Metaheuristics Application on a Financial Forecasting Problem

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

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

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

***Abstract*- EDDIE is a Genetic Programming (GP) tool, which is used to tackle problems in the field of financial forecasting. The novelty of EDDIE is in its grammar, which allows the GP to look in the space of technical analysis indicators, instead of using pre-specified ones, as it normally happens in the literature. The advantage of this is that EDDIE is not constrained to use pre-specified indicators; instead, thanks to its grammar, it can choose any indicators within a pre-defined range, leading to new solutions that might have never been discovered before. However, a disadvantage of the above approach is that the algorithm's search space is dramatically larger, and as a result good solutions can sometimes be missed due to ineffective search. This paper presents an attempt to deal with this issue by applying to the GP three different meta-heuristics, namely Simulated Annealing, Tabu Search, and Guided Local Search. Results show that the algorithm's performance significantly improves, thus making the combination of Genetic Programming and meta-heuristics an effective financial forecasting approach.**

I. INTRODUCTION

Financial forecasting is a well-known and applied method in the industry. Its importance has led investors and researchers to focus on the creation of more efficient ways to apply financial forecasting. In the field of Computational Intelligence, a new financial forecasting tool called EDDIE 8 (ED8) has been presented [1]. ED8, which is an extended version of EDDIE (Evolutionary Dynamic Data Investment Evaluator) [2], uses Genetic Programming [3], [4] in order to make predictions. The new feature of this version, in comparison to its predecessor EDDIE 7 (ED7), and to every other financial forecasting tool in the literature, was its extended grammar (BNF) [5], which provided the algorithm with the ability to search in the space of technical analysis[1] indicators to form Genetic Decision Trees (GDTs).[2]

In ED7 the indicators used were limited and pre-specified by the user, for instance 12 and 50 days Moving Average. On the other hand ED8 was not limited to the above periods and was able to choose between a range of periods specified by the user, for instance any period between the range of 2 and 65 days Moving Average. The GP would then be responsible for searching the above period range and suggest appropriate periods for the technical indicators.

Results in [1], [6] showed that thanks to its extended grammar, ED8 could reach new and improved solutions. However, it was observed that occasionally the performance could be compromised, as a consequence of this new grammar. This was because ED8's search space had dramatically increased, and thus ED8 could not always search effectively its large search space. This will be discussed in more details in Section II-B. In order to overcome the problem of ineffective search, several heuristics were applied by Kampouridis et al. in [7], and then combined into different hyper-heuristics frameworks, which were proven beneficial for the performance of EDDIE.

However, only a few low-level heuristics were examined in [7], e.g. different implementations of hill-climbers. While hill climbing is an effective search algorithm, it is also known for getting stuck in local optimums. Furthermore, [7] also concluded that more search algorithms should be applied to ED8, to investigate if further performance improvements could take place. To this extend, this paper applies three search algorithms, namely Simulated Annealing, Tabu Search, and Guided Local Search. The advantage of these algorithms is that they fall in the category of meta-heuristics, thus have the potential of overcoming local optima [8], [9], [10]. Our goal is to show that as a result of the application of the above meta-heuristics, ED8's search effectiveness can significantly improve and lead to even better solutions, thus make EDDIE a beneficial algorithm for the financial forecasting community.

The rest of this paper is as follows: Section II presents EDDIE and the purpose of using heuristics, Section III focusses on the metaheuristics applied for the purpose of this project, Section IV analyses the experimental design, Section V includes a presentation and discussion of the experimental results, and finally, Section VI concludes this paper.

II. EDDIE

Computational Intelligence (CI) techniques have been extensively used for financial forecasting. Genetic Programming (GP) [3], [4] is a CI technique that has received much attention for this type of problems. Some examples of recent GP applications for financial forecasting are: [11], [12]. EDDIE is a financial forecasting algorithm that uses GP to evolve trading strategies and predict future movements of the stock market. The rest of Section II presents EDDIE in detail.

[1] Technical analysis is a financial forecasting method, used to predict a future movement based on existing patterns.

[2] Due to the fact that the decision trees created by EDDIE were a result of Genetic Programming, they were referred as 'Genetic Decision Trees'.

A. THE GENERAL EDDIE PROCESS

This Section focusses on providing some basic information about the way EDDIE works and specifically the EDDIE 7 (ED7) version, which EDDIE 8 (ED8) is extending.

For starters, EDDIE tries to answer the question: *"Will the price of a stock X increase by r% within the next n days?"* The algorithm uses three basic inputs: technical analysis indicators, historical data (daily closing prices of stocks and indices) and binary target signals of buy or not-to-buy (1, 0). The twelve indicators used from technical analysis in ED7 are the "Moving Average" of 12 and 50 days, the "Trade Break Out" of 12 and 50 days, the "Filter" of 12 and 50 days, the "Volatility" of 12 and 50 days, the "Momentum" of 12 and 50 days and finally the "Momentum Moving Average" of 12 and 50 days. The common factors of these are the pre-specified periods of 12 and 50 days which cannot change in this version of EDDIE; therefore the indicators are considered as constants. Additionally, the historic data used, can be obtained from online websites, for instance *http://finance.yahoo.com*. Last but not least, the signals are estimated by looking *n* days ahead of the closing price and by checking to see if the price has risen by *r%*.

