Guided Fast Local Search for Speeding Up a Financial Forecasting Algorithm

Ming Shao*, Dafni Smonou*, Michael Kampouridis†, Edward Tsang*
*Centre for Computational Finance and Economic Agents, University of Essex, Wivenhoe Park, CO4 3SQ, UK
†School of Computing, University of Kent, ME4 4AG, UK

***Abstract*—Guided Local Search is a powerful meta-heuristic algorithm that has been applied to a successful Genetic Programming Financial Forecasting tool called EDDIE. Although previous research has shown that it has significantly improved the performance of EDDIE, it also increased its computational cost to a high extent. This paper presents an attempt to deal with this issue by combining Guided Local Search with Fast Local Search, an algorithm that has shown in the past to be able to significantly reduce the computational cost of Guided Local Search. Results show that EDDIE's computational cost has been reduced by an impressive 75%, while at the same time there is no cost to the predictive performance of the algorithm.**

I. Introduction

In the area of computational finance, financial forecasting plays a more and more important role and is widely applied in the industry [1]. Due to its importance, an increasing number of investors and researchers have been attracted to investigate in the field [2]–[5]. In recent years, with the latest technological advancements, many financial forecasting tools, which take advantage of computational intelligence, have been released [6]. A new version of a Genetic Programming (GP) financial forecasting tool called EDDIE-8 has been released [3], which is the extended version of EDDIE (Evolutionary Dynamic Data Investment Evaluator) [2], [3], [7], [8]. The new feature of EDDIE-8, as compared to its predecessor EDDIE-7, is the enriched and extended grammar of Genetic Decision Trees (GDTs) [3], [5], [7]. More details on EDDIE will be provided in Section II-B.

Results in [3], [9] have shown that better solutions have been reached by EDDIE-8 thanks to its extended grammar. However, this new grammar led to an enlarged search space, which could potentially lead to ineffective search. To be more specific, due to the significantly larger search space, it is easier for good solutions to be missed. One of the works that attempted tackling this issue was Smonou et al. [2]. This work implemented three Meta-heuristics (Simulated Annealing (SA), Tabu Search (TS), and Guided Local Search (GLS)), which were applied to the period nodes of the GP trees. Results showed that all of these meta-heuristics provided significant improvements to the predictive performance of EDDIE-8. Smonou et al. [2] also state that the GLS algorithm returned the best performance out of the three, significantly outperforming its opponents. However, a new issue arose, which was the algorithm's computational cost, as the GLS introduced a high number of additional fitness evaluations. The above is thus a drawback that needs to be tackled, as fast algorithms are of great importance in the field of financial forecasting.

Fast Local Search (FLS) [2], [3] is a heuristic, which has been proven helpful in improving the efficiency of GLS [10], as it has been found that it can significantly speed up the algorithm. Our goal thus in this paper is to implement FLS and combine it with GLS, as an attempt to reduce the computational time of GLS, while at the same time maintaining the algorithm's predictive performance at the same levels. The combination of GLS and FLS is called Guided Fast Local Search (GFLS) [10]. Doing the above is an important step towards making EDDIE an even better financial forecasting algorithm, which is vital, because of the significance of the financial forecasting field itself, which requires the continuous development of new and improved algorithms that can be used for real-life trading.

The rest of the paper is divided into five parts: Section II offers more detail on the EDDIE-8 algorithm. Section III presents the local search algorithms that are used as part of EDDIE-8. Section IV discusses the experimental designs, and Section V presents the results. Finally, Section VI concludes the paper and discusses potential future work.

II. EDDIE-8

EDDIE is a Genetic Programming financial forecasting tool. It attempts at answering the question: "Will the price of stock X go up by r% within the next n days?" [4], [11], [12]. This Section will present the process of the latest version of EDDIE, called EDDIE-8.

A. The general process of EDDIE-8

As already mentioned, EDDIE-8 is a GP algorithm. In this section, we are going to present the unique features of the algorithm, which are: data preparation, GDT initialization, and GDT evaluation. The rest of the algorithm is a standard GP process.

1) Data preparation: A time series of historical data need to be fed into system [2]. The data used are daily closing prices of stocks and indices and are freely available online such as in Yahoo Finance website http://finance.yahoo.com [2]. The time series is divided into two parts, which are the training and testing data. The training data are used to discover a GDT with the highest fitness value, which is then applied to the testing data for evaluation.

Secondly, six technical indicators [13] which have been implemented in EDDIE are the Moving Average (MA), Trade Break Out (TBR), Filter (FLR), Volatility (Vol), Momentum (Mom), and Momentum Moving Average (MomMA) [4]. The period values of indicators are inside the interval [MinP, MaxP]. More information about the interval is provided in Section IV. The actual values of each indicator over the training data are calculated.

Lastly, as mentioned earlier, EDDIE is trying to answer the question: "Will the price of stock X go up by r% within the next n days?" The actual signals of time series are calculated by looking ahead of n days [3]. The signals are assigned the value 1 if the prices do increase by r% within the coming n days, or 0 if they do not. The values r and n are specified by the user.

