Noname manuscript No.
(will be inserted by the editor)

On the investigation of hyper-heuristics on a financial forecasting problem

Michael Kampouridis · Abdullah Alsheddy · Edward Tsang

Received: date / Accepted: date

Abstract Financial forecasting is a really important area in computational finance, with numerous works in the literature. This importance can be reflected in the literature by the continuous development of new algorithms. Hyper-heuristics have been successfully used in the past for a number of search and optimization problems, and have shown very promising results. To the best of our knowledge, they have not been used for financial forecasting. In this paper we present pioneer work, where we use different hyper-heuristics frameworks to investigate whether we can improve the performance of a financial forecasting tool called EDDIE 8. EDDIE 8 allows the GP (Genetic Programming) to search in the search space of indicators for solutions, instead of using pre-specified ones; as a result, its search area has dramatically increased and sometimes solutions can be missed due to ineffective search. We apply 14 different low-level heuristics to EDDIE 8, to 30 different datasets, and examine their effect to the algorithm's performance. We then select the most prominent heuristics and combine them into three different hyper-heuristics frameworks. Results show that all three frameworks are competitive, and are able to show significantly improved results, especially in the case of best results. Lastly, analysis on the weights of the heuristics shows that there can be a constant swinging among some of the low-level heuristics, which denotes that the hyper-

M. Kampouridis
School of Computer Science and Electronic Engineering, University of Essex, UK
Tel.: +44 (0) 1206 87 2678
E-mail: mkampo@essex.ac.uk

A. Alsheddy
College of Computer and Information Sciences (CCIS), Imam Muhammad Bin Saud Islamic University, Saudi Arabia

E. Tsang
Centre for Computational Finance and Economic Agents, School of Computer Science and Electronic Engineering, University of Essex, UK

heuristics frameworks are able to 'know' the appropriate time to switch from one heuristic to the other, based on their effectiveness.

Keywords Hyper-heuristics · Genetic Programming · Financial Forecasting

1 Introduction

Financial forecasting is an important area in computational finance [34]. There are numerous works that attempt to forecast the future price movements of a stock; several examples can be found in [10,7]. A number of different methods have been used for forecasting. Such examples are, for instance, Support Vector Machines [38,26,9,14,16], Learning Classifier Systems [27,28], Bayesian Kernel Models[31], Fuzzy Logic [17] and Neural Networks [6,37,29]. Genetic Programming [20,25] (GP) is an evolutionary technique that has widely been used for financial forecasting. Some recent examples are [30,6,1,12,36,35], where GP was used for time series forecasting.

Recently, we presented EDDIE 8 (ED8) [18], which was an extension of the financial forecasting tool EDDIE (Evolutionary Dynamic Data Investment Evaluator) [32,33]. EDDIE is a machine learning tool that uses Genetic Programming to make its predictions. The novelty of ED8 was in its extended grammar, which allowed the GP to search in the space of indicators to form its Genetic Decision Trees. In this way, ED8 was not constrained in using pre-specified indicators, but it was left up to the GP to choose the optimal ones. We then proceeded to compare ED8 with its predecessor, namely EDDIE 7 (ED7), which used indicators that were pre-specified by the user. Results showed that thanks to the new grammar, ED8 could find new and improved solutions. However, those results also suggested that ED8's performance could have been compromised by the enlarged search space. With the old grammar (ED7), which was also discussed in [18], EDDIE used 6 indicators from technical analysis with two pre-specified period lengths. For instance, if one of the indicators was Moving Average, then the two period lengths used would be 12 and 50 days. On the contrary, ED8 could use any period within a given parameterized range, which for our experiments was set to 2-65 days. Thus, the GP could come up with any indicator within that range, and not just with 12 and 50 days. As we can see, the search space of ED8 was much bigger than the one of its predecessor. To make this clearer, let us give an example: if a given GP tree can have a maximum of k indicators, then the permutations of the available 12 indicators[1] under ED7 are 12^k; on the other hand, if ED8 is using the same 6 indicators with periods within the range of 2 to 65 days, then the permutations of the available 384 indicators[2] are 384^k. It is thus obvious that ED8's search space is significantly larger, which can therefore explain the difficulties of ED8 of consistently finding good solutions.

[1] We are using 6 different indicators, with 2 periods each, thus $6 * 2 = 12$.
[2] We are using 6 different indicators with 65-1=64 periods each, thus $64 * 6 = 384$.

Furthermore, hyper-heuristics is a well-known method that has been used in a variety of search and optimization problems [24], such as transportation [15], scheduling [11], and timetabling [8], and has returned very promising results. To the best of our knowledge, hyper-heuristics have not applied before to a financial forecasting problem. The only exception to this is another previous work of ours [19], where we introduced a simple hyper-heuristics framework which was applied to ED8. Nevertheless, our approach in that work was simple: we selected two heuristics and two mutators and formed a hyper-heuristics framework. However, a drawback of that approach was that it did not offer any justification on the selection of the low-level heuristics that consisted the framework. In addition, results did not show any improvement in the average performance of the algorithm, but only an improvement on the minimum value of a single performance metric.

In this paper, we follow a more thorough investigation of the effect of the heuristics and hyper-heuristics. We start by applying 14 different heuristics to ED8, one at a time, to 30 different datasets. After this, we select the best performing heuristics and form hyper-heuristics frameworks. This paper thus constitutes the first rigorous work on the application of hyper-heuristics to a financial forecasting problem. Our aim is to demonstrate that the introduction of hyper-heuristics to this type of problems is advantageous. This is very important and significant, because of the significance of the financial forecasting field itself, which requires the continuous development of new and improved algorithms. In order to demonstrate the above, we apply hyper-heuristics to ED8, which has been established to be a useful financial forecasting tool [18, 32,33].

The rest of this paper is organized as follows: Section 2 presents the ED8 algorithm, Sect. 3 presents the hyper-heuristics framework, along with its low-level heuristics, Sect. 4 presents the experimental setup, Sect. 5 presents and discusses the results, and finally, Sect. 6 concludes this paper and also discusses future work.

2 Presentation of EDDIE 8

EDDIE is a forecasting tool, 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: daily closing price of a stock, a number of attributes and signals. Stocks' daily closing prices can be obtained online in websites such as `http : //finance.yahoo.com` and also from financial statistics databases like *Datastream*. The attributes are indicators commonly used in technical analysis [13]; which indicators to use depends on the user and his belief of their relevance to the prediction. The technical indicators that we use in this work are: Moving Average (MA), Trade Break Out (TBR), Filter (FLR), Volatility (Vol), Momentum (Mom), and Momentum Moving Average (MomMA).[3]

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 $r\%$ [32]. 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 we feed the data to the system, EDDIE creates and evolves a population of GDTs. Figure 1 presents the Backus Normal Form (BNF) [4] (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 see from the grammar in Fig. 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. As a result, ED8 can return decision trees with indicators like 15 days Moving Average, 17 days Volatility, etc. The period is not an issue and it is up to ED8, and as a consequence up to the GP and the evolutionary process, to decide which lengths are more valuable for the prediction. A sample GDT is presented in Fig. 2. As we can observe, the periods 12 and 50 are now in a leaf node, and thus are subject to genetic operators, such as crossover and mutation.

[3] We use these indicators because they have been proved to be quite useful in developing GDTs in previous works like [23], [2] and [3]. Of course, there is no reason why not 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.