Furthermore, GP is used as a basic tool for EDDIE. A population of Genetic Decision Trees (GDTs) is generated randomly and these trees are evolved for a number of generations. The grammar (BNF) of ED7 is illustrated in Fig.1. A point that needs to be highlighted is the *"<Variable>",* which can be any of the twelve pre-specified technical indicators mentioned earlier. However, the fact of using these pre-specified indicators was considered to be a limitation of EDDIE, and the first attempt to tackle this issue took place in ED8 [1], as we will see in Section II-B.

```
<Tree> ::= If-then-else <Condition> <Tree> <Tree> | Decision
<Condition> ::= <Condition> "And" <Condition> |
                <Condition> "Or" <Condition> |
                "Not" <Condition> |
                Variable <RelationOperation> Threshold
<Variable> ::= MA 12 | MA 50 | TBR 12 | TBR 50 | FLR 12 |
               FLR 50 | Vol 12 | Vol 50 | Mom 12 | Mom 50 |
               MomMA 12 | MomMA 50
<RelationOperation> ::= ">" | "<" | "="
Decision is an integer, Positive or Negative implemented
Threshold is a real number
```

Fig. 1. The Backus Normal Form of ED7 [6].

The evaluation of EDDIE's GDTs is based on the Confusion Matrix [13], from which we derive three performance measures:

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

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

Those three metrics RC, RMC, RF are combined to create the fitness function which was defined as:

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

where w_1, w_2, w_3 are the weights for RC, RMC and RF, chosen specifically to reflect investor's preferences [14].

B. EDDIE 8

ED8 was created to overcome ED7's limitations of using pre-specified period indicators. Instead, ED8 would allow the algorithm to choose any periods within a specific range.

As was described in the previous section, ED7 used 6 indicators with 2 pre-specified periods (12 and 50 days). On the contrary, as it can be seen in Fig. 2, instead of the constant *"<Variable>"* ED8 now uses a function called *"<VarConstructor>"*. This new feature is a function that takes two children, the "Indicator" and "Period". The indicators are the same with the previous version; however, the periods can now take any values within a range *[P_{min}, P_{max}]*. Consequently, ED8 can now produce GDTs which contain indicators such as MA 18 days, Mom of 46 days and so on. This is an important difference from ED7, because it made the algorithm much more flexible and dynamic. ED8 has many more options available, instead of being restricted to only 12 indicators, and it is up to the GP and the evolutionary process to determine the best periods for each decision tree.

```
<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> ::= ">" | "<" | "="
          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. 2. The Backus Normal Form of ED8 [6].

However a serious issue that arose was that the new grammar had dramatically increased the search space of the GP. As it was explained in [7], if a given GDT can have a maximum of k indicators then, the permutations of the 12 indicators (6 indicators * 2 periods) under ED7 are 12^k. On the contrary, if ED8 is using the same 6 indicators with periods within the range of [2, 65] days, then the permutations of the 384 indicators (6 indicators *64 periods) are 384^k. ED8's search space was now significantly larger than ED7's; therefore, ED8 could occasionally miss good solutions due to ineffective search. For that reason, it was decided that different heuristics should be applied to the leaf periods of the trees, in an attempt to make the search more effective.

C. HEURISTICS AND METAHEURISTICS

In order to deal with ED8's problem of ineffective search, Kampouridis et al. [7] combined certain heuristics under hyper-heuristic frameworks. In that work, 14 low-level heuristics were applied to 30 datasets on ED8. Those heuristics were applied to the indicators and the periods.[3] The rationale behind this was that these heuristics would offer exploitation of the search area of the technical indicators and their periods.

Moreover, the best of those heuristics (in terms of performance) were chosen to be combined under three different hyper-heuristics frameworks. The results were very promising as all three hyper-heuristics improved the average and best solutions; therefore they significantly improved the overall performance of ED8. That was the first time that hyper-heuristics had been applied to financial forecasting.

However, it was also pointed out that the low-level heuristics used in [7] (different hill-climbers and mutators) were not an exhaustive list of heuristics, and that other search algorithms could offer even more effective search.

The above can be particularly true for meta-heuristics, which are designed with the goal of over-coming local optima. *Our purpose thus in this paper is to experiment with three such meta-heuristics and investigate their effect to the performance of ED8.* The next section presents these meta-heuristics and also explains how they were applied in the EDDIE framework.

III. METHODOLOGY

As explained earlier, in order to improve ED8's search, we will be applying different meta-heuristics to the trees' period nodes. Thus, at every generation a number of trees is selected, and then a meta-heuristic is applied to each tree.

This Section describes the implementation details for these meta-heuristics, namely Simulated Annealing (SA), Tabu Search (TS) and Guided Local Search (GLS). The reason behind the choice of the above algorithms is their proven good performance within various research fields [15], [16], [17]. In addition, due to the fact that SA and TS are well-known algorithms, we will not focus on general descriptions of them, but on explaining how they behave in the EDDIE framework.

A. SIMULATED ANNEALING

Simulated Annealing (SA) is a meta-heuristic that allows the local search to probabilistically visit worse solutions, with the view that other solutions in the same neighborhood may provide a better overall solution. More detailed information about SA and its application to genetic algorithms can be found in [8], [18].

On EDDIE, SA was applied to the periods of the indicators. The probability of acceptance initially receives a high value, which is then gradually decreased throughout the iterations (the maximum number of which is represented by the name "*kmax*"), tending to zero. The probability function with which a worse solution can be accepted is:

$$P = e^{(\text{ new fitness - old fitness}) / \text{temperature}} \qquad (5)$$

The SA process for EDDIE is as follows: first, a tree is probabilistically selected. Then, all period branches of the tree are identified and form the neighborhood of the GDT. A neighbor can then be obtained by making a marginal change (k) to the value of the periods, such as $-10 \leq k \leq 10$. The search starts from an initial solution (a random period branch is selected), and then iteratively moves to other solutions of the same or other period branches of the tree. The SA principles, where worse solutions can probabilistically be accepted, are applied. Fig. 3 summarizes a general SA process tailored to EDDIE.