2) GDT initialization: To begin with, the Backus Normal Form (BNF) [2], [3] (grammar) is illustrated in Fig. II-A2. In addition, a very simple GDT is presented in Fig.II-A2.From the Fig.2 it can be seen that, the root of GDT is an "If-then-else" statement. The "If" branch is either a Boolean or a logic operator which eventually returns a Boolean value "True" or "False". If "True", it goes to "then" branch, which is another GDT, otherwise it goes to "else" branch which can be either a GDT or a decision. The decision is a constant value 1 or 0 which represents buy or not-to-buy respectively.

The VarConstructor statement has two children, which are a technical indicator and a period. The values of the indicators and periods are all randomly picked by the system from the interval [MinP, MaxP]. For the functionality, the number of combinations between indicator and period are much greater than earlier versions of EDDIE. Section II-B gives more detailed illustrations on this point.

<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 BNF of GDT [3].

3) GDT evaluation: The financial forecasting problem that EDDIE is dealing with is a typical classification problem. To evaluate the fitness of EDDIE's GDTs, we use the following performance metrics: Rate of Correctness (RC), Rate of Missing Chances (RMC), and Rate of Failure (RF). These are calculated by using Equations 1, 2 and 3 respectively, where TP represents the True Positive, TN the True Negative, FP the False Positive and finally FN the False Negative predictions.

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

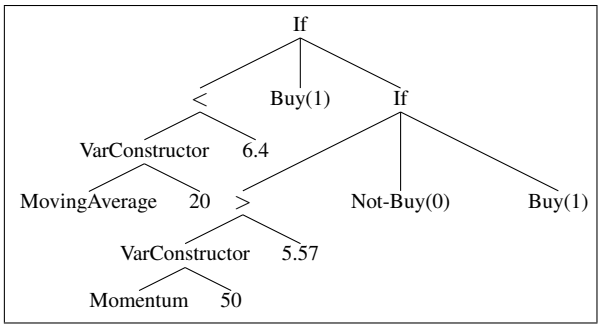

Fig. 2. Sample GDT generated by EDDIE-8.

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

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

These metrics are then combined in Equation 4, which is the fitness function. Different combinations of weights represent different types of investors [12]. For example, a conservative investor would prefer to avoid incorrect predictions by giving higher weight of RF [3], [12].

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

As explained, the rest of the algorithm follows a typical GP process. The experimental parameters of the GP will be presented in Section IV.

B. Extended search space

In the industry and the literature so far, predictions are made with the use of technical analysis indicators with pre-specified period lengths. For instance, when using the Moving Average (MA) indicator, traders usually use a short-term and a long-term MA, e.g., 12 days and 50 days MA. However, the problem with the above approach is that it does not guarantee that the selected periods are the optimal ones. Why should a 12 days MA be better across all possible datasets than a 15 days MA?

To overcome the above limitations, EDDIE-8 was introduced. As we saw in the previous section, EDDIE-8 is a more flexible algorithm, as it allows the GP to search in the space of the periods and create new technical indicators. Thus, the algorithm's user is not restricted in pre-specified indicators. The advantage of this approach was that it led to new and improved technical indicators, which had never been used before.

However, this flexibility led to an increased search space, which caused the GP to occasionally miss good solutions. To illustrate this, we'll give an example by comparing EDDIE-8 to its predecessor, EDDIE-7, which also uses the same six indicators, but with two fixed periods, 12 and 50 days (similar to what happens in other works in the literature, and also in the industry). If a given GDT has a maximum of k indicator nodes, then the number of possible combinations,

under EDDIE-7, for the GDT's indicators is $(6 \times 2)^k$. On the other hand, since EDDIE-8 uses the same six indicators with period values between the interval [2,65], which contains 64 period values, then the number of possible combinations is $(6{\times}64)^k$. Comparing these two numbers $(6{\times}2)^k$ and $(6{\times}64)^k$, it can be seen that the number of possible combinations for a GDT's indicators in EDDIE-8 is significantly larger than it is in EDDIE-7. As a result, while EDDIE-8 on average outperformed EDDIE-7, it could occasionally miss good solutions due to ineffective search.

C. Heuristics, Meta-heuristics and Hyper-heuristics

To date, a lot of research has been done in order to improve the search quality, and consequently, the performance of EDDIE-8. A significant step towards this was [2], where Smonou et al. applied three meta-heuristics, namely Simulated Annealing (SA), Tabu Search (TS), and Guided Local Search (GLS), to the period nodes of EDDIE-8. Results showed that the algorithm's prediction performance had significantly improved. In addition, what was remarkable was that the GLS algorithm led to 35 significant improvements in the best results of the performance metrics, and only to 4 diminutions. However, a disadvantage of the GLS approach was that it made EDDIE-8 significantly slower.

To address the above issue, our goal in this paper is to speed up the GLS algorithm. To achieve this, we will be using an algorithm called Fast Local Search (FLS), which is a heuristic that has been proven helpful in the past in improving the efficiency of GLS [10]. As has been shown by [10], FLS drastically speeds GLS up by redefining the neighborhood search process. The next section explains how these algorithms are implemented.

III. Methodology

In this section, we will first start by presenting how a Hill Climber (HC) was applied to EDDIE-8's period nodes. This is because Guided Local Search is a meta-heuristic that sits on top of HC. We will then move to the presentation of GLS and FLS.