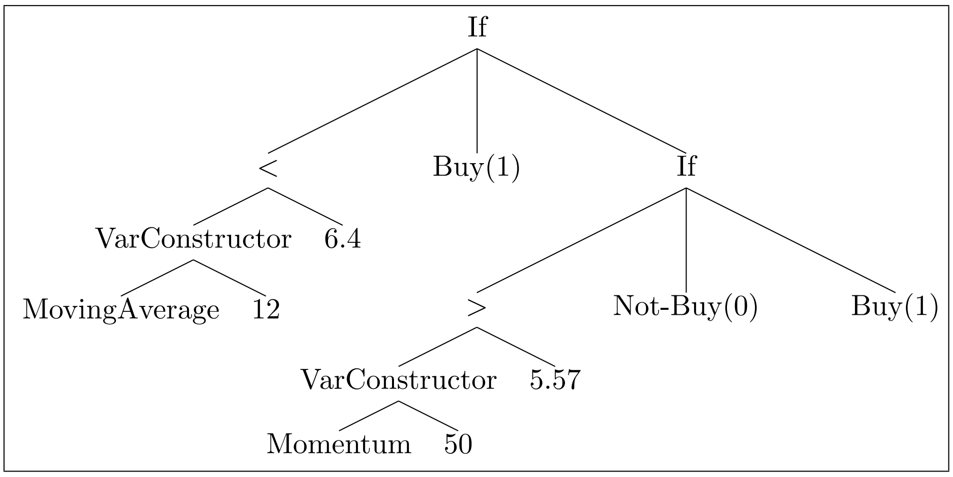

Fig. 2 Sample GDT generated by ED8.

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 our experiments, we choose to include strategies that mainly focus on 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 [33, 22]. The constraint is denoted by R, which consists of two elements represented by a percentage, given by

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

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[4] should fall into this range. When this happens, then w_1 remains as it is (i.e. 0.6 in our experiments). Otherwise, w_1 takes the value of zero.

This concludes this short presentation of ED8. In the next section we briefly present the heuristics used in our framework, and then present the hyper-heuristics framework itself.

3 Hyper-heuristics framework

3.1 Heuristics

In EDDIE 8, each GP individual represents a possible GDT whose basic component is the variable constructor. As mentioned earlier, a variable constructor comprises an *indicator* and a *period*. Therefore, the objective is to design heuristics that improve a given GDT by exploring the space of both indicators and periods.

The proposed heuristics can be divided into two groups, namely indicator based-heuristics and period-based heuristics. An indicator-based heuristic searches the space of indicators by making a small change to the indicators presented in the considered GDT; whereas a period-based heuristic focuses on the space of the periods in the GDT.

Each proposed heuristic takes 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

[4] As we have 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.

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, we devise the following period-based heuristics:

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. In this paper we experiment with 8 values of k: 1 ($PMut1$), 3 ($PMut3$), 5 ($PMut5$), 7 ($PMut7$), 10 ($PMut10$), 11 ($PMut11$), 13 ($PMut13$), and 15 ($PMut15$). These values were arbitrarily selected. It is not the scope of this paper to look for the optimal heuristic, but only to demonstrate the effectiveness of different heuristics combined under hyper-heuristics frameworks.
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.

The heuristics that focus on indicators are as follows:

1. Shuffle Mutation ($ShufMut$). This mutator puts the variable constructors of the considered GDT in a sequence following a depth-first order. Then, it shuffles the sequence, and then updates the GDT accordingly.
2. Swap Mutation ($SwapMut$). This mutator is similar to $ShufMut$, however, a single swap between two random variable constructors is performed, instead of shuffling the complete sequence.
3. Indicator-based hill climbing (IHC). This is an iterative hill climbing procedure in which a solution is represented by a sequence of variable constructors, representing their order in the considered GDT. In IHC, the neighbourhood includes any sequence that can be obtained from the current solution by performing a single swap between any two variable constructors.
4. Indicator-based single-step hill climbing ($sIHC$). This is the single-step hill climbing version of IHC.

3.2 The Framework

In this simple framework, all low level heuristics are used simultaneously. Inspired by the Population Based Incremental Learning algorithm [5] and the alike Estimation of Distribution Algorithms [21,39], all low level heuristics are initially given a weight w of being selected, where $w = \frac{1}{\#heuristics}$. Thus, if a framework consists of 4 heuristics, then the initial weight of each heuristic is $\frac{1}{4}$. 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, we slightly penalize the selected heuristic by decreasing its weight by $r/5$. Lastly, there is a punishment of r in the case of decrease in the performance.

At the moment, r is static throughout the GP run. We have left it as a future work to investigate different weight update setups, such as linear or exponential.[5] Let us now move to the next section, which presents the experimental setup.

4 Experimental Setup

The data we feed to ED8 consist of daily closing prices. These closing prices come from 23 arbitrary stocks from FTSE 100, and from 7 international indices. The 23 FTSE 100 stocks are: Aggreko, Amec, Amlin, AngloAmerican, Barclays, British Airways (BA), British Petroleum (BP), Cadbury, Carnival, Centrica, Easyjet, First, Hammerson, Imperial Tobacco, Marks and Spencer, Next, Royal Bank of Scotland (RBS), Schroders, Sky, Tesco, Unilever, and Xstrata. The 7 indices are: Athens Stock Exchange (Greece), Dow Jones Industrial Average (USA), Hang Seng Index (Hong Kong), MDAX (Germany), NASDAQ (USA), NIKEI (Japan), and NYSE (USA). The training period is 1000 days and the testing period 300.

The reason for using that many datasets (30 in total), is because it is possible that the heuristics that we have selected in this paper do not offer

[5] The same also applies to the fact that we have set the reward in the second scenario above to $r/5$ and not some other fraction of r. The reader should keep in mind that it is outside the scope of this work to look for the optimal parameter values. Our focus is on investigating whereas hyper-heuristics can improve the performance of ED8, under the same set of parameter values. This principle applies to any other parameters that can be found in this work.

any improvements to the performance of some of the stocks. Thus, it would be futile to apply a hyper-heuristics framework to these datasets. Therefore, while we start our experiments with 30 datasets, it is very likely that hyper-heuristics will be applied to only a fraction of them.

Table 1 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
Period (ED8)	[2,65]

The GP parameters are presented in Table 1. For statistical purposes, we run the GP for 50 times. Thus, the process that is followed is that 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.

In addition, we should emphasize that we require that the datasets have a satisfactory number of actual positive signals. By this we mean that we are neither interested in datasets with a very low number of signals, neither with an extremely high one. Such cases would be categorized as chance discovery, where people are interested in predicting rare events, such as a stock market crash. Clearly this is not the case in our current work, where we use EDDIE for investment opportunities forecasting. We are thus interested in datasets that have opportunities around 50-70% (i.e. 50-70% of actual positive signals). Therefore, we need to calibrate the values of r and n (see Sect. 2) accordingly, so that we can obtain the above percentage from our data. For our experiments, the value of n is set to 20 days. The value of r varies, depending on the dataset. This is because one dataset might reach a percentage of 50-70% with $r = 4\%$, whereas another one might need a higher or lower r value. Accordingly, we need to calibrate the value of the R constraint, so that EDDIE produces GDTs that forecast positive signals in a range which includes the percentage of the *actual* positive signals of the dataset we are experimenting with. R thus takes values in the range of $[-5\%, +5\%]$ of the number of positive signals that the dataset has. For instance, if under $r = 4\%$ and $n = 20$ days, a dataset has 60% of actual positive signals, then R would be set to [55,65].