```
t ← GDT(s)                                  // Probabilistically select a tree.
neighborhood (s)                            //All period branches of "t".
s ← s0; e ← E(s)                            // Initial periods, fitness.
sBest ← s; eBest ← e                        // Initial "best" solution.
k ← 0                                       // Energy evaluation count.
while k < kmax                              // While time left.
  T ← temperature(k/kmax)                   // Temperature calculation.
  sOld ← neighbor(s)                        // Pick some period branch.
  sNew ← sOld+rand(-k:k)                    // Marginal change to period.
  eNew ← E(sNew)                            // Compute its fitness.
  if P(e, eNew, T) > random() then          // Probabilistically accept
                                            //new solution.
     s ← sNew; e ← eNew                     // Yes, change period.
  end
  if e >eBest then                          // Is this a new best?
    sBest ← sNew; eBest ← eNew              // Save 'new neighbor'.
  end
  k ← k + 1                                 // One more evaluation done.
end
return sBest                                // Return the best solution found.
```

Fig. 3. SA Pseudocode (Based on: [19])

B. TABU SEARCH

With the application of Tabu Search (TS) we aim to overcome the problem of local search, by allowing the search to focus on other areas that are believed to contain better solutions. In other words with TS the search is guided in such way so that it is not likely to get stuck in a local optima [9], [20], [21], [22], [23].

On EDDIE, TS has been applied on the periods, like SA. The process starts again with the random selection of a tree. Then, all period branches of the tree are identified and form the neighborhood of the GDT. A neighbor can then be obtained by making a marginal change (k) to the value of the periods, such as $-10 \leq k \leq 10$. Additionally, it is checked whether this new period is contained in the aspiration criteria.[4] If yes, then the new fitness is calculated and if it is better than the old one, the new period replaces the old one. If it is worse, the new period is discarded. Additionally, if the period is not

[3] As mentioned earlier, with the term "indicators" we refer to the 6 technical indicators used by EDDIE such as Moving Average, Momentum etc., whereas with the term "periods" we refer to the periods of those indicators that could take any integer value between a specific range for instance [2, 65]. We should note that for the purposes of this paper, we will only focus to period-based heuristics, i.e., heuristics that apply local search only to the period (e.g., 12 days) and not to the indicator (e.g., Moving Average). We have left the latter investigation as a future work.

[4] The "aspiration criteria" refer to the solutions (periods) that we believe to be promising for better fitness. The solutions that are included in the aspiration criteria can be tested even if they are part of the tabu list.

in the aspiration criteria, we check whether it is a tabu[5]. If yes then we discard it, otherwise the new fitness is calculated and if it is better, the new period is kept. If it is worse the new period is discarded. This process is repeated until the termination criteria are met.

Before each iteration the tabu list and the aspiration criteria are updated accordingly. Specifically, a period along with its r closest neighbors will be added to the tabu list as soon as it is visited. By the r closest neighbors, we mean the $[-r/2, +r/2]$ area of the period. For instance if we currently examine the period 10, and $r=4$ in the tabu list we will add the periods 8, 9, 10, 11, 12. The advantage of this is that, for short term, the algorithm will be prevented from revisiting solutions that are very similar to the one already tested. Furthermore, if a period provides better fitness, it is added to the “aspiration criteria” list along with its m closest neighbors ($[-m, +m]$ area). This enables the search to visit those periods again, despite the fact that it could be part of the “tabu list”.

Fig. 4 summarizes a TS process tailored to EDDIE. Section IV-B provides more information on the parameter values.