A. Hill Climbing

In computer science, Hill Climbing (HC) is a "Mathematical Optimization" [14] technique and a branch of "Local Search" [14]. It is also an "Iterative Improvement" [15] heuristic that starts with a random solution, and attempts to improve it by iteratively changing one or more attributes of it. Russell et al. [15] mention that it is a practical approach and is more likely to generate better results to a problem [15]. However, the drawback of HC is the high possibility of getting stuck in local optima so it is not guaranteed to locate the best solution [15].

The process where HC is applied to EDDIE-8 is illustrated in the Algorithm 1. Firstly, a proportion of the GDTs population is randomly selected to undergo the HC process. Then, for each selected GDT all period nodes are identified. These will constitute the neighborhood of the HC. In the next step, the period nodes are visited in a random order and updated by a random marginal change k. For instance, if a given period is 23, and $k = +5$, then the period would be updated to 28. After this update, the fitness of the GDT is calculated and compared to the fitness before the period change. If the new fitness is higher, then the new period will be kept. This process is repeated for all period nodes of the selected GDTs, until the fitness function has failed to improve a number of m consequent times. The pseudo-code of HC is presented in Algorithm 1.

Algorithm 1 Pseudo code of HC. (Based on [15])

Input: GDT, original GDT wants to be improved;
maxFailure, local optimum condition;
failures ← 0;
while failures < maxFailure **do**
  pickedVarConstructor ← pickVarConstructor(GDT); // *Randomly picked a varConstructor node from inputted GDT.*
  fitnessBeforeHC ← Fitness(GDT); // *Calculate fitness value of inputted GDT.*
  k ← MarginChange(); // *Pick a random modification value.*
  GDT' ← copyGDT(K); // *Duplicate a GDT with new period value.*
  fitnessAfterHC ← Fitness(GDT');
  if fitnessAfterHC > fitnessBeforeHC **then**
    GDT ← GDT'; // *Keep new GDT if the fitness has been improved.*
  else
    failure++;
  end if
end while
return GDT;

B. Guided Local Search

Guided Local Search (GLS) is a meta-heuristic, which sits on top of local search methods (e.g. Hill Climbing) to change its behaviours in order to guide them to escape from local optima [10]. It is realized by using three terms (solution features, augmented fitness function and penalties) to confine the local search to focus on promising regions of search space [10].

First of all, the term *solution features* is employed by GLS to characterize solutions [10]. The domain of variables is divided into a number of non-overlapping and equally-sized intervals [16]. Every feature has a cost value (constant or variable), which indicates the extent of impact the corresponding solution property has [10]. The indicator function of solution features f_i is shown in Equation 5:

$$I_i(s) = \begin{cases} 1 & \text{solution s has property i} \\ 0 & \text{otherwise} \end{cases}, s \in S \quad (5)$$

In addition, in GLS, the Augmented Fitness Function (AFF) is used instead of EDDIE-8's fitness function (Equation 4), to guide the local search to escape from local maximum. The idea is to make the local maximum more costly, so that GLS has more chances to visit its neighbors [10]. The AFF is formed as shown in Equation [10]:

$$h(s) = g(s) - \lambda \times \sum_{i=1}^{M} p_i \times I_i(s) \quad (6)$$

where: g is original fitness function,
M is the number of features defined over solutions,
p_i is penalty parameter corresponding to feature f_i,
λ is regularization parameter and represents the importance of penalties.

Last but not least, whether the penalty parameter of feature f_i is modified depends on their utility values. Utility is calculated using Equation 7:

$$util(s, f_i) = I_i(s) \times \frac{c_i}{1 + p_i} \quad (7)$$

where: c_i is the cost value of feature f_i.

The penalty parameters are increased by one for the features that have the maximum utility value [10]. They record how many times the corresponding features have been penalized. From equation 7 it can be seen that the higher the cost value, the more likely the corresponding feature will be penalized. In contrast, the more times a particular feature has been penalized, the lower the utility of penalizing it again [17].

GLS has already been implemented on EDDIE by Smonou et al. in [2]. In their implementation GLS sits on top of a basic HC (as mentioned in section III-A), to guide it to escape from local maxima. The stopping criterion of GLS is when HC has been called 10 times. The procedure of GLS in EDDIE-8 is shown in Algorithm 3. At the beginning of GLS, indicators and penalties are initialized (Lines 2 - 5). In each iteration of GLS, the augmented fitness value is calculated (Line 8). Then HC is executed based on augmented fitness value (Lines 10 - 21). Lastly, the GLS indicators and penalties are adjusted based on a new GDT and utility value (Lines 22 - 28).

Algorithm 2 Pseudo code of GLS on EDDIE-8

Input: GDT, maxFailure
1: M ← 16; *// Initialize number of features.*
2: **for** i ← 1 until M **do**
3: I_i ← initIndicator(GDT); *// Initialize indicators based on GDT.*
4: p_i ← 0; *// Initialize penalties to 0.*
5: **end for**
6: k ← 0;
7: **while** k < 10 **do**
8: h ← g - $\lambda \times \sum_{i=1}^{M} p_i \times I_i(s)$;
9: failures ← 0;
10: **while** failures < maxFailure **do**
11: pickedVarConstructor ← pickVarConstructor(GDT); *// Randomly picked a varConstructor node from inputted GDT.*
12: fitnessBeforeHC ← Fitness(GDT); *// Calculate fitness value of inputted GDT.*
13: k ← MarginChange(); *// Pick a random modification value.*
14: GDT' ← copyGDT(K); *// Duplicate a GDT with new period value.*
15: fitnessAfterHC ← Fitness(GDT');
16: **if** fitnessAfterHC > fitnessBeforeHC) **then**
17: GDT ← GDT'; *// Keep new GDT if the fitness has been improved.*
18: **else**
19: failure++;
20: **end if**
21: **end while**
22: **for** i ← 1 until M **do**
23: $util(s, f_i) = I_i(s) \times \frac{c_i}{1 + p_i}$;
24: I_i ← initIndicator(GDT'); *// Update indicators based on GDT'.*
25: **end for**
26: **for** each i such that $util_i$ is maximum **do**
27: $p_i \leftarrow p_i + 1$;
28: **end for**
29: **end while**
30: **return** GDT;

