Elsevier, 2004, vol. 19.

[4] 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, accepted for Publication.

[5] M. Kampouridis and E. Tsang, "Using hyperheuristics under a GP framework for financial forecasting," in *Proc. Fifth International Conference on Learning and Intelligent Optimization (LION5)*, ser. Lecture Notes in Computer Science, C. A. Coello Coello, Ed., vol. 6683. Springer, Heidelberg, 2011, pp. 16–30.

[6] ——, "EDDIE for investment opportunities forecasting: Extending the search space of the GP," in *Proceedings of the IEEE World Congress on Computational Intelligence*, Barcelona, Spain, 2010, pp. 2019–2026.

[7] E. Tsang, J. Li, S. Markose, H. Er, A. Salhi, and G. Iori, "EDDIE in financial decision making," *Journal of Management and Economics*, vol. 4(4), 2000.

[8] E. Tsang, S. Markose, and H. Er, "Chance discovery in stock index option and future arbitrage," *New Mathematics and Natural Computation, World Scientific*, vol. 1(3), pp. 435–447, 2005.

[9] E. Özcan, B. Bilgin, and E. E. Korkmaz, "A comprehensive analysis of hyper-heuristics," *Intelligent Data Analysis*, vol. 12, no. 1, pp. 3–23, 2008.

[10] E. Hart, P. Ross, and J. Nelson, "Solving a real-world problem using an evolving heuristically driven schedule builder," *Evol. Comput.*, vol. 6, no. 1, pp. 61–80, 1998.

[11] P. Cowling and K. Chakhlevitch, "Hyperheuristics for managing a large collection of low level heuristics to schedule personnel," vol. 2, dec. 2003, pp. 1214 – 1221 Vol.2.

[12] E. Burke, B. MacCloumn, A. Meisels, S. Petrovic, and R. Qu, "A graph-based hyper heuristic for timetabling problems," *European Journal of Operational Research*, vol. 176, pp. 177–192, 2006.

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

[14] P. Rattadilok, "An investigation and extension of a hyper-heuristic framework," *Informatica*, vol. 34, pp. 523–534, 2010.

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

[16] R. Poli, W. Langdon, and N. McPhee, *A Field Guide to Genetic Programming*. Lulu.com, 2008.

[17] R. Edwards and J. Magee, "Technical analysis of stock trends," *New York Institute of Finance*, 1992.

[18] S. Martinez-Jaramillo, "Artificial financial markets: An agent-based approach to reproduce stylized facts and to study the red queen effect," Ph.D. dissertation, CFFEA, University of Essex, 2007.

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

[20] M. Austin, G. Bates, M. Dempster, V. Leemans, and S. Williams, "Adaptive systems for foreign exchange trading." *Quantitative Finance*, vol. 4(4), pp. 37–45, 2004.

[21] J. Backus, "The syntax and semantics of the proposed international algebraic language of Zurich," in *International Conference on Information Processing*. UNESCO, 1959, pp. 125–132.

[22] J. Li, "FGP: A genetic programming-ased financial forecasting tool," Ph.D. dissertation, Department of Computer Science, University of Essex, 2001.

[23] M. Kampouridis and E. Tsang, "Investment opportunities forecasting: Extending the grammar of a gp-based tool," *International Journal of Computational Intelligence Systems*, vol. 5, no. 3, pp. 530–541, 2012.

[24] S. Baluja, "Population-based incremental learning: a method for integrating genetic search based function optimisation and competitive learning," 1994, technical Report, Carnegie Mellon University.

[25] P. Larranaga and J. Lozano, *Estimation of Distribution Algorithms: A New Tool for Evolutionary Computation*. Norwell, MA: Kluwer, 2001.

[26] Q. Zhang, J. Sun, and E. Tsang, "Evolutionary algorithm with guided mutation for the maximum clique problem," *IEEE Transactions on Evolutionary Computation*, vol. 9(2), pp. 192–200, 2005.

[27] E. Tsang and J. Li, *EDDIE for financial forecasting*, ser. Genetic Algorithms and Genetic Programming in Computational Finance. Springer-Verlag New York, LLC, 2002, ch. 7, pp. 161–174.

[28] J. Li and E. Tsang, "Investment decision making using FGP: a case study," in *Proceedings of Congress on Evolutionary Computation*, 1999.