```
t ← GDT(s)                                   // Probabilistically select a tree.
neighborhood (s)                             //All period branches of "t".
s ← s0                                       // Initial periods.
sBest ← s                                    // Initial "best" solution.
tabuList ← null                              // Initialize empty Tabu List.
aspirCriteria ← null                         // Initialize empty Aspiration Criteria.
while (not stoppingCondition())              // Termination Criteria.
    sOld ← neighbor(s)                       // Pick some period branch.
    sNew ← sOld+rand(-k:k)                   // Marginal change to period.
    if(not containsTabuElements(sNew,tabuList)          //Is new period
       or containsAspiration(sNew, aspirCriteria)       // tabu or AC?
        if(fitness(sNew) > fitness(sBest))   // Is new fitness better?
            sBest ← sNew                     // Keep new period.
            tabuList ← featureDifferences(sNew, sBest)        // Update TL
            aspirCriteria ← featureDifferences(sNew,sBest)    //Update AC
            while(size(tabuList) > maxTabuListSize)           // Tabu List FIFO
                ExpireFeatures(tabuList)     // Oldest element discarded
            end
        end
    end
end
return(sBest)                                // Return Best Solution.
```

Fig. 4. TS Pseudocode (Based on: [19])

C. GUIDED LOCAL SEARCH

Guided Local Search (GLS) is the final metaheuristic we applied. As is mentioned by Voudouris and Tsang [10], GLS has the advantage of being easily adaptable to a wide range of combinatorial optimisation problems. Several applications of the GLS can be found in [17], [24].

GLS is a method that is added to a local search algorithm, such as Hill Climbing (HC), in order to manipulate its choices. With the use of GLS, the HC procedure is “guided” to escape local optima. This is accomplished by using penalties on solution features, as well as an augmented fitness function which is carefully modified according to our problem and its objective is to bring the search out of the local optima [25].

As explained in [26], we define for each feature f_i (in our case a period) an indicator[6] function Ii (Equation 6), with which it is indicated whether a feature is present in the current solution s[7] or not:

$$\mathrm{Ii(s)} = \begin{cases} 1, & \text{solution has property i} \\ 0, & \text{otherwise} \end{cases} \quad , s \in S \qquad (6)$$

Furthermore, when a Local Search algorithm returns a local maximum, the GLS algorithm penalizes all features present in that solution which have maximum utility, *util(s,fi)* as illustrated in Equation (7):

$$\mathrm{util(s, fi)} = \mathrm{I_i(s)} \cdot \frac{\mathrm{c_i}}{1 + \mathrm{p_i}} \qquad (7)$$

where, $\mathrm{I_i}$ is the indicator for solution i, $\mathrm{c_i}$ is the cost of features of solution i and $\mathrm{p_i}$ is the penalty of solution i. Finally, GLS uses an augmented fitness function as can be seen in Equation (8), to guide the Local Search out of the local maximum:

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

where, g(s) is the fitness function, M is the number of features, $\mathrm{p_i}$ is the penalty parameter for feature $\mathrm{f_i}$ and λ is the regularization parameter. As it is explained in [27], the advantage of using this augmented function is that the local maxima encountered by local search, when GLS is used, are with respect to Equation (8) and may be different than the local maxima with respect to the original fitness function (Equation 4). Before any penalties are applied, these two are identical but as search progresses, the local maxima with respect to the original fitness may not be local maxima with respect to the augmented function. This allows local search to escape from the local maxima of the original fitness since GLS is altering the local maxima status under the augmented fitness function using the penalty modification mechanism explained in the next paragraph.

GLS was implemented on EDDIE as part of our HC process. It was added on top of a hill climber to be performed when the algorithm gets stuck in a local maximum area. The process starts with the Hill Climbing Algorithm on the periods of the Tree. The GLS process begins when the Hill Climbing returns a local maximum. The first thing that is calculated when the GLS initiates, is the vector I (which consists of binary elements). Additionally, as part of the GLS, the utility function (Equation 7) is calculated and used to penalize the current solution features (periods). More specifically, if the feature maximizes the utility then it is penalized by incrementing the previous penalty of this feature by 1. The new penalties along with the GLS indicators Ii and the fitness are used to calculate the augmented fitness function (Equation 8). This will be used in the HC process from now on instead of

[5] The “tabu” represents areas that we do not wish to be revisited.

[6] Here there is some overlapping terminology between financial forecasting and GLS. The term indicator here has nothing to do with technical analysis indicators that EDDIE uses.

[7] By the term “solution” on EDDIE we mean a Genetic Decision Tree.

the simple fitness function (Equation 4). Last but not least the process will return the best overall solution, with respect to the highest fitness function.

Moreover, the regularization parameter λ, represents the relative importance of penalties and provides a way of controlling the influence of the information on the search process. For more information about the role of λ the reader can refer to [24]. At this point it should be mentioned that λ, is problem dependent thus, it has no standard formula or value. However, in the literature [25] it has been shown that an effective way to calculate it is by using the formula:

$$\lambda = a \cdot \frac{g(s)}{|F(s)|} \tag{9}$$

where $0 \leq a \leq 1$ and $|F(s)|$ is the total number of indicators, in other words the number of features present in each solution. Therefore, we decided to use the same formula for our experiments. Finally, the number of times that the GLS process while continue is pre-specified (termination criteria). Fig. 5 illustrates a GLS Pseudocode tailored to EDDIE.

```
t ← GDT(s)                                  // Probabilistically select a tree.
neighborhood (s)                            //All period branches of "t".
s ← s0                                      // Initial periods.
If (Hill Climbing ← local maximum) // If HC returns local maxima.
   begin GLS                                // Start GLS process.
        while (not stoppingCondition)       // Termination Criteria.
            for (i ← 1 until M)             // For all periods.
                $p_i$ ← 0                   // set penalties to 0.
            end
            for (i ← 1 until M)             // For all periods.
              $util_i$ ← $I_i(s_{k+1})$  $c_i$ / (1 + $p_i$)   // Calculate utility
                                            // function.
            end
            for (each i such that $util_i$ is maximum)
              $p_i$ ← $p_i$ + 1;            // Penalize current periods.
            end
            h ← g - λ * $\Sigma p_i$ * $I_i$    // Calculate augmented fitness
                                            // function h.
            $s_{k+1}$ ←Hill Climbing($s_k$, h)  // Hill Climbing using h.
            end
        end
        s* ← best solution found with respect to fitness function g;
        return s*;
   end GLS

