Martín, C., Quintana, D., Isasi, P. (2019). Grammatical Evolution-based ensembles for algorithmic trading. *Applied Soft Computing*, 84, 105713.

DOI: 10.1016/j.asoc.2019.105713

Grammatical Evolution-based Ensembles for Algorithmic Trading

Carlos Martín[a], David Quintana[a,*], Pedro Isasi[a]

[a] *Universidad Carlos III de Madrid, Department of Computer Science, Avda. Universidad 30, Leganés, Madrid, Spain*

Abstract

The literature on trading algorithms based on Grammatical Evolution commonly presents solutions that rely on static approaches. Given the prevalence of structural change in financial time series, that implies that the rules might have to be updated at predefined time intervals. We introduce an alternative solution based on an ensemble of models which are trained using a sliding window. The structure of the ensemble combines the flexibility required to adapt to structural changes with the need to control for the excessive transaction costs associated with over-trading. The performance of the algorithm is benchmarked against five different comparable strategies that include the traditional static approach, the generation of trading rules that are used for single time period and are subsequently discarded, and three alternatives based on ensembles with different voting schemes. The experimental results, based on market data, show that the suggested approach offers very competitive results against comparable solutions and highlight the importance of containing transaction costs.

Keywords: grammatical evolution, trading, ensembles, finance

1. Introduction

Stock market is highly dynamic and subject to structural changes. Trading rules might be very profitable during specific periods of time and progressively loose their effectiveness as market dynamics evolve over time. This

*Corresponding author

Email addresses: carlos.m.fernandez@alumnos.uc3m.es (Carlos Martín), david.quintana@uc3m.es (David Quintana), isasi@ia.uc3m.es (Pedro Isasi)

requires developing approaches capable of detecting and adapting to these structural changes.

Since Allen and Karjalainen [1] published their seminal piece on evolution of trading rules using Genetic Programming (GP), many authors have made related contributions either based on the same technique, or Grammatical Evolution (GE). Among them, [2, 3, 4, 5, 6, 7, 8, 9].

Most of these contributions generate investment rules based on a combination of raw market data and technical indicators and, unlike related approaches that use genetic algorithms or evolution strategies to optimize predefined rules, these have the advantage of creating flexible structures automatically. Other algorithms within the evolutionary computation framework, like the ones discussed in [10, 11], would be readily applicable to related financial problems like mean-variance portfolio optimization, but would require significant adaptations and domain expertise to evolve the functional trees that GP and GE generate.

A common limitation is that it is often the case that the methods are static, and do not take into account structural changes. Given that this phenomenon is very prevalent in financial time series, decision rules are commonly derived from market environments that do not hold for long periods.

The problem of adjusting to structural changes is that we must choose between two opposite extremes: keeping the same model over time, or updating it constantly. Even though the second might seem, at least in principle, more appropriate, there is a possibility that the constant change in the model will have undesirable consequences due to transaction costs. The evolutionary process of GP/GE considers commissions within the fitness function, and that makes it select rules that generate a limited number of signals. However, it is possible that constant model updates might interfere with that endogenous control mechanism of the number of purchase and sale orders.

This paper introduces a dynamic trading system based on the use of ensembles and GE. The approach combines the possibility of changing the model, as a reaction to changes in the price generation mechanism, with an inertia component that mitigates the consequences of overtrading.

An ensemble in this context can be compared to a collegiate decision committee in which the votes of several judges are combined to arrive at a final decision [12]. The idea behind this method is to take advantage of the good local behavior of each of the judges, in order to increase the accuracy and reliability in the environment of a global scenario [13]. In Statistics and Automatic Learning, ensemble methods use multiple learning algorithms to obtain a better predictive performance than that which could be obtained from any of the constituent learning algorithms separately [14, 15, 16]. Unlike a statistical set, which is generally infinite, an automatic learning set consists of only a finite set of alternative models but typically allows a much more flexible structure to exist by combining those alternatives.