Moreover, Table 2 presents the parameters of the hyper-heuristics framework. The probability of applying hyper-heuristics is set for this work at 35%. Thus, 35% of the GDTs' periods can be updated through hyper-heuristics at each generation. We did not want to set a higher probability, because this could increase the computational times. As mentioned earlier, the initial weight of each heuristic is set to $\frac{1}{\#heuristics}$, where $\#heuristics$ denotes the number of heuristics that exist in a hyper-heuristics framework. Lastly, the reward is set to be equal to 5% of the initial weight. So in the above example the reward would be $\frac{1}{4} \times 0.05 = 0.0125$. At the moment, the reward is static throughout the entire GP run. We leave it as a future work to investigate the effects of different reward update setups.

Lastly, it should be mentioned that before deciding which heuristics are going to be combined in the hyper-heuristics framework, we are interested in examining the performance of each one of them individually. This will then allow us to select the most prominent ones and combine them in the framework. Therefore, the next section starts by reporting on the performance of the individual heuristics (Section 5.1). Then, in Section 5.2 we present the results from the hyper-heuristics framework.

Table 2 hyper-heuristics Parameters.

hyper-heuristics Parameters	
hyper-heuristics probability	0.35
Initial Weight	$\frac{1}{\#heuristics}$
Reward/Punishment	$\frac{1}{\#heuristics} \times 0.05$

5 Results

5.1 Individual heuristics results

In this section, we are interested in identifying the heuristics that have offered the most improvements to the fitness and performance metrics of the 30 datasets. In addition, we are interested in also identifying the most "promising" datasets, i.e., the datasets that benefited the most by the heuristics that were applied to them.

Table 3 presents the total number of improvements that were introduced to each dataset's Fitness, RC, RMC and RF. In order to calculate an improvement, we run a two tailed t-test at 10% significant level[6], and compare the distribution of the original ED8 version with the distribution of each one of the 14 heuristics. The null hypothesis (H_0) is that the two distributions that are compared have equal means and equal but unknown variances, whereas the

[6] At this point, we did not want to impose a stricter significance level (5%), because this could result in excluding many datasets and heuristics.

alternative hypothesis (H_1) is that the means are not equal.[7] In this way, we can have a clear idea of how many times the introduction of the heuristics has improved ED8, and in which datasets. The calculation of the improvements results to a 'league' table. This means that whenever there is an improvement in any performance metric (Fitness, RC, RMC, RF), in any of the 14 heuristics, the respective dataset receives 1 point. Let us give an example. If $PMut1$ has significantly improved all 4 performance metrics of a dataset X, then this would count as 4 points. We would then continue with the remaining 13 heuristics and calculate how many points were collected under the same dataset. In the end, we would sum up all points for the X dataset and present them in Table 3. Results are sorted and presented in descending order. The first column presents the 10 best performing datasets,[8] the second column the next 10 best, and the third column the last 10 datasets. The first dataset in the list is Carnival, which as we can observe leads the table with a total number of 32 improvements.

Table 3 Datasets league table. Results have been sorted by the datasets' points in descending order.

Dataset	Points	Dataset	Points	Dataset	Points
Carnival	32	Hammerson	6	Amec	2
Xstrata	27	Schroders	4	NIKEI	2
Athens	22	DJIA	4	Tesco	1
Centrica	12	BP	3	NASDAQ	1
Next	12	Marks&Spencer	3	First	1
MDAX	11	HSI	3	BA	1
Amlin	10	Barclays	3	AngloAmerican	0
BAT	10	Imp.Tobacco	3	RBS	0
Aggreko	8	NYSE	3	Sky	0
Easyjet	8	Unilever	2	Cadbury	0

As we can observe from Table 3, there are quite a few stocks at the end of the table that have very few or no improvements at all. It would thus be futile to apply hyper-heuristics to these datasets; since the low-level heuristics did not manage to improve the datasets' performance metrics (or offered very few improvements), we believe that hyper-heuristics would also not offer any improvements, since after all they use the exact same heuristics. We thus choose to apply hyper-heuristics only to the first 10 datasets in the list. These datasets are: Carnival, Xstrata, Athens, Centrica, Next, MDAX, Amlin, BAT, Aggreko, and Easyjet. To decide which heuristics should be included in the hyper-heuristics framework, we consult Table 4.

[7] The null hypothesis remains the same throughout all t-tests of this paper. Thus, whenever we mention that we run a t-test between ED8 and a heuristic/hyper-heuristic, we run it under the null hypothesis that the means of these two algorithms are the same.

[8] A 'well-performing' dataset is considered to be the one that has received many points in the league table-thus has received many improvements in its performance metrics.

Table 4 Heuristics league table. Results have been sorted by the heuristics' points in descending order. This league table takes into account heuristics' results under all 30 datasets tested.

Heuristic	Points
$PMut11$	18
$PMut15$	18
$PMut1$	17
$sPHC$	17
$PMut13$	16
$PMut3$	15
PHC	14
$sIHC$	14
IHC	14
$ShufMut$	14
$SwapMut$	14
$PMut7$	11
$PMut10$	9
$PMut5$	7

Table 4 presents the number of improvements each heuristic has introduced to all 30 datasets. The process is similar to Table 3: we 'iterate' through each heuristic and count the number of significant improvements that it has introduced to each dataset. Each improvement counts for 1 point. In the end, we sum up the number of points each heuristic has collected and presented in Table 4. Results are again sorted in descending order. $PMut11$ and $PMut15$ are the two leading heuristics with 18 improvements. However, as we can observe, the majority of the heuristics have performed similarly well. For this reason, we choose to use a hyper-heuristics framework that will include all 14 heuristics. We call this framework HH-1. HH-1's advantage is that it is a very general framework, because it uses all 14 heuristics.

Nevertheless, while the above is an advantage, it also poses at the same time a disadvantage: HH-1 does not 'give credit' to those heuristics that have done significantly well under the 10 datasets that we use for our hyper-heuristics experiments (best 10 datasets from Table 3).[9] Therefore, we believe that it would be beneficial to create hyper-heuristics frameworks that acknowledge the fact that certain heuristics can be highly beneficial to specific datasets. This kind of frameworks are more 'focused', and have thus the potential to result to even higher performance. For this reason, we re-construct the league table, with the difference that now we only take into account the improvements that have taken place for the 10-best performing datasets. In addition, since we are now examining the heuristics' performance to the best performing datasets only, we stricken the significance level of the t-test from 10% to 5%. As we

[9] As we have already mentioned, we are not going to be using the 20 datasets from the last two columns of Table 3 for our hyper-heuristics experiments. However, when constructing HH-1, we took into account even the improvements that took place to these 20 datasets. As explained, the advantage of that approach is that it creates a very general framework, without the need of having to disregard certain heuristics.

can observe from Table 5, results are slightly different. To begin with, not all heuristics have offered many improvements. For instance, $PMut7$ and $PMut5$ have only offered 2 and 1 improvements, significantly. This table leads us to construct two additional hyper-heuristics frameworks. One with the top-5 heuristics (PHC, $PMut15$, $PMut11$, $PMut13$, and $sPHC$), and one with the top-10 heuristics (PHC, $PMut15$, $PMut11$, $PMut13$, $sPHC$, $PMut1$, IHC, $ShufMut$, $PMut3$, and $sIHC$). We call these two new frameworks HH-2 and HH-3, respectively.