C. Guided Fast Local Search

Prior to explaining the GFLS, it is important to mention the way FLS works. The current neighborhood is divided into a number of small sub-neighborhoods with a binary bit attached to each of them [10]. The idea is to scan the sub-neighborhoods with bit equals to 1 in a given order continuously. Initially, every bit is set to 1. The bit is switched to 0 if the corresponding sub-neighborhood is examined and does not make any improvements; otherwise the bit is kept as 1. As the solution improves, the number of bits, which equal to 1 gets fewer and fewer, and eventually the procedure ends up with every bit being 0.

As Voudouris [10] states, although FLS does not generally generate ideal results, it becomes a very powerful optimization algorithm (called GFLS) when combined with GLS. The idea is to associate solution features to sub-neighborhoods. In this case, the indicators are consistent with bits. Voudouris [10] also states that with FLS, GLS can be significantly sped up. Thus, for this reason FLS is decided to be implemented and embedded in GLS. The pseudo code of GFLS is illustrated in Algorithm 3.

IV. Experimental Design

A. Data

As mentioned in section II-A1, the datasets that have been used in the project are daily closing prices of stocks and indices. These stocks are from FTSE 100 (UK) and are the following: Aggreko, Barclays, British Petroleum, Cadbury, Carnival, Easyjet, First, Hammerson, Imperial Tobacco, Marks & Spencer, Next, Royal Bank of Scotland (RBS), Schroders, Sky, Tesco, Vodafone, and Xstrata. The indices are DJIA (USA), HSI (Hong Kong), MDAX (Germany), NASDAQ (USA), NIKEI (Japan) and NYSE (USA).

B. EDDIE-8 Parameters

The parameters of EDDIE-8 are kept the same as in [2], and [3], due to consistency purposes. The general experimental setting parameters are listed in Table I.

TABLE I
PARAMETERS OF EDDIE-8.

Parameters	Value
Training period	1000
Testing period	300
n days	20
{MinP, MaxP}	{2,65}

C. GP Parameters

The GP parameters are listed in Table II. All of them are kept the same as in [2] for the purpose of results comparison, even through the results are not so sensitive to changes in

Algorithm 3 Pseudo code of GLS combined with FLS.

Input: GDT
M ← 16; *// Initialize number of features.*
for i ← 1 until M **do**
  I_i ← initIndicator(GDT); *// Initialize indicators based on GDT.*
  p_i ← 0; *// Initialize penalties to 0.*
end for
k ← 0;
while k < 10 **do**
  h ← g - $\lambda \times \sum_{i=1}^{M} p_i \times I_i(s)$;
  for i ← 1 until M **do**
    $bit_i \leftarrow indicator_i$;
  end for
  while ∃ bit, bit=1 **do**
    pickedVarConstructor ← pickVarConstructor(GDT);
    period ← Period(pickedVarConstructor); *// Get period value from inputted pickedVarConstructor.*
    bitIndex ← Index(period); *// Get index number of inputted period.*
    if $bit_{bitIndex}$ =1 **then**
      fitnessBeforeHC ← Fitness(GDT); *// Calculate fitness value of inputted GDT.*
      k ← MarginChange(); *// Pick a random modification value.*
      GDT' ← copyGDT(K); *// Duplicate a GDT with new period value.*
      fitnessAfterHC ← Fitness(GDT');
      if fitnessAfterHC > fitnessBeforeHC **then**
        GDT ← GDT'; *// Keep new GDT if the fitness has been improved.*
        $bit_{bitIndex}$ =1; *// Set bit to 1 if the corresponding period has improved the fitness of GDT.*
      else
        $bit_{bitIndex}$ =0; *// Set bit to 0 if not.*
      end if
      for i ← 1 until M **do**
        $util(s, f_i) = I_i(s) \times \frac{c_i}{1 + p_i}$;
        I_i ← initIndicator(GDT'); *// Update indicators based on GDT'.*
      end for
      for each i such that $util_i$ is maximum **do**
        $p_i \leftarrow p_i + 1$;
      end for
    end if
  end while
end while
return GDT;

these parameters [3]. In order to analyze the results, GP is run 50 times for each dataset. The mean and the best values are calculated and recorded. The details of the results are demonstrated in Section V.

TABLE II
PARAMETERS OF GP.

GP parameters	Value
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

D. HC Parameters

As it was mentioned in section III-A, the HC would quit when there was a predefined number of failures in terms of fitness improvement. In previous works [2], [7], the maximum number of failures was set to $m = 10$ times. However, in our current work we were interested in investigating whether we could achieve further fitness improvements if we were to let the HC ran longer. Thus, we experimented by increasing "Failure" to 20 and 30. Results showed that there was indeed an improvement in the average fitness of the GDTs. However, we also found that the computational time of the algorithm was increasing linearly with the m number of "Failures". Therefore, we decided not to use anything higher than 30, as it was extremely computationally expensive. In this paper, the "Failure" value 30 will be used for the rest of our experiments, since it provides a good balance between the performance and the computational cost.