It is worth noting that the solution that we introduce integrates the output of several trading rules to generate a combined recommendation that makes financial sense in dynamic environments. This differs from other algorithmic solutions that rely on ensembles of models to tackle more standard classification or regression tasks, such as [17, 18]. The aim of trading algorithms is obtaining profitable rules, and one might consider that this task implicitly requires solving two problems. The first one would involve predicting market movements based on past information, for instance, it might predict whether the market is expected to go up, down, or remain stable, while the second one would be exploiting the previous information to obtain investment recommendations. Algorithms like the ones that we just mentioned might well excel at the first task, but lack the second layer.

Joint solutions are widely used in Artificial Intelligence, especially in neural networks (Hansen and Salamon [19]; Perrone and Cooper [20]; Opitz and Shavlik [21]). In these cases, several classifiers, usually neural networks with different topologies and/or parameters, are used to classify the same input pattern and their votes are combined using a specific rule such as majority, arithmetic mean, weighted average, etc. However, there are other works related to GP, such as Grosan et al. [22], that use the technique of ensembles in the context of obtaining investment models in financial markets. In this work, as we will discuss, the decisions committees are formed by different trading rules obtained using GE as the basic optimization algorithm and a sliding window.

The structure of the rest of the document is as follows: the next section describes the main references on GE for algorithmic trading and Evolutionary Computation (EC) based on adaptive approaches. Then, section 3 describes the proposed approach. That will be followed by section 4, focused on the experimental analysis. Finally, section 5 will be devoted to summary and conclusions.

2. Previous Work

The academic literature on GE for algorithmic trading is not as ample as the one based on GP. However, there are a number of relevant contributions that deserve to be mentioned.

One of the first works, in which evolutionary grammars are used to discover trading rules based on technical indicators, is that of Brabazon and O'Neill [23]. These authors explored the possibility of using this technique to generate investment rules for the money market. In their study, they combined a small set of technical indicators with a metric to penalize commercial risk. The experimental work relies on a relatively low amount of currency data from the London market, from 10/23/92 to 10/13/97 and their results outperformed the benchmark on five out of the six test sets.

Dempsey et al. [24] used a GE-based methodology to discover technical investment rules targeting the S&P 500 and Nikkei 225 indices. The authors addressed two approaches, one with a single set of population rules that was adapted throughout time, and another by which a new population was created in each generation step. They concluded that there was a profitable return for the chosen investment periods, with evident advantages in the case of the adaptive population rules. However, they also found that there very limited opportunities to beat the S&P 500 index. Nevertheless, in the Nikkei 225 index, GE generated investment returns with an average improvement of 74% over the index.

In their article, Contreras et al. [25] presented a system based on GE seeking to obtain profitable trading rules. They validate its performance against historical returns of a group of companies in the Spanish market using data from 2012. In addition to that, they compared the results of the GE system with a previous approach [26] based on Genetic Algorithms (GA). The trading system implemented with the GE obtained gains around 14%, while the GA-based alternative generated losses around 20%. Further analysis, with an expanded set of nine selected Spanish companies, showed that the overall benefit of the investment was higher than the B&H strategy.

More recently, Schmidbauer et al. [27] developed an evolutionary computation tool based on a grammar-guided GP framework. The system selects negotiation rules which curb the data-snooping bias of performance evaluation. The core of its approach resides in the concept of a priori robustness, and the objective is identifying rules that work well with both the original price series and other similar ones. For the evaluation, they chose a multiobjective fitness criterion which involved the original as well as modified time series. They used intraday data of FOREX trading of Euro/USD exchange from the first semester of 2011 to test the method, and their findings suggest that their a-priori robustness criterion provides better results and also prevents overfitting. Their experimental results show that their method improved performance, but not to the point of deriving profitable strategies.