Table 5 Heuristics league table for the 10-best datasets. Results have been sorted by the heuristics' points in descending order. This league table takes into account heuristics' results under the 10-best performing datasets.

Heuristic	Points
PHC	12
$PMut15$	10
$PMut11$	8
$PMut13$	8
$sPHC$	8
$PMut1$	7
IHC	7
$ShufMut$	7
$PMut3$	5
$sIHC$	5
$PMut10$	4
$SwapMut$	3
$PMut7$	2
$PMut5$	1

5.2 Hyper-heuristics results

5.2.1 Hyper-heuristics framework 1

Tables 6 and 7 present the average and best[10] results, over 50 runs, from the comparison of the original version of ED8 with the first hyper-heuristics framework, denoted by HH-1. As already mentioned, this framework consists of all 14 heuristics. In order to see if HH-1 has offered a significant improvement to the mean values of the performance metrics, we again use a two-tailed t-test, between the original version of EDDIE 8, and HH-1. When HH-1 significantly

[10] 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.

improves a metric at 5% significance level,[11] we set the respective value of HH-1 in bold font. When, on the other hand, there is a significant diminution in a metric's value, we denote it by underlying the HH-1's value.[12]

Dataset	Heuristic	Fitness	RC	RMC	RF
Aggreko	Original	0.2424	0.5919	0.3132	0.2716
	HH-1	0.2351	0.5813	0.3586	0.2595
Amlin	Original	0.1469	0.5207	0.4155	0.4132
	HH-1	0.1591	0.53	0.3792	0.4031
Athens	Original	0.1469	0.5475	0.2481	0.5227
	HH-1	<u>0.1224</u>	<u>0.5084</u>	<u>0.4289</u>	**0.4658**
BAT	Original	0.2122	0.5458	0.4318	0.2403
	HH-1	0.2304	0.5531	0.4292	0.2276
Carnival	Original	0.1784	0.553	0.3658	0.3895
	HH-1	**0.1975**	**0.5701**	**0.3422**	0.3783
Centrica	Original	0.1694	0.46	0.5824	0.1612
	HH-1	0.185	0.4826	0.5442	0.1672
Easyjet	Original	0.0685	0.4051	0.747	0.3328
	HH-1	0.1054	0.4329	0.7136	**0.2965**
MDAX	Original	0.0767	0.4852	0.3235	0.607
	HH-1	0.0802	0.4883	0.3172	0.6026
Next	Original	0.1316	0.4825	0.5348	0.348
	HH-1	0.1213	0.4705	0.546	<u>0.3629</u>
Xstrata	Original	0.2061	0.5362	0.3942	0.2541
	HH-1	0.2117	0.5411	0.3693	0.2699

Table 6 Average Results for HH-1.

As we can observe, HH-1 is quite competitive. In terms of average values, HH-1 has improved 4 different datasets (Amlin, BAT, Carnival, Easyjet). However, what is more impressive is the improvements in the best values (Table 7). We can see that HH-1 has improved the best value of ED8 in *all 10 datasets* by at least 1%.[13] Moreover, in the majority of the datasets, these improvements happen in more than one metric. In total, HH-1 has offered 24 improvements to the metrics of the datasets, while it has caused only 7 diminutions. Lastly, it is worth mentioning that in some cases HH-1 has offered remarkable improvements in the best values: Athens' RF: improvement by about 27%, Centrica's Fitness, RC and RMC: improvements by about 15, 23, and 43%, respectively. Overall, we can argue that HH-1 is better than the original ED8 version.

[11] Because of the fact that we are running multiple comparisons between EDDIE 8 and a hyper-heuristics framework (i.e., we compare the two versions in terms of four different metrics: Fitness, RC, RMC, RF), we apply Bonferroni adjustment to ensure that the statistical error is indeed 5%. Since we compare EDDIE 8 under 10 datasets and in terms of 4 metrics, we have a total of 10×4 tests. Thus, the 'real' statistical error, which is equivalent to 5%, is $\frac{1-0.05}{4\times 10} = 0.02375$ or 2.375%.

[12] The above practice of using bold fonts and underlining the significantly worsened values also applies to the remaining sections of the paper.

[13] We cannot of course run a t-test for the comparison of the best values, since we are not dealing with distributions, but only with a single value (best). We consider a significant improvement/diminution when the difference of HH-1 with the original ED8 version is above 1%.

Dataset	Heuristic	Fitness	RC	RMC	RF
Aggreko	Original	0.3256	0.6933	0.0607	0.1373
	HH-1	**0.342**	**0.7133**	**0.0**	0.1856
Amlin	Original	0.2838	0.6633	0.0568	0.269
	HH-1	0.2676	0.6467	**0.0**	0.3226
Athens	Original	0.2198	0.6333	0.0794	0.4545
	HH-1	**0.2491**	0.6233	**0.0**	**0.1831**
BAT	Original	0.3303	0.6667	0.278	0.1083
	HH-1	**0.369**	**0.7433**	**0.0**	0.1053
Carnival	Original	0.2851	0.6667	0.1561	0.2536
	HH-1	0.2507	0.63	**0.0**	0.2632
Centrica	Original	0.2802	0.5833	0.4327	0.0571
	HH-1	**0.435**	**0.8167**	**0.0**	0.0735
Easyjet	Original	0.2788	0.6433	0.1773	0.129
	HH-1	**0.309**	**0.6767**	**0.0**	0.129
MDAX	Original	0.1849	0.6067	0.0973	0.5143
	HH-1	0.189	**0.6467**	**0.0**	**0.4186**
Next	Original	0.2687	0.6333	0.2727	0.2672
	HH-1	**0.294**	**0.66**	**0.0**	**0.2171**
Xstrata	Original	0.3761	0.7433	0.115	0.0513
	HH-1	0.378	0.7533	**0.0**	**0.0125**

Table 7 Best Results for HH-1.

5.2.2 Hyper-heuristics framework 2

This section presents the results from the second hyper-heuristics framework, called HH-2. As we have already explained, this framework consists of the best-5 performing heuristics for the 10 datasets we have selected to experiment with. These heuristics are: PHC, $PMut15$, $PMut11$, $PMut13$, and $sPHC$. Tables 8 and 9 present the average and best results. Improved values are again in bold fonts, and worsened values are underlined.

We can observe from Table 8 that HH-2 has improved the average values in 5 datasets: Aggreko, Athens, BAT, Carnival, and Xstrata. Only one dataset (Athens Stock Exchange) experiences worsened results in some of its metrics (Fitness, RC, RMC).

Regarding the best values (Table 9), HH-2, like HH-1, shows improvements to all 10 datasets. The total number of improvements is 29, whereas the diminutions are only 8. We should again mention some remarkable improvements by HH-2 in the best values: Amlin's RF by 27%, BAT's RMC by 21%, Centrica's Fitness, RC, and RMC by 15, 23, and 43%, respectively, and Next's RMC by 27%. Overall, we can argue that HH-2 is also better than the original ED8 version.