where S: search space, g: fitness function, h: augmented fitness function,
λ:lambda parameter, $I_i$: indicator function for period i, $c_i$: cost for period i, M:
total number of periods present to current solution, $p_i$: penalty for period i.
```

Fig. 5. GLS Pseudocode (Based on: [27]).

IV. EXPERIMENTAL DESIGN

This Section will present the data and parameter values used in our experiments for Simulated Annealing, Tabu Search and Guided Local Search algorithms.

The data used for our experiments can be found in *http://finance.yahoo.com* and in "Datastream". For the purpose of our experiments, 10 datasets were chosen due to their observed good performance with EDDIE from previous experimental works [1], [6], [7]. These datasets are daily closing prices of the following stocks and indices: Aggreko, Athens, Barclays, BAT, Cadbury, Imperial Tobacco, NYSE, Schroders, Sky and Tesco.

As it was discussed in Section II-A, there are several parameters that we need to specify in order to run our experiments. To begin with, we defined the training period to be 1000 days and the testing period to be 300 days. Additionally, as it can be seen from Table I, n is 20 days, r is 4% and the indicators' periods can take any value between 2 and 65.

TABLE I
EDDIE PARAMETERS

EDDIE Parameters	Value
n	20
r	4
period	[2, 65]

Furthermore, we specify the GP parameters (Table II). All of these experimental parameter values are the same as [7], and we decided to keep them unchanged for comparison purposes.

In order to be able to analyze the statistical results, we run the GP 50 times. For processing purposes of the results of those 50 runs, the average and best[8] performance measures are calculated. The results are illustrated in Section V.

At this point, it is also important to explain the individual experimental parameters values that were chosen for each metaheuristic.

To begin with, for the Simulated Annealing algorithm, it was essential to decide upon the values of "temperature" and "kmax", which are the maximum number of iterations. After several tests, we concluded in using the value 0.9 for "temperature" and 8 for "kmax". The basic idea behind the choice of "temperature" value was the fact that we wanted the probability of acceptance to fall gradually until it reaches zero. The value 0.9 was fulfilling this requirement and was proven under the tests to give similar or best results with the other values[9], but in less computational time. That was evidence enough for us to conclude in this value as the most appropriate. Moreover, the value kmax was chosen in a similar way.

TABLE II
GP PARAMETERS

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

Additionally, regarding Tabu Search's *r* and *m* parameters of Tabu List and Aspiration Criteria, we set *r* equal to 4 and *m*

[8] Fitness and RC are maximization problems therefore the best result will be the maximum value. On the contrary RMC and RF are minimization problems so the best will be the minimum value.
[9] The values tested varied from 0.05 to 15.

equal to 2. The *r=4* was chosen in a way to prevent, for short term, the search from revisiting solutions that are very similar to the one already tested. The choice of *m=2* was based on the idea that when a period provides better fitness, then its close neighbors could also lead to further improvements.

Lastly, for the Guided Local Search algorithm we decided that for simplicity, a would be equal to 1 (Equation 9). We do not argue that this is the optimal value, but we have left the investigation of this as a future research. Additionally, as there is no direct associated cost for each feature in EDDIE, we decided to assign the same cost value to all features. Therefore, the cost value c_i, used in the utility function (Equation 7) was set equal to 1 for each feature of i. Finally, we decided to allow for the GLS process to perform 10 iterations. We have left it as future work to see if the results can be further improved with more iteration.

V. RESULTS

This Section will present the results over 50 runs of Simulated Annealing, Tabu Search and Guided Local Search algorithms. First we will present and compare ED8's results with and without each one of the above meta-heuristics. Then, an overall discussion for these results will follow in Section V-D.

As mentioned earlier, the results for each algorithm are divided into average and best. Any significant improvements are denoted with bold fonts, whereas any significant diminutions are underlined. In order to test for any significantly improved average results we run a two sample Kolmogorov Smirnov (K-S) test under the significance level of 5%, for the distributions of the 4 metrics (Fitness, RC, RMC, RF). When testing the results between two data vectors (for instance 50 Fitness runs from original ED8 and ED8 with SA), the null hypothesis H_0 was that the two vectors come from the same continuous distribution, whereas the alternative H_1 is that they don't. Moreover, to account for the fact that we make four comparisons for each dataset, it was essential that we applied Bonferroni correction[10]. Since we compare 4 different metrics between each metaheuristic and ED8, the p-value after the Bonferroni correction at 95% confidence is $p = (1 - 0.95)/4 = 0.0125$, where 4 is the number of metrics used to compare the results of the EDDIE versions. As far as the best results are concerned, it was not possible to apply statistical tests on them, as they are only single values and not distributions. Therefore, for consistency purposes with the Kampouridis et al. [7], we denote a significant improvement/diminution when the difference of the metaheuristic with the original ED8 is above 1%.