E. GLS/GFLS Parameters

As in previous works, GLS is applied to the period values. The period interval used by EDDIE-8 is [2,65]. Therefore, the solution features are defined in intervals of four consecutive numbers. For instance, periods [2, 3, 4, 5] will constitute Feature 1, periods [6, 7, 8, 9] will constitute Feature 2, and so on. There are in total 64 period values and can be divided into 16 features.

Moreover, the value of the regularization parameter λ, which is part of the AFF (Equation 6), is presented in Equation 8. According to [10], this is an effective and generalized calculation for λ.

$$\lambda = a \times \frac{g(s)}{|F(s)|} \tag{8}$$

V. RESULTS AND DISCUSSION

This Section demonstrates the results of EDDIE with GLS and EDDIE with GFLS. Each algorithm has been run 50 times for every dataset specifically for the purpose of the statistical analysis of the results. The computational times of each algorithm have been calculated and presented in Section V-A. Moreover, the results of our experiments are divided into Average and Best (Sections V-B and V-C respectively) and are presented by the four metrics of Fitness, RC, RMC and RF.

In addition to the results illustration, in order to reach more sufficient conclusions about the algorithms performance, the non-parametric Friedman Test [18], [19], has been applied. This test presents the average ranking of each algorithm, over all datasets tested in this work, for each performance metric (Fitness, RC, RMC, RF). The lower the ranking, the better the performance of the algorithm.

Last but not least, the post-hoc Bonferroni-Dunn test [20] has been implemented in order to measure the significance of the algorithms ranking under a 5% and a 10% significance level. The null hypothesis H_0 is rejected when the algorithms are significantly different in terms of ranking values.

A. Computational Cost

As it was mentioned in Section II-C, FLS is combined with GLS with the view to speed up the GLS complicated process. The idea is once a new GDT is examined and found not better

than the original GDT, it is then regarded as a valley; its neighbors are also then identified as being in a valley and thus are not visited. In order to calculate the average computational time for each algorithm, we looked into the 50 runs of a single dataset, which was the Aggreko dataset[1]. The average computational time of the two algorithms, EDDIE-GLS and EDDIE-GFLS, for a single run has been calculated and is presented in Table III.

TABLE III
AVERAGE COMPUTATIONAL TIME OF EDDIE-GLS AND EDDIE-GFLS.

Algorithm	Computational time
GLS	168.6774 minutes
GFLS	40.7612 minutes

By comparing the figures in Table III we can reach the conclusion that FLS has managed to help reducing the computational time of GLS by 75%, from 168 minutes down to 40. This is a significant improvement and meets the main goal of this paper, which was to reduce the computational times of GLS.

From the above figures, we can conclude that the effect of FLS seems is quite significant. Thanks to the breaking down of the neighborhood to sub-neighborhoods, the search focused only on the effective parts of the landscape. As a result, the GFLS algorithm was able to move significantly faster. We can thus conclude that FLS had indeed been very valuable in speeding up the algorithm.

In the next section we will be comparing the performance results between GLS and GFLS. We do not expect GFLS to be better than GLS, because as [10] said, GFLS does not generate ideal results; nevertheless, as long as these results are not significantly different than the GLS, then we can conclude that it is beneficial using this algorithm, due to the speed up improvements.

B. Average Results

Tables IV, V, VI and VII present the mean results of Fitness, RC, RMC and RF for the two algorithms. The best performing algorithm for a given metric is denoted with **bold** fonts, and the average ranking value of each algorithm produced by the Friedman Test is indicated in the last row of each Table.

As it can be observed by the average results, the GFLS algorithm has managed to maintain the average results of EDDIE at the same level as the GLS. Moreover, GLS has performed better in a total of 49 cases while GFLS in 44 cases. Therefore, both algorithms have similar performance in terms of average results. Furthermore, the average rankings of GFLS are better than the GLS in terms of Fitness and RF, and the GFLS ranks approximately the same with the GLS in RC. However, with the use of the post-hoc Bonferroni-Dunn test all differences in the average rankings were proven to be non significant at 5% level. Therefore at this point we can argue that GFLS has managed to maintain the same good level of performance in terms of average results, but with significantly reduced computational cost.

[1]Since the datasets have exactly the same length and the GP experiments share the same parameters in all datasets, we do not expect the computational times to differ among the datasets.

TABLE IV
THE RESULTS OF MEAN FITNESS.

Dataset	GLS	GFLS
Aggreko	**0.1211**	0.1038
Barclays	0.1729	**0.1765**
BP	**0.1854**	0.1781
Cadbury	0.2560	**0.2629**
Carnival	**0.1248**	0.1063
DJIA	**0.2872**	0.2817
Easyjet	**0.1269**	0.1252
First	0.1623	**0.1628**
Hammerson	**0.1323**	0.1231
HSI	0.2536	**0.2599**
Imp	**0.2046**	0.1999
Marks&Spencer	0.1152	**0.1240**
MDAX	**0.1230**	0.1141
NASDAQ	0.1874	**0.1983**
Next	**0.1331**	0.1171
NIKEI	0.1515	**0.1538**
NYSE	**0.1730**	0.1385
RBS	0.1639	**0.1690**
Schroders	0.1881	**0.1894**
Sky	0.1675	**0.1813**
Tesco	**0.2377**	0.2343
Vodafone	0.1005	**0.1265**
Xstrata	0.2180	**0.2227**
Ranking	**1.5217**	**1.4783**

TABLE V
THE RESULTS OF MEAN RC.