5.2.3 Hyper-heuristics framework 3

This section presents the results from the third hyper-heuristics framework, called HH-3. This framework consists of the best-10 performing heuristics for the 10 datasets we have selected to experiment with. These heuristics are: PHC, $PMut15$, $PMut11$, $PMut13$, $sPHC$, $PMut1$, IHC, $ShufMut$,

Dataset	Heuristic	Fitness	RC	RMC	RF
Aggreko	Original	0.2424	0.5919	0.3132	0.2716
	HH-2	0.252	0.5941	0.3392	**0.2523**
Amlin	Original	0.1469	0.5207	0.4155	0.4132
	HH-2	0.1756	0.5423	0.3663	0.3869
Athens	Original	0.1469	0.5475	0.2481	0.5227
	HH-2	0.1176	0.5049	0.4253	**0.4760**
BAT	Original	0.2122	0.5458	0.4318	0.2403
	HH-2	0.2192	0.5431	0.4606	**0.2173**
Carnival	Original	0.1784	0.553	0.3658	0.3895
	HH-2	**0.2085**	**0.5788**	**0.278**	0.3811
Centrica	Original	0.1694	0.46	0.5824	0.1612
	HH-2	0.1888	0.4869	0.5353	0.1661
Easyjet	Original	0.0685	0.4051	0.747	0.3328
	HH-2	0.0862	0.4137	0.7609	0.2972
MDAX	Original	0.0767	0.4852	0.3235	0.607
	HH-2	0.0701	0.4759	0.3117	0.613
Next	Original	0.1316	0.4825	0.5348	0.348
	HH-2	0.1258	0.4772	0.5276	0.3591
Xstrata	Original	0.2061	0.5362	0.3942	0.2541
	HH-2	**0.2474**	**0.5903**	**0.2868**	0.2602

Table 8 Average Results for HH-2.

Dataset	Heuristic	Fitness	RC	RMC	RF
Aggreko	Original	0.3256	0.6933	0.0607	0.1373
	HH-2	**0.342**	**0.7133**	**0.0**	**0.0845**
Amlin	Original	0.2838	0.6633	0.0568	0.269
	HH-2	0.2707	0.6467	**0.0**	**0.0**
Athens	Original	0.2198	0.6333	0.0794	0.4545
	HH-2	**0.2362**	0.61	**0.0124**	**0.3764**
BAT	Original	0.3303	0.6667	0.278	0.1083
	HH-2	**0.3598**	**0.7233**	**0.0628**	**0.0658**
Carnival	Original	0.2851	0.6667	0.1561	0.2536
	HH-2	0.2652	0.6467	**0.0**	0.2598
Centrica	Original	0.2802	0.5833	0.4327	0.0571
	HH-2	**0.4357**	**0.8167**	**0.0**	0.0688
Easyjet	Original	0.2788	0.6433	0.1773	0.129
	HH-2	**0.3045**	**0.6733**	**0.0985**	**0.0**
MDAX	Original	0.1849	0.6067	0.0973	0.5143
	HH-2	0.1691	**0.6233**	**0.0**	0.5301
Next	Original	0.2687	0.6333	0.2727	0.2672
	HH-2	**0.294**	**0.66**	**0.0**	**0.2476**
Xstrata	Original	0.3761	0.7433	0.115	0.0513
	HH-2	0.3835	**0.76**	**0.0**	0.056

Table 9 Best Results for HH-2.

*PMut*3, and *sIHC*. Tables 10 and 11 present the average and best results. Improved values are again in bold fonts, and worsened values are underlined.

HH-3 has improved the average values of Amlin's RMC, Athens' RF, Carnival's RC and RMC, and Easyjet's Fitness, RC, and RF. This makes 4 datasets with improved metrics. On the other hand, we can only see that there are only 2 datasets with worsened average results (Athens and Next). The remaining datasets do not show any significant improvements/diminutions. Thus, it can be argued that HH-3 is somehow better than the original ED8 version.

Dataset	Heuristic	Fitness	RC	RMC	RF
Aggreko	Original	0.2424	0.5919	0.3132	0.2716
	HH-3	0.2264	0.5717	0.3618	0.2682
Amlin	Original	0.1469	0.5207	0.4155	0.4132
	HH-3	0.1692	0.5404	0.3322	0.4062
Athens	Original	0.1469	0.5475	0.2481	0.5227
	HH-3	0.1316	<u>0.5134</u>	<u>0.4015</u>	**0.4544**
BAT	Original	0.2122	0.5458	0.4318	0.2403
	HH-3	0.223	0.5476	0.4444	0.2193
Carnival	Original	0.1784	0.553	0.3658	0.3895
	HH-3	0.1893	0.5645	**0.3177**	0.392
Centrica	Original	0.1694	0.46	0.5824	0.1612
	HH-3	0.1646	0.4522	0.5821	0.1755
Easyjet	Original	0.0685	0.4051	0.747	0.3328
	HH-3	**0.1235**	**0.4441**	0.6985	**0.2735**
MDAX	Original	0.0767	0.4852	0.3235	0.607
	HH-3	0.0687	0.4808	0.3437	0.6154
Next	Original	0.1316	0.4825	0.5348	0.348
	HH-3	0.1252	0.4797	0.5192	<u>0.369</u>
Xstrata	Original	0.2061	0.5362	0.3942	0.2541
	HH-3	0.235	0.566	0.3375	0.2538

Table 10 Average Results for HH-3.

Let us now move to the best results presented in Table 11. As with the previous two frameworks, HH-3 has significantly improved the best values of the performance metrics *in all 10 datasets*. The total number of improvements is 32, and the total diminutions are only 2. This is very impressive, because it shows that in the majority of the times somebody would use HH-3, would benefit significantly. There are again, like in the previous frameworks, some datasets that have shown significant improvements: Athens' RF by 45%, BAT's RMC by 24%, Carnival's RMC by 15%, Centrica's Fitness, RC, and RMC by 15, 23, and 43%, respectively, Next's RMC by 27%, and Xstrata's RMC by 11%.

5.2.4 Discussion

As we have seen in the previous three subsections, all hyper-heuristics frameworks have improved the performance of ED8 in terms of the mean values. Table 12 presents a summary of the improvements that the frameworks have introduced, at 5% significance level. As we can observe, both HH-1 and HH-3 have improved ED8 in 3 out of the 10 datasets. In addition, HH-2 has improved ED8 in 5 datasets. Lastly, the total number of improvements (values with bold fonts in Tables 6, 8, 10) is consistently higher than the number of diminutions (underlined values in Tables 6, 8, 10), for all 3 datasets (HH-1: 5-4, HH-2: 9-2, HH-3: 5-3). The above thus allow us to argue that all three hyper-heuristics

Dataset	Heuristic	Fitness	RC	RMC	RF
Aggreko	Original	0.3256	0.6933	0.0607	0.1373
	HH-3	**0.3447**	**0.7167**	**0.0**	0.1364
Amlin	Original	0.2838	0.6633	0.0568	0.269
	HH-3	0.2794	0.66	**0.0**	<u>0.3019</u>
Athens	Original	0.2198	0.6333	0.0794	0.4545
	HH-3	**0.2452**	0.6333	**0.0**	**0.0**
BAT	Original	0.3303	0.6667	0.278	0.1083
	HH-3	**0.3639**	**0.7367**	**0.0359**	**0.0**
Carnival	Original	0.2851	0.6667	0.1561	0.2536
	HH-3	**0.3126**	**0.6933**	**0.0**	**0.2387**
Centrica	Original	0.2802	0.5833	0.4327	0.0571
	HH-3	**0.435**	**0.8167**	**0.0**	<u>0.0723</u>
Easyjet	Original	0.2788	0.6433	0.1773	0.129
	HH-3	**0.309**	**0.6767**	**0.0**	**0.0769**
MDAX	Original	0.1849	0.6067	0.0973	0.5143
	HH-3	0.1888	**0.6233**	**0.0**	0.5196
Next	Original	0.2687	0.6333	0.2727	0.2672
	HH-3	**0.294**	**0.66**	**0.0**	**0.2165**
Xstrata	Original	0.3761	0.7433	0.115	0.0513
	HH-3	**0.3918**	**0.77**	**0.0**	**0.0**

Table 11 Best Results for HH-3.

frameworks are competitive to ED8. In addition, HH-2 is doing better than the other two frameworks.