A. SIMULATED ANNEALING

Tables III and IV illustrate the average and best results of the application of SA to ED8, using the experimental parameter values mentioned in Section IV-B.

[10] For more information the reader can refer to [28].

As it can be observed from Table III, the average results of the SA are as good as ED8 average results. There is only one case, the RC of Athens, which is worsened at the 5% significance level.

On the other hand, as one can observe from the best results in Table IV, the SA managed to improve the performance metrics for all 10 datasets. Additionally, *the SA improved at least two metrics in 7 out of these 10 datasets*. Overall, the SA algorithm has improved 27 metrics of the datasets and worsened only 7 metrics. We should take into consideration that in several cases the best results were impressively improved, for instance Sky and Barclay's fitness by 10% and 7% respectively. Therefore, we can argue that the addition of SA was proven quite beneficial for ED8's best results.

TABLE III
AVERAGE RESULTS FOR SA

Dataset	***Heuristic***	***Fitness***	***RC***	***RMC***	***RF***
Aggreko	Original	0.2424	0.5919	0.2132	0.2716
	S.A.	0.2175	0.5550	0.4036	0.2656
Athens	Original	0.1541	0.5335	0.3583	0.4486
	S.A.	0.1253	0.5118	0.4008	0.4724
Barclays	Original	0.2651	0.5780	0.4301	0.1291
	S.A.	0.2417	0.5417	0.4755	0.1374
BAT	Original	0.2122	0.5458	0.4318	0.2403
	S.A	0.2134	0.5290	0.4694	0.2362
Cadbury	Original	0.2793	0.6573	0.2170	0.3112
	S.A.	0.2599	0.6309	0.2615	0.3227
Imp Tob	Original	0.1945	0.5343	0.6445	0.205
	S.A.	0.1919	0.5387	0.5702	0.248
NYSE	Original	0.1637	0.5379	0.2494	0.4470
	S.A.	0.1441	0.5213	0.3593	0.4565
Schroders	Original	0.1827	0.5530	0.4142	0.3588
	S.A	0.1798	0.5496	0.4166	0.3610
Sky	Original	0.1248	0.6270	0.7736	0.5802
	S.A.	0.1056	0.5981	0.7443	0.5962
Tesco	Original	0.2602	0.6169	0.3304	0.2563
	S.A.	0.2478	0.6033	0.3313	0.2703

TABLE IV
BEST RESULTS FOR SA

Dataset	***Heuristic***	***Fitness***	***RC***	***RMC***	***RF***
Aggreko	Original	0.3256	0.6933	0.0607	0.1373
	S.A.	**0.3529**	**0.7267**	**0**	0.1855
Athens	Original	0.2579	0.6467	0.0124	0.3571
	S.A.	0.2337	0.6200	**0**	0.3672
Barclays	Original	0.3633	0.7100	0.2449	0.0411
	S.A.	**0.4350**	**0.8167**	**0**	0.0541
BAT	Original	0.3303	0.6667	0.2780	0.1083
	S.A	**0.3690**	**0.7433**	**0**	**0**
Cadbury	Original	0.3685	0.7533	0.1341	0.2131
	S.A.	0.3733	0.7600	**0**	0.2179
Imp Tob	Original	0.2802	0.6367	0.3946	0
	S.A.	**0.2929**	**0.6533**	**0**	0
NYSE	Original	0.2341	0.6067	0.0123	0.3780
	S.A.	0.2283	0.6100	**0**	0.3893
Schroders	Original	0.2369	0.6100	0.2333	0.2456
	S.A	**0.3054**	**0.6800**	**0**	**0.1780**
Sky	Original	0.2066	0.6800	0.5922	0.4222
	S.A.	**0.3059**	**0.6967**	**0**	**0**
Tesco	Original	0.3044	0.6667	0.2255	0.1667
	S.A.	**0.3216**	**0.6900**	**0**	0.2083

B. TABU SEARCH

Table V presents the average results of the TS. As we can observe, there are no significant differences between the two versions of ED8 (with and without TS), with the exceptions of only two metrics (Athens and Sky's RC).

With regards to the best results, Table VI informs us that *TS managed to offer improvements to all of the datasets*. Additionally, it is worth mentioning that TS improved at least two performance measures per dataset, and in some cases like Cadbury and Sky, *it improved all 4 metrics*. In total, the TS algorithm has improved 31 metrics of the datasets and worsened only 4 metrics. We can again notice some impressively improved results, e.g., Sky and Barclay's fitness by 12% and 7% respectively. Therefore, we can once more argue, that the addition of TS was proven beneficial for ED8.

TABLE V
AVERAGE RESULTS FOR TS

Dataset	***Heuristic***	***Fitness***	***RC***	***RMC***	***RF***
Aggreko	Original	0.2424	0.5919	0.2132	0.2716
	T.S.	0.2187	0.5589	0.3884	0.2745
Athens	Original	0.1541	0.5335	0.3583	0.4486
	T.S.	0.1118	0.5028	0.4588	0.4799
Barclays	Original	0.2651	0.5780	0.4301	0.1291
	T.S.	0.2674	0.5832	0.4140	0.1371
BAT	Original	0.2122	0.5458	0.4318	0.2403
	T.S.	0.2138	0.5429	0.4547	0.2216
Cadbury	Original	0.2793	0.6573	0.2170	0.3112
	T.S.	0.2614	0.6335	0.2667	0.3212
Imp Tob	Original	0.1945	0.5343	0.6445	0.2053
	T.S.	0.1976	0.5255	0.6198	0.2028
NYSE	Original	0.1637	0.5379	0.2494	0.4470
	T.S.	0.1551	0.5334	0.3423	0.4433
Schroders	Original	0.1827	0.5530	0.4142	0.3588
	T.S.	0.1917	0.5487	0.4723	0.3419
Sky	Original	0.1248	0.6270	0.7736	0.5802
	T.S.	0.1011	0.6025	0.7551	0.6151
Tesco	Original	0.2602	0.6169	0.3304	0.2563
	T.S.	0.2514	0.6075	0.3274	0.2679

TABLE VI
BEST RESULTS FOR TS

Dataset	***Heuristic***	***Fitness***	***RC***	***RMC***	***RF***
Aggreko	Original	0.3256	0.6933	0.0607	0.1373
	T.S.	**0.3420**	**0.7133**	**0**	0.2015
Athens	Original	0.2579	0.6467	0.0124	0.3571
	T.S.	0.2225	0.6100	**0**	**0.2899**