If we consider adaptive solutions, even if we open the possibilities from GE to EC in general, the number of relevant approaches is still quite limited. Among the references more closely related to the solution suggested in this study we could highlight the following:

Grosan et al. (2006) [22] used two genetic programming variations, Multi Expression Programming (MEP) and Linear Genetic Programming (LGP), to build a prediction ensemble of two different stock indices: the Nasdaq-100 index and the stock S&P CNX NIFTY. They benchmark their solution against four alternatives: an artificial neural network trained using the Levenberg-Marquardt algorithm; the neuro-diffuse model of Takagi-Sugeno; and the genetic programming algorithms MEP and LGP. To evolve the GP-based solutions, they applied a multiobjective evolutionary optimization algorithm (MOEA) known as "*Non-dominated Sorting Genetic Algorithm II*" (NSGAII) [28]. Their empirical results revealed that the joint approach constituted a very promising method for stock market forecasting. The authors concluded that the results obtained by the ensemble were better than those achieved by each of the GP variations separately.

Wilson et al. (2011) [29] used a LGP system that applied a generated trading model to multiple intraday time frames. They established two decision systems to determine the final trading action coming from the trading model applied to all time frames. One of them, based on majority vote, generates purchase or sell recommendations where the signal accounts for half (or more) of all the signals for each time scale. The other relies on temporal proximity to the the purchase or sale recommendation. They found that the temporal proximity decision mechanism was more restrictive and traded slightly less often than the majority-based decision counterpart and it was found not to work as well. Majority decision involving more time frames was more conservative and less reactive to changes in price trends. Increasing the number of voters through more time frames was found to be better than shorter time frame combinations because it encouraged remaining in the market and reduced the number of transactions.

Shangkun Deng et al. (2013) [30] used a GA to generate currency trading rules based on the Relative Strength Index (RSI), a technical indicator. They added as an input to the GA three time frames from which to extract features. The target trading currency pair used in their experimental analysis was EUR/USD, and the trading time horizon was one hour. They fed the system with a combined signal from a relatively longer time frame of two hours and a shorter time frame of 30 minutes, besides the target time frame of one hour. The data set covered the period from January 3 2011 to December 30 2011, with a total of 6,178 observations of hourly data. Based on their experimental results, they concluded that the combined signal from multiple time frames improved performance.

Following a different strategy, that combined an ensemble of multiple rules based on technical indicators, Jayanthy et al. (2014) [31] suggested a new approach to obtain an investments strategy for stock markets. Their experimental results showed good performance on two major Indian stock indices.

Machado et al. (2015) [32], introduced a GP-based solution that relies on technical indicators to obtain trading rules that exploit trend-following. In order to improve the robustness of the resulting solutions, they combined three time frames with different weights to form the final market position of the system for each day: Long-Term (LT), Medium- Term (MT) and Short-Term (ST). They run tests using a sliding window of 6 years and an out-of-sample window of 6 months on the American stock index S&P 500. The experimental results show annualized rates of return in excess of 10% for some configurations.

Finally, Pimenta et al. (2017) [13] presented a system based on GP that implements an outlier filtering procedure together with a feature selection method and decision ensembles to obtain efficient trading rules using technical indicators. Their solution follows the process that follows: the first phase comprises filtering outliers; the second is aimed at selecting the appropriate technical indicators to reduce the solution space and the third one involves evolving investment rules using GP. Finally, the obtained rules are then used to build an ensemble whose decision committee consists of the individuals that are part of the final approximation of the Pareto set delivered by the algorithm. The authors showed in their conclusions that their system was able to obtain financial returns considerably above the mere share price variations. They also highlighted that their approach was not tailored to the specific characteristics of any stock market in particular, and that it could easily be applied to other types of financial time series, with small adjustment to a few parameters.

3. Proposed Approach

This section starts with an introduction to the standard approach to evolve trading rules with flexible representation using GE. As it was mentioned before, the standard method suffers from some limitations in a domain where structural change is prevalent. For this reason, we then introduce an adaptive ensemble approach designed to overcome them and improve performance.