Framework	No of Improved Stocks (Mean Results)	Improvements	Diminutions
HH-1	3/10	5	4
HH-2	5/10	9	2
HH-3	3/10	5	3

Table 12 Summary Average Results.

Furthermore, we saw in the previous subsections that all three frameworks offered remarkable improvements to the best values of ED8, with some of them being even above 10 or 20%. Table 13 informs us that HH-1 offered 24 improvements (bold values in Table 7) and 7 diminutions (underlined values in Table 7), HH-2 offered 29 improvements (bold values in Table 9) and 8 diminutions (underlined values in Table 9), and HH-3 offered 32 improvements (bold values in Table 11) and only 2 diminutions (underlined values in Table 11). To better interpret these results, we should take into account that we were examining 4 performance measures (Fitness, RC, RMC, RF) for 10 datasets; thus, there were in total 40 cases that we were examining. Hence, HH-1 had improvements in 60% of these cases (24 out of 40), HH-2 72.5%, and HH-3 improved the 80% of these cases. These figures are very high and impressive, especially if we also consider the very low percentage of the diminutions (HH-1: 17.5%, HH-2: 20%, HH-3: 5%).

We should mention that we are especially interested in the best values, because if an investor was using ED8 to assist him with his investments, he

Framework	Improvements	Diminutions
HH-1	24	7
HH-2	29	8
HH-3	32	2

Table 13 Summary Best Results.

would run the algorithm many times and then select the best tree that was produced. Thus, having such improved results in terms of best has very practical advantages in the financial world.

The above thus allow us to claim that the introduction of hyper-heuristics frameworks to EDDIE has been very advantageous. All three frameworks have shown positive results. For instance, HH-1 is a more general framework that uses all 14 heuristics, while HH-2 and HH-3 are more 'focused'. Also, all 3 of them have offered similar number of improvements to both the mean and best values of the performance metrics, with HH-2 doing better in terms of mean results, and HH-3 doing better when it comes to the best values' results. We believe that these improvements in the average values of the performance metrics, and especially in the best values, perfectly demonstrate the effectiveness of the application of hyper-heuristics to EDDIE 8.

Furthermore, Table 14 presents the computational times of a single run for the Aggreko dataset, under ED8 and the three hyper-heuristics frameworks.[14] As we can observe, the original version of ED8 is faster than the other three algorithms. However, this is justifiable because of the number of extra heuristics all these three frameworks use. In addition, we believe that the computational cost observed in HH-1, HH-2 and HH-3 is worthy, because of the significantly improved results in the mean and best values of the performance metrics, as presented earlier.

Algorithm	Computational Time
Original ED8	2 mins & 47 secs
HH-1	4 mins & 35 secs
HH-2	6 mins & 55 secs
HH-3	4 mins & 15 secs

Table 14 Computational times of a single run for the Aggreko dataset.

Lastly, in order to understand the contribution of each heuristic to the effectiveness of the framework, the evolution over time of the weights of the different heuristics of a single run produced by HH-2 for the Athens Stock Exchange and the BAT datasets is plotted in Figures 3 and 4.

Figure 3 clearly shows that the effectiveness of each heuristic, in comparison with others, varies as the search progress. While all 5 heuristics started

[14] The computational times of the other 9 datasets we experimented with are similar to Aggreko's. For this reason, we chose to provide the times of one dataset only (Aggreko), as reference.

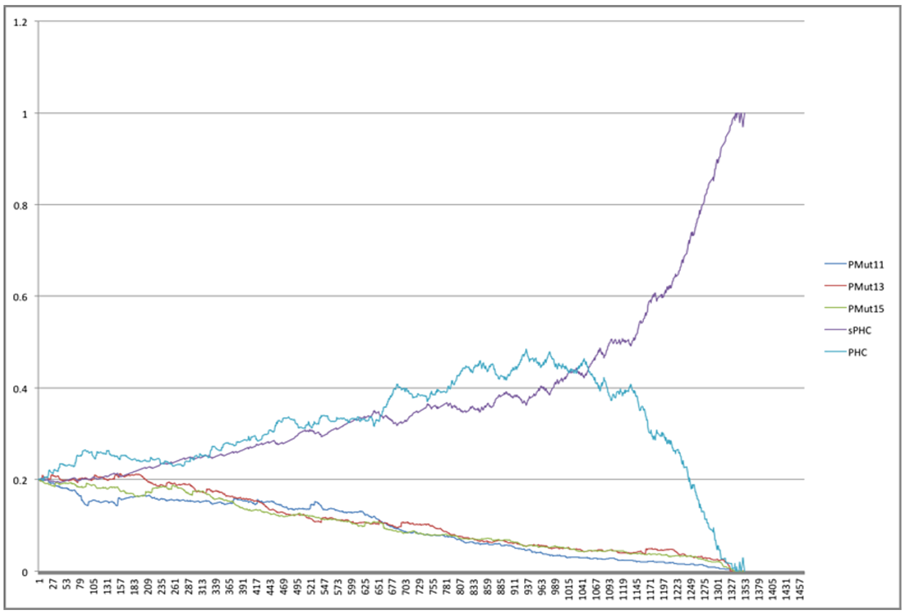

Fig. 3 Heuristics weights over time for HH-2. Single run of Athens Stock Exchange.

with the same weight, soon after that 3 of them start losing weight ($PMut11$, $PMut13$, $PMut15$), while the remaining 2 heuristics ($sPHC$ and PHC) see their weights increasing by time. Nevertheless, $PMut11$, $PMut13$, and $PMut15$ still continue to contribute to the search, even if this contribution reduces over time. It thus seems that even with the lower weights of these 3 heuristics, they are still able to offer improvements that boost the average performance of HH-2. Another important observation is that $sPHC$ and PHC seem to complement each other; when there is an increase in one, there is a decrease in the other. However, towards the end PHC seems to be taking over all other heuristics. Its weight keeps increasing and reaches around 100%, while the other 4 heuristics' weights end up around 0%.