Barclays	Original	0.3633	0.7100	0.2449	0.0411
	T.S.	**0.4350**	**0.8167**	**0**	0.0392
BAT	Original	0.3303	0.6667	0.2780	0.1083
	T.S.	0.3323	**0.6900**	**0.2287**	**0**
Cadbury	Original	0.3685	0.7533	0.1341	0.2131
	T.S.	**0.3817**	**0.7700**	**0**	**0.1928**
Imp Tob	Original	0.2802	0.6367	0.3946	0
	T.S.	**0.2989**	**0.6567**	**0.0541**	0
NYSE	Original	0.2341	0.6067	0.0123	0.3780
	T.S.	**0.2606**	**0.6500**	0.0309	**0.3125**
Schroders	Original	0.2369	0.6100	0.2333	0.2456
	T.S.	**0.2815**	**0.6567**	**0.0444**	0.2429
Sky	Original	0.2066	0.6800	0.5922	0.4222
	T.S.	**0.3207**	**0.7000**	**0.1165**	**0**
Tesco	Original	0.3044	0.6667	0.2255	0.1667
	T.S.	**0.3253**	**0.6967**	**0.0294**	0.1765

C. GUIDED LOCAL SEARCH

Finally, in Tables VII and VIII the average and best results of the application of GLS on the original ED8 are presented.

Following the same analysis as the previous two Sections, we can see in Table VII the average results of the GLS, which are as good as ED8 average results.

In terms of best results, as we can observe from Table VIII *the GLS has improved all of the 10 datasets*. Additionally, the GLS has improved at least two metrics in all 10 datasets, and all 4 metrics in 7 of them. This thus indicates that the GLS has done extremely well. In total, the GLS algorithm has improved the metrics of the datasets 35 times, and only worsened them 3 times. We can once again observe some impressively improved results, e.g., NYSE and Barclay's RC by 12% and 10% respectively. Therefore, we can suggest, that the addition of GLS was proven really valuable for ED8.

TABLE VII
AVERAGE RESULTS FOR GLS

Dataset	***Heuristic***	***Fitness***	***RC***	***RMC***	***RF***
Aggreko	Original	0.2424	0.5919	0.2132	0.2716
	GLS	0.2339	0.5762	0.3694	0.2497
Athens	Original	0.1541	0.5335	0.3583	0.4486
	GLS	0.1491	0.5250	0.3117	0.4564
Barclays	Original	0.2651	0.5780	0.4301	0.1291
	GLS	0.2894	0.6091	0.3824	0.1260
BAT	Original	0.2122	0.5458	0.4318	0.2403
	GLS	0.2254	0.5416	0.4390	0.2337
Cadbury	Original	0.2793	0.6573	0.2170	0.3112
	GLS	0.2698	0.6353	0.2553	0.3158
Imp Tob	Original	0.1945	0.5343	0.6445	0.2053
	GLS	0.1969	0.5349	0.6139	0.2087
NYSE	Original	0.1637	0.5379	0.2494	0.4470
	GLS	0.1569	0.5339	0.2825	0.4504
Schroders	Original	0.1827	0.5530	0.4142	0.3588
	GLS	0.1872	0.5495	0.4319	0.3514
Sky	Original	0.1248	0.6270	0.7736	0.5802
	GLS	0.1175	0.6187	0.7608	0.5922
Tesco	Original	0.2602	0.6169	0.3304	0.2563
	GLS	0.2533	0.5967	0.3615	0.2777

TABLE VIII
BEST RESULTS FOR GLS

Dataset	***Heuristic***	***Fitness***	***RC***	***RMC***	***RF***
Aggreko	Original	0.3256	0.6933	0.0607	0.1373
	GLS	**0.3420**	**0.7133**	**0**	**0**
Athens	Original	0.2579	0.6467	0.0124	0.3571
	GLS	0.2389	0.6267	**0**	**0.3411**
Barclays	Original	0.3633	0.7100	0.2449	0.0411
	GLS	**0.4350**	**0.8167**	**0**	**0.0260**
BAT	Original	0.3303	0.6667	0.2780	0.1083
	GLS	**0.3690**	**0.7433**	**0**	**0**
Cadbury	Original	0.3685	0.7533	0.1341	0.2131
	GLS	**0.4153**	**0.8067**	**0**	**0.1897**
Imp Tob	Original	0.2802	0.6367	0.3946	0
	GLS	**0.3197**	**0.6767**	**0**	0
NYSE	Original	0.2341	0.6067	0.0123	0.3780
	GLS	**0.2464**	**0.6200**	**0**	**0.3540**
Schroders	Original	0.2369	0.6100	0.2333	0.2456
	GLS	**0.2909**	**0.6700**	**0**	**0**
Sky	Original	0.2066	0.6800	0.5922	0.4222
	GLS	**0.2214**	0.6733	**0**	0.4706
Tesco	Original	0.3044	0.6667	0.2255	0.1667
	GLS	**0.3619**	**0.7400**	**0**	**0.1467**

D. DISCUSSION

As it was observed in the previous Sections, all three metaheuristics maintained the average ED8 results at the same level, while significantly improving all the datasets in terms of the best results. First of all, we should note that the fact that the average results were not improved by the metaheuristics is not alarming. Due to the large search space that ED8 is dealing with, it is not easy to be consistent and always return significantly improved solutions.

Nevertheless, the heuristics have indeed introduced significant improvements, as it can be seen from the best results. *This is an extremely important achievement* due to the fact that an investor in the real world would be running

EDDIE multiple times and then use *the best resulted GDT* for his investments; thus, due to the fact that the metaheuristics have improved ED8's best results, this investor would see an increase to his profit margin.

Furthermore, Table IX summarizes the improvements and diminutions of the three algorithms, in terms of best results. As we can observe, *GLS was proven to be the most beneficial* in terms of improvements and diminutions, as it managed to improve 35 metrics, while it only worsened 3. SA and TS also offered a significant difference between the number of improvements and diminutions. This achievement is really impressive and indicates that with the addition of metaheuristics, ED8 best GDT's would provide more accurate information to the investor.

To sum up, from the above discussion we are able to support that the application of metaheuristics to EDDIE can lead to significantly improved performance, thus being a really valuable addition to EDDIE.

TABLE IX
SUMMARY BEST RESULTS

Metaheuristic	Improvements	Diminutions
SA	27	7
TS	31	4
GLS	35	3

VI. CONCLUSION