Dataset	GLS	GFLS
Aggreko	**0.5074**	0.4887
Barclays	0.5556	**0.5598**
BP	**0.5429**	0.5351
Cadbury	0.6316	**0.6388**
Carnival	**0.5381**	0.5229
DJIA	**0.6494**	0.6430
Easyjet	**0.4657**	0.4646
First	0.5033	**0.5042**
Hammerson	**0.5135**	0.5043
HSI	0.6135	**0.6193**
Imp	**0.5664**	0.5577
Marks&Spencer	0.4796	**0.4875**
MDAX	**0.4963**	0.4908
NASDAQ	0.5400	**0.5521**
Next	**0.4861**	0.4688
NIKEI	**0.5313**	0.5294
NYSE	**0.5462**	0.5153
RBS	0.5287	**0.5349**
Schroders	0.5573	**0.5589**
Sky	0.5231	**0.5368**
Tesco	**0.5942**	0.5897
Vodafone	0.4842	**0.5163**
Xstrata	0.5640	**0.5716**
Ranking	**1.4783**	**1.5217**

TABLE VI
THE RESULTS OF MEAN RMC.

Dataset	GLS	GFLS
Aggreko	**0.3088**	0.3307
Barclays	**0.4161**	0.4423
BP	**0.4162**	0.4226
Cadbury	0.2061	**0.1880**
Carnival	**0.1435**	0.1926
DJIA	**0.1642**	0.1805
Easyjet	**0.6340**	0.6382
First	0.5347	**0.5272**
Hammerson	**0.4807**	0.4828
HSI	**0.2178**	0.2273
Imp	**0.4696**	0.5027
Marks&Spencer	0.5139	**0.4953**
MDAX	**0.2098**	0.2529
NASDAQ	0.4229	**0.3972**
Next	**0.4808**	0.5181
NIKEI	0.2806	**0.2661**
NYSE	**0.2119**	0.3184
RBS	0.3896	**0.3755**
Schroders	**0.3217**	0.3430
Sky	**0.4468**	0.4510
Tesco	**0.3154**	0.3341
Vodafone	0.3217	**0.3182**
Xstrata	0.3190	**0.2913**
AverageRank	**1.3478**	**1.6522**

TABLE VII
THE RESULTS OF MEAN RF.

Dataset	GLS	GFLS
Aggreko	**0.5081**	0.5211
Barclays	0.3962	**0.3840**
BP	**0.3292**	0.3357
Cadbury	0.3412	**0.3387**
Carnival	**0.6124**	0.6273
DJIA	**0.2867**	**0.2867**
Easyjet	**0.2971**	0.2991
First	**0.2872**	0.2901
Hammerson	**0.4258**	0.4371
HSI	0.3092	**0.2967**
Imp	0.2942	**0.2813**
Marks&Spencer	0.4040	**0.3966**
MDAX	**0.5127**	0.5171
NASDAQ	0.3145	**0.3108**
Next	**0.3683**	0.3745
NIKEI	0.4642	**0.4575**
NYSE	**0.4451**	0.4627
RBS	0.3830	**0.3812**
Schroders	0.3803	**0.3721**
Sky	0.3388	**0.3189**
Tesco	0.2911	**0.2868**
Vodafone	0.5262	**0.5050**
Xstrata	**0.2951**	0.3037
AverageRank	**1.5217**	**1.4783**

C. Best Results

Tables VIII, IX, X and XI present the best results of each algorithm for the four individual metrics Fitness, RC, RMC

and RF. Once again, the best performing algorithm will be denoted with **bold** fonts, and the average ranking value of the algorithms is included in the bottom row of each Table.

As it can be observed from Table VIII (Best Fitness), there are certain cases where GFLS has managed to improve the best fitness of the datasets impressively in comparison to GLS. More specifically, on the BP dataset, GFLS has improved the best fitness of BP by 6.4% on Next by 14.5%, on RBS by 7%, on Schroders by 6% and on Xstrata by 8%. Finally even though the ranking of the GFLS is better than the GLS, this has been proven to be non significant at a 95% confidence level.

In addition, the above pattern can also be found in the remaining Best results (RC, RMC, RF), which are presented in Tables IX - XI. The GFLS has again offered some impressive improvements: Barclay's RMC by 13%, BP's RMC by 12%, Next's RC and RMC by 15 and 18%, respectively, NIKEI's RC and RMC by 19 and 36%, respectively, and Xstrata's RMC by 14%. However, the overall ranking for each of the three metrics is again close for GLS and GFLS, and insignificant at 5% level. Therefore, all of the best results observations lead us to argue that GFLS has maintained the same good performance of GLS under all four metrics, in terms of Best results.

TABLE VIII
THE RESULTS OF BEST FITNESS.