We can make similar conclusions for Figure 4. However, the difference here is that there is no heuristic that is a 'clear winner' (like PHC for Athens Stock Exchange). No heuristic's weight reaches neither 0, nor 100%. It seems that the search space of BAT is very different from Athens Stock Exchange. For BAT, $sPHC$ and PHC complement each other throughout the whole process; while $sPHC$ seems to be winning at first, PHC reaches and goes above $sPHC$, around the middle of the process. Towards the end, $sPHC$ is momentarily able to outperform PHC once again. This is a very interesting observation, because it implies that the hyper-heuristics framework is able to "know" the appropriate time to switch from one heuristic to another. This swinging, which happens a few times, demonstrates the strength of combining individual heuristics into a hyper-heuristics framework. Furthermore, as we mentioned we can observe that the heuristics' weights evolve differently

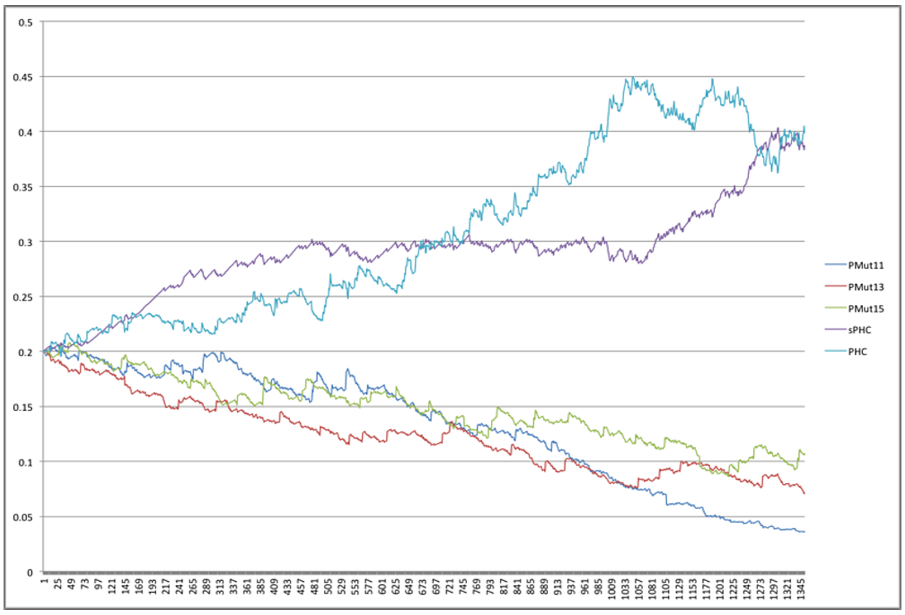

Fig. 4 Heuristics weights over time for HH-2. Single run of BAT.

between the two datasets, BAT and Athens Stock Exchange. This is another advantage of the hyper-heuristics approach, because it allows different datasets to benefit from different heuristics.

To summarize, both of these figures demonstrate the clear advantages of using a hyper-heuristics framework. In particular, they indicate the effectiveness of the weight updating scheme, with which the proposed framework has the ability of deciding which heuristic is more efficient at a given time and apply it to the trees of the GP population. Secondly, they also demonstrate the effectiveness of hyper-heuristics, with which different datasets can take advantage of different heuristics.

6 Conclusion

To conclude, this paper presented work on the application of hyper-heuristics to a financial forecasting problem. Hyper-heuristics is a well-known heuristic method which has been applied to many different problems. This paper presents the first, to the best of our knowledge, work on applying hyper-heuristics to financial forecasting (with the only exception being our previous work [19]). In that work, we introduced a simple hyper-heuristics framework and examined its utility after we applied it to EDDIE 8, a financial forecasting tool. The novelty of EDDIE 8 is that it allows the GP to search in the search space of indicators for solutions, instead of using pre-specified ones; however, a consequence of this is that its search area is quite large and sometimes solutions can be missed due to ineffective search. The use of hyper-heuristics in our previous work [19] showed promising results, because we saw a significant improvement in the best results of a single performance metric. Nevertheless, the hyper-heuristics framework used was quite simple, and also provided no justification for the selection of the low-level heuristics that consisted the framework.

In this paper, we followed a more thorough approach. First, 14 different heuristics were devised and individually examined. This allowed us to identify those heuristics that were the most promising in terms of performance (fitness). Then, the most successful heuristics were combined into three different hyper-heuristics frameworks. Furthermore, we applied these 14 heuristics to 30 different datasets. The high number of datasets allowed us to examine and afterwards focus only on the datasets that received the biggest number of improvements in their performance metrics (Fitness, RC, RMC, RF). Thus, after applying the heuristics to all 30 datasets, we selected 10 and tested the three hyper-heuristics frameworks on them.

Results showed that all frameworks were quite competitive. The first framework, HH-1, showed that it could improve the average performance of the metrics in 4 out of 10 datasets. The second and third framework offered improvements in the performance metrics of all 5 and 4 datasets, respectively. More importantly, all three frameworks improved the best results of the performance metrics *for all 10 datasets.* This is an extremely important result, because it indicates that an investor using the best tree of either of HH-1, HH-2 or HH-3, would be always guaranteed significantly improved results, which could lead to a significant increase in his profit.

Both of the above results demonstrate the effectiveness of hyper-heuristics to our financial forecasting problem. The original version of EDDIE 8 can perform well, but it can sometimes miss some ‘good’ solutions due to ineffective search [18]. Therefore, the introduction of the hyper-heuristics has allowed EDDIE 8 to come with improved average and best solutions.

In addition, further investigation on the weight updating scheme used by the framework revealed its effectiveness on controlling the weight of each heuristic. As we saw, this scheme enables the algorithm to know when it is more efficient to focus on the period of an indicator, and when to focus on the indicator itself, and switch accordingly. Lastly, the scheme also enables the algorithm to know which heuristics are more effective under certain datasets.

Future work will focus on testing hyper-heuristics under more datasets and under more heuristics. We believe that this will allow us to generalize our results and end up with a framework that can be applied *to any dataset given.* Moreover, we plan to do more work on the weights of the heuristics. Currently, weights are updated by the same, predefined percentage, every time a heuristic is selected. We plan to experiment with different setups of weight updating, such as linear and exponential. This would allow smaller changes at the beginning of the GP run, and bigger changes towards the end. We believe that this could have a significant effect on the hyper-heuristics framework, and lead to even better performance results. A way of doing the above could be to use a co-evolutionary approach for constructing the weight update scheme. For instance, another GP could be co-evolved to build a set of weight equations that are evaluated at various intervals on a sample test of the trees' population. This would thus allow for a variety of weight update schemes to be searched.

To sum up, this paper investigated the utility of applying hyper-heuristics to a well-known financial forecasting tool, EDDIE 8. Experimental results clearly showed that the EDDIE 8 system can be significantly improved by the addition of such heuristics. Future work entails the investigation of the optimal set of hyper-heuristics frameworks for EDDIE 8 and comparing this optimized system to there state-of-the-art financial forecasting algorithms.

Acknowledgements The version has been revised in light of two anonymous referees' very helpful reviews, for which the authors are most grateful.