This paper presented work on the application of the three meta-heuristics, namely Simulated Annealing, Tabu Search and Guided Local Search, to a Genetic Programming financial forecasting algorithm called EDDIE 8. The novelty of ED8 is that it allows the GP to search in the space of technical indicators for solutions, instead of using pre-specified ones, as it usually happens in the literature. However, a consequence of this is that ED8's search area is quite large, and sometimes solutions can be missed due to ineffective search.

In order to address this issue, we applied the three above-mentioned metaheuristics to the period nodes of ED8's trees. Results showed that the algorithm's performance improves significantly, thus *making the combination of Genetic Programming and meta-heuristics an effective financial forecasting approach*.

More specifically, we found that all three metaheuristics were beneficial and important. We also particularly emphasized the significance of the improved best results, due to the fact that an investor, who would use the best tree of those experiments, *could experience an exceptional boost to his profit.*

The fact that the metaheuristics tested in this paper have improved EDDIE's performance is very encouraging. Our next goal is to combine these meta-heuristics into hyper-heuristics frameworks, as it has happened before in [7]. This will have the additional benefit that such frameworks can combine the strengths of each meta-heuristic, and thus improve the EDDIE's performance results even further.

REFERENCES

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

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

[3] R. Poli, W.B. Langdon, N.F. McPhee, "*A Field Guide to Genetic Programming*", Lulu Enterprises, UK , 2008.

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

[5] J. W. Backus, "*The Syntax and Semantics of the Proposed International Algebraic Language of the Zurich",* International Conference on Information Processing, UNESCO, pp.125-132, 1959.

[6] M. Kampouridis, E. Tsang, "*Investment Opportunities Forecasting: Extending the Grammar of a GP-based tool*", International Journal of Computational Intelligence Systems, 5 (3), pp. 530-541, 2012.

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

[8] S. Kirkpatrick, C.D. Gelatt, M.P. Vecchi, "*Optimization by Simulated Annealing*", *Science* 220 (4598): 671–680, 1983.

[9] F. Glover, et al. "*A user's guide to tabu search*", University of Colorado Annals of Operations Research 41 3-28, 1993.

[10] C. Voudouris, E. Tsang, "Guided Local Search", Technical Report CSM-247, University of Essex, 1995.

[11] H. Xie, M. Zhang, and P. Andreae, "*Genetic programming for new Zealand CPI inflation prediction",* Proceedings of the IEEE Conference on Evolutionary Computation, Singapore, pp. 2538–2545, 2007.

[12] M. Bernal-Urbina, A. Flores-Mendez, "*Time series forecasting through polynomial artificial neural networks and genetic programming*", Proceedings of the IEEE Conference on Evolutionary Computation, Hong Kong, pp. 3324–3329, 2008.

[13] R. Kohavi, F. Provost, "*Glossary of Terms Machine Learning*", Vol 30, 1998.

[14] J. Li, "*FGP: A genetic programming based financial forecasting tool*", Ph.D. dissertation, University of Essex, 2001.

[15] M. Laguna, J. Barnes, F. Glover, "*Intelligent scheduling with tabu search: An application to jobs with linear delay penalties and sequence-dependent setup costs and times",* Journal of Applied Intelligence Vol. 3 No. 2 pp. 159-172, 1993.

[16] M. Lemes, C. Zacharias, A. Dal Pino, *"Generalized simulated annealing Application to silicon clusters",* Physical Review B (Condensed Matter), Volume 56, Issue 15, pp.9279-9281, 1997.

[17] A. Alsheddy, "*Empowerment scheduling: a multi-objective optimization approach using Guided Local Search*", PhD Thesis, Uni. of Essex, 2011.

[18] L. Davis, "*Genetic algorithms and simulated annealing*", Pitman, 1987.

[19] J. Brownlee, "*Clever Algorithms: Nature-Inspired Programming Recipes*", PhD First Edition, Lulu Enterprises, 2011.

[20] F. Glover, "*Future paths for Integer Programming and Links to Artificial Intelligence*", Computers & Operations Research, 5:533-549, 1986.

[21] F. Glover and C. McMillan, "*The general employee scheduling problem: an integration of MS and AI*", Computers and Operations Research Vol.13, No. 5, pp. 563-573, 1986.

[22] M. Gendreau, "*Introduction to tabu search*", in handbook of Metaheuristics, pages 37–54, Springer, 2003.

[23] F. Glover, M. Laguna, "*Tabu Search*", Kluwer Academic Publishers, 1998.

[24] C. Voudouris, "*Guided local search for combinatorial optimization problems*", PhD Thesis, University of Essex, 1997.

[25] C. Voudouris and E. Tsang, "*7: Guided Local Search*", in Handbook of Metaheuristics, pp.185-218, Springer, 2003.

[26] P. Mills, "*Extensions to Guided Local Search*", PhD Thesis, Department of Computer Science, University of Essex, 2002.

[27] C. Voudouris and E.Tsang, "*Guided Local Search and its application to the Travelling Salesman problem*", European Journal of Operational Research 113 pp.469-499, 1999.

[28] H. Abdi, *"Bonferroni and Šidák corrections for multiple comparisons"*, In N.J. Salkind Encyclopedia of Measurement and Statistics, Thousand Oaks, CA: Sage, 2007.