Dataset	GLS	GFLS
Aggreko	**0.1916**	0.1183
Barclays	0.1679	**0.1730**
BP	0.1025	**0.1668**
Cadbury	**0.2732**	0.2032
Carnival	**0.1746**	0.1166
DJIA	**0.2987**	0.2134
Easyjet	**0.2300**	0.1981
First	**0.1654**	0.0934
Hammerson	**0.1379**	0.0920
HSI	0.2711	**0.2830**
Imp	**0.1999**	0.1509
Marks&Spencer	**0.1293**	0.1292
MDAX	**0.1432**	0.0951
NASDAQ	**0.2317**	0.2076
Next	0.0478	**0.1929**
NIKEI	-0.078	**0.1442**
NYSE	**0.1799**	0.1582
RBS	0.1597	**0.2285**
Schroders	0.1646	**0.2267**
Sky	**0.1897**	0.1655
Tesco	**0.2672**	0.2304
Vodafone	0.0621	**0.1054**
Xstrata	0.1082	**0.1953**
AverageRank	**1.3913**	**1.6087**

TABLE IX
THE RESULTS OF BEST RC.

Dataset	GLS	GFLS
Aggreko	**0.5933**	0.5200
Barclays	0.5533	**0.5567**
BP	0.4567	**0.5233**
Cadbury	**0.6500**	0.5733
Carnival	**0.6000**	0.5333
DJIA	**0.6633**	0.5633
Easyjet	**0.5867**	0.5500
First	**0.5100**	0.4267
Hammerson	**0.5167**	0.4733
HSI	0.6333	**0.6467**
Imp	**0.5667**	0.5133
Marks&Spencer	**0.4933**	**0.4933**
MDAX	**0.5067**	0.4800
NASDAQ	**0.5900**	0.5633
Next	0.4000	**0.5467**
NIKEI	0.3367	**0.5300**
NYSE	**0.5600**	0.5267
RBS	0.5233	**0.5967**
Schroders	0.5333	**0.6000**
Sky	**0.5400**	0.5200
Tesco	**0.6267**	0.5867
Vodafone	0.4267	**0.5067**
Xstrata	0.4433	**0.5400**
AverageRank	**1.4130**	**1.5870**

D. Further Discussion

From the results presentation it was observed that GFLS has notably reduced the computational cost of GLS, while maintaining similar performance to GLS, as all rankings were proven non significant at a 95% confidence level. This is an important achievement, because not only GFLS is significantly faster than GLS, it has also proven to be competitive as well.

TABLE X
THE RESULTS OF BEST RMC.

Dataset	GLS	GFLS
Aggreko	**0.3403**	0.4375
Barclays	0.5482	**0.4217**
BP	0.5436	**0.4205**
Cadbury	**0.1508**	0.2737
Carnival	**0.1304**	0.1739
DJIA	**0.1095**	0.2333
Easyjet	**0.3153**	0.3990
First	**0.4686**	0.6908
Hammerson	**0.4379**	0.5325
HSI	0.1659	**0.1024**
Imp	**0.3243**	0.5892
Marks&Spencer	0.5591	**0.5323**
MDAX	**0.1020**	0.3605
NASDAQ	**0.3134**	0.3234
Next	0.6364	**0.4545**
NIKEI	0.7764	**0.4161**
NYSE	0.2840	**0.2037**
RBS	**0.3587**	0.4130
Schroders	0.3833	**0.3444**
Sky	0.5385	**0.5077**
Tesco	**0.2990**	0.3039
Vodafone	**0.2587**	0.4406
Xstrata	0.4860	**0.3458**
AverageRank	**1.4348**	**1.5652**

TABLE XI
THE RESULTS OF BEST RF.

Dataset	GLS	GFLS
Aggreko	**0.4345**	0.5000
Barclays	**0.3644**	0.3962
BP	0.3904	**0.3506**
Cadbury	**0.3391**	0.3780
Carnival	**0.5745**	0.6200
DJIA	**0.2943**	0.3374
Easyjet	**0.3015**	0.3068
First	0.3125	**0.3118**
Ham	**0.4277**	0.4626
HSI	**0.3077**	0.3160
Imp	0.3590	**0.3274**
M&S	**0.3692**	0.3786
MDAX	**0.5019**	0.5228
NASDAQ	**0.3030**	0.3267
Next	0.4286	**0.2987**
NIKEI	0.6727	**0.4405**
NYSE	**0.4257**	0.4580
RBS	0.3949	**0.2941**
Schrod	0.3901	**0.3295**
Sky	**0.2683**	0.3191
Tesco	**0.2629**	0.3039
Vodafone	0.5602	**0.5152**
Xstrata	0.3642	**0.3137**
AverageRank	**1.3913**	**1.6087**

Speeding up financial forecasting algorithms is extremely valuable, as it offers the investors important advantages against their competitors. In an era where a big proportion of trading is done algorithmically, it is crucial to have fast algorithms. Therefore, a speed-up of an already successful algorithm in a scale of 75% gives an additional advantage to its users, since they can obtain trees in a timelier manner. As a result, this could lead to potentially higher profit margins.