References

1. Agapitos, A., O'Neill, M., Brabazon, A.: Evolutionary learning of technical trading rules without data-mining bias. In: R. Schaefer, C. Cotta, J. Kołodziej, G. Rudolph (eds.) Parallel Problem Solving from Nature – PPSN XI, *Lecture Notes in Computer Science*, vol. 6238, pp. 294–303. Springer (2010)
2. Allen, F., Karjalainen, R.: Using genetic algorithms to find technical trading rules. Journal of Financial Economics **51**, 245–271 (1999)
3. Austin, M., Bates, G., Dempster, M., Leemans, V., Williams, S.: Adaptive systems for foreign exchange trading. Quantitative Finance **4(4)**, 37–45 (2004)
4. Backus, J.: The syntax and semantics of the proposed international algebraic language of Zurich. In: International Conference on Information Processing, pp. 125–132. UNESCO (1959)
5. Baluja, S.: Population-based incremental learning: a method for integrating genetic search based function optimisation and competitive learning (1994). Technical Report, Carnegie Mellon University
6. Bernal-Urbina, M., Flores-Méndez, A.: Time series forecasting through polynomial artificial neural networks and genetic programming. In: Proceedings of the IEEE Congress on Evolutionary Computation, pp. 3324–3329. Hong Kong (2008)
7. Binner, J., Kendall, G., Chen, S.H. (eds.): Applications of Artificial Intelligence in Finance and Economics, *Advances in Econometrics*, vol. 19. Elsevier (2004)
8. Burke, E., MacCloumn, B., Meisels, A., Petrovic, S., Qu, R.: A graph-based hyper heuristic for timetabling problems. European Journal of Operational Research **176**, 177–192 (2006)
9. Cao, L., Tay, F.: Support vector machine with adaptive parameters in financial time series forecasting. IEEE Transactions on Neural Networks **14**, 1506–1518 (2006)
10. Chen, S.H.: Genetic Algorithms and Genetic Programming in Computational Finance. Springer-Verlag New York, LLC (2002)
11. Cowling, P., Chakhlevitch, K.: Hyperheuristics for managing a large collection of low level heuristics to schedule personnel. pp. 1214 – 1221 Vol.2 (2003). DOI 10.1109/CEC.2003.1299807
12. Dempsey, I., O'Neill, M., Brabazon, A.: Live trading with grammatical evolution. In: Proceedings of the Grammatical Evolution Workshop (2004)
13. Edwards, R., Magee, J.: Technical analysis of stock trends. New York Institute of Finance (1992)
14. Gestel, T., Suykens, J., Baestaens, D.E., Lambrechts, A., Lanckriet, G., Vandaele, B., Moor, B., Vandewalle, J.: Financial time series prediction using least squares support vector machines within the evidence framework. IEEE Transactions on Neural Networks **12**, 809–821 (2001)

15. Hart, E., Ross, P., Nelson, J.: Solving a real-world problem using an evolving heuristically driven schedule builder. Evol. Comput. **6**(1), 61–80 (1998)
16. Huang, W., Nakamori, Y., Wang, S.Y.: Forecasting stock market movement direction with support vector machine. Computers & Operations Research (2004)
17. Kablan, A.: Adaptive neuro fuzzy inference systems for high frequency financial trading and forecasting. pp. 105 –110 (2009). DOI 10.1109/ADVCOMP.2009.23
18. Kampouridis, M., Tsang, E.: EDDIE for investment opportunities forecasting: Extending the search space of the GP. In: Proceedings of the IEEE Conference on Evolutionary Computation, pp. 2019–2026. Barcelona, Spain (2010)
19. Kampouridis, M., Tsang, E.: Using hyperheuristics under a GP framework for financial forecasting. In: C.A. Coello Coello (ed.) Proc. Fifth International Conference on Learning and Intelligent Optimization (LION5), Lecture Notes in Computer Science 6683, pp.16–30. Springer, Heidelberg (2011).
20. Koza, J.: Genetic Programming: On the programming of computers by means of natural selection. Cambridge, MA: MIT Press (1992)
21. Larranaga, P., Lozano, J.: Estimation of Distribution Algorithms: A New Tool for Evolutionary Computation. Norwell, MA: Kluwer (2001)
22. Li, J.: FGP: A genetic programming-ased financial forecasting tool. Ph.D. thesis, Department of Computer Science, University of Essex (2001)
23. Martinez-Jaramillo, S.: Artificial financial markets: An agent-based approach to reproduce stylized facts and to study the red queen effect. Ph.D. thesis, CFFEA, University of Essex (2007)
24. Özcan, E., Bilgin, B., Korkmaz, E.E.: A comprehensive analysis of hyper-heuristics. Intelligent Data Analysis **12**(1), 3–23 (2008)
25. Poli, R., Langdon, W., McPhee, N.: A Field Guide to Genetic Programming. Lulu.com (2008)
26. Sapankevych, N., Sankar, R.: Time series prediction using support vector machines: A survey. Computational Intelligence Magazine, IEEE **4**(2), 24 –38 (2009). DOI 10.1109/MCI.2009.932254
27. Schulenburg, S., Ross, P.: Strength and money: An LCS approach to increasing returns, pp. 291–298. Lecture Notes in Computer Science. Springer Berlin / Heidelberg (2001)
28. Schulenburg, S., Ross, P.: Explorations in LCS models of stock trading, *P.L. Lanzi, W. Stolzmann, and S.W. Wilson, editors, Advances in Learning Classifier Systems: 4th International Workshop, IWLCS 2001*, vol. 2321, pp. 150–179. Springer Berlin / Heidelberg (2002)
29. Shachmurove, Y.: Business applications of emulative neural networks. International Journal of Business **10** (2005)
30. Sharma, V., Srinivasan, D.: Evolutionary computation and economic time series forecasting. In: Proceedings of the IEEE Conference on Evolutionary Computation, pp. 188–195. Singapore (2007)
31. Tino, P., Nikolaev, N., Yao, X.: Volatility forecasting with sparse bayesian kernel models. In: Proceedings of the 4th International Conference on Computational Intelligence in Economics and Finance (CIEF'05), pp. 1150–1153. Salt Lake City, Utah, USA (2005)
32. Tsang, E., Li, J., Markose, S., Er, H., Salhi, A., Iori, G.: EDDIE in financial decision making. Journal of Management and Economics **4(4)** (2000)
33. Tsang, E., Markose, S., Er, H.: Chance discovery in stock index option and future arbitrage. New Mathematics and Natural Computation, World Scientific **1(3)**, 435–447 (2005)
34. Tsang, E., Martinez-Jaramillo, S.: Computational finance. IEEE Computational Intelligence Society Newsletter pp. 3–8 (2004)
35. Wagner, N., Michalewicz, Z.: Adaptive and self-adaptive techniques for evolutionary forecasting applications set in dynamic and uncertain environments. Foundations of Computational Intelligence **4**, 3 – 21 (2009)
36. Wagner, N., Michalewicz, Z., Kouja, M., McGregor, R.: Time series forecasting for dynamic environments: The dyfor genetic program model. IEEE Transactions on Evolutionary Computation **11**(4), 433 – 452 (2007)
37. Wong, B., Selvi, Y.: Neural network applications in finace: A review and analysis of literature (1990-1996). Information & Management **34**, 129–139 (1998)

38. Yuan, X., Zou, Y.: Technology program financial forecast model based on caco-svm. In: Intelligent Systems and Applications, 2009. ISA 2009. International Workshop on, pp. 1 –4 (2009). DOI 10.1109/IWISA.2009.5073158
39. Zhang, Q., Sun, J., Tsang, E.: Evolutionary algorithm with guided mutation for the maximum clique problem. IEEE Transactions on Evolutionary Computation **9(2)**, 192–200 (2005)