VI. CONCLUSION

This paper presented work on the application of Fast Local Search (FLS) algorithm to a meta-heuristic called Guided Local Search (GLS), which was originally applied to a Genetic Programming (GP) financial forecasting algorithm called EDDIE 8. The feature that made EDDIE 8 unique was its ability to search in the space of technical indicators for solutions, instead of using specific ones. As a result EDDIE 8's search area has increased dramatically leading to occasionally missed solutions. GLS was applied as part of attempts to tackle with this issue. Although it was proven beneficial for EDDIE 8, it has increased the computational time of the algorithm to a high extent.

In order to address this problem, a combination of FLS and GLS has been implemented to the period nodes of EDDIE8's Genetic Decision Trees. Results showed that GFLS sped up the GLS algorithm by an impressive 75%, while maintaining similar predictive performance in both average and best results. This thus led us to believe that the combination of FLS and GLS is a very valuable addition to the EDDIE-8 algorithm. As we explained, the importance of the above achievement lies on the fact that investors gain a significant competitive advantage that can potentially lead to higher profit.

GFLS could be quite a market potential meta-heuristic in the financial forecasting industry. Our next goal would be to reduce its computational cost and improve its performance with EDDIE even more by redefining its neighborhood function and test it under more datasets. Furthermore, GFLS could be combined with other algorithms under Hyper-heuristic frameworks, in order to combine the advantages of different meta-heuristics under the umbrella of a single algorithm.

REFERENCES

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

[2] D. Smonou, M. Kampouridis, and E. Tsang, "Metaheuristics application on a financial forecasting problem," in *Proceedings of the IEEE Congress on Evolutionary Computation*, Cancun, Mexico, 20-23 June 2013.

[3] M. Kampouridis and E. Tsang, "EDDIE for investment opportunities forecasting: Extending the search space of the GP," in *Proceedings of the IEEE Congress on Evolutionary Computation*, Barcelona, Spain, 18-23 July 2010, pp. 2019–2026.

[4] E. Tsang, P. Yung, and J. Li, "EDDIE-automation, a decision support tool for financial forecasting," *Decision Support Systems*, vol. 37, no. 4, pp. 559–565, 2004, data mining for financial decision making. [Online]. Available: http://cswww.essex.ac.uk/CSP/finance/papers/TsYuLi-Eddie-Dss2004.pdf

[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] M. Bernal-Urbina and A. Flores-Mendez, "Time series forecasting through polynomial artificial neural networks and genetic programming," in *Neural Networks, 2008. IJCNN 2008. (IEEE World Congress on Computational Intelligence). IEEE International Joint Conference on*, 2008, pp. 3325–3330.

[7] 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.

[8] M. Kampouridis, "An initial investigation of choice function hyper-heuristics for the problem of financial forecasting," in *Proceedings of the IEEE Congress on Evolutionary Computation (CEC)*, Cancun, Mexico, 2013.

[9] 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.

[10] C. Voudouris, "Guided local search for combinatorial optimisation problems," Ph.D. dissertation, Department of Computer Science, University of Essex, UK, 1997. [Online]. Available: ftp://ftp.essex.ac.uk/pub/csp/Voudouris-PhD97-pdf.zip

[11] E. Tsang, J. Li, S. Markose, H. Er, A. Salhi, and G. Iori, "EDDIE in financial decision making," *Journal of Management and Economics*, 2000. [Online]. Available: http://cswww.essex.ac.uk/CSP/finance/papers/Tsang-Eddie-JMgtEcon2000.ps

[12] J. Li, "Fgp: A genetic programming based financial forecasting tool," Ph.D. dissertation, School of Computer Science and Electronic Engineering, University of Essex, UK, 7 October 2000. [Online]. Available: http://www.kampouridis.net/papers/thesis.pdf

[13] F. Allen and R. Karjalainen, "Using genetic algorithms to find technical trading rules," *Journal of Management and Economics*, pp. 245–271, 1999. [Online]. Available: http://finance.wharton.upenn.edu/~rlwctr/papers/9520.pdf

[14] G. Kendall, "Artificial intelligence methods, an overview of search algorithms," Lecture slide of moodle G5BAIM, University of Nottingham, 2013. [Online]. Available: www.cs.nott.ac.uk/~gxk/courses/g5baim/006hc/powerpoint/hc.ppt?

[15] S. J. Russell, P. Norvig, J. F. Candy, J. M. Malik, and D. D. Edwards, *Artificial intelligence: a modern approach*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 1996.

[16] C. Voudouris, "Guided local search - an illustrative example in function optimisation," in *In BT Technology Journal, Vol.16, No.3*, 1998, pp. 46–50.

[17] E. Tsang and C. Voudouris, "Fast local search and guided local search and their application to british telecom's workforce scheduling problem," Operations Research Letters, Tech. Rep., 1995.

[18] J. Demšar, "Statistical comparisons of classifiers over multiple data sets," *J. Mach. Learn. Res.*, vol. 7, pp. 1–30, Dec 2006. [Online]. Available: http://dl.acm.org/citation.cfm?id=1248547.1248548

[19] S. Garcia and F. Herrera, "An extension on statistical comparisons of classifiers over multiple data sets for all pairwise comparisons," *Journal of Machine Learning Research*, vol. 9, pp. 2677–2694, 2008. [Online]. Available: http://jmlr.csail.mit.edu/papers/v9/garcia08a.html

[20] O. J. Dunn, "Multiple comparisons among means," *Journal of the American Statistical Association*, vol. 56, no. 293, pp. 52–64, 1961. [Online]. Available: http://dx.doi.org/10.2307/2282330