3.1. Obtaining Trading Rules with Grammatical Evolution

Grammatical evolution is an algorithm within the field of EC that was introduced by Ryan et al. [33]. The technique is closely related to GP [34] but, unlike the latter, individuals are encoded as vectors of integers that represent production rules from a context-free grammar. Even though there are some other differences, the aim of the technique and the core elements of the algorithmic loop (mutation, crossover...) are very similar.

While GE genotypes consist of vectors of integers, phenotypes, usually take the form of tree structures. This dual representation is managed by a grammar, often specified by the user in Backus-Naur form (BNF), that describes the basic components, terminals and non-terminals, and the syntax. The latter element is key, as it effectively restricts the search space. The main advantages of this are two: a more efficient type control, and the possibility to incorporate domain knowledge.

The initialization of individuals requires arrays of random integers in the range from 0 to the maximum number of productions of the grammar rule with the largest number of them minus one. Then, the elements are processed one by one according to the grammar, and the appropriate rules and required arguments are identified applying the modulus (remainder operation). The mapping process gradually constructs the tree (or s-string) in preorder, replacing non-terminal symbols with the right hand of the selected grammar productions.

It is clear that the choice of the appropriate set of terminal and non-terminal functions is a key element of the process. In that regard, we will rely on previous studies and use the set suggested by Lohpetch and Corne [5]. The difference, however, is that we will use daily stock returns instead of monthly ones. Bearing that in mind, the terminal elements will be index prices and technical indicators. Non-terminal ones would be the basic relational operators ($>$ and $<$) and the main logical ones (*And*, *Or* and *Not*).

The set of technical indicators considered in the study includes the following:

- Index prices. Opening, closing, high and low daily index prices. (*Open*, *Close*, *Max* and *Min*)

- Simple 2, 3, 5 and 10-month moving averages. These trend-following/ lagging indicators smooth out price volatility and are computed as the simple average of closing prices over a predefined number of periods. (*M2*, *M3*, *M5* and *M10*)

- Rate of Change Indicator (3-month and 12-month). This momentum technical indicator tracks percentage differences between current prices and past prices n periods ago. (*Roc3* and *Roc12*)

- Price Resistance Indicators. These price points define virtual upper and lower bounds that are unlikely to be broken in the short term. We consider the last two 3-Month moving average minima and maxima. (*Mx1*, *Mx2*, *Min1* and *Min2*).

- Trend Line Indicators. These upper and lower resistance lines based on the slope of the last two maxima and minima, respectively, express the direction and speed of price changes. (*UR* and *LR*).

The grammar used to represent the investment rules generates logical expressions. In this regard, we follow the approach described in [5], which is reported in BNF-form in Table 1. The evaluation of any rule would be either "0" or "1", depending on market conditions. We will consider that "1" represents a recommendation to be invested in the market, while "0" would be interpreted as a suggestion to be in cash.

The rule generation process and its interpretation can be illustrated with an example. Given the grammar reported in table 1 and the pseudo-random sequence of integers in the range from 0 to 255 that follows:

015 161 179 174 202 215 078 089 216 129 003 029

Table 1: Trading Rules Grammar

Nº	Modulus	Grammar Rule
1	1	<Rule> ::= <bool>
2	5	<Bool> ::= (**And** <bool> <bool>)| (**Or** <bool> <bool>)
		<Bool> ::= (**Not** <bool>)
		<Bool> ::= (**>** <exp> <exp>) | (**<** <exp><exp>)
3	16	<Exp> ::= (**Open**)|(**Close**) | (**Max**) | (**Min**)
		<Exp> ::= (**M2**) | (**M3**) | (**M5**) | (**M10**)
		<Exp> ::= (**Roc3**)| (**Roc12**)
		<Exp> ::= (**Mx1**) | (**Mx2**) | (**Min1**) | (**Min2**)
		<Exp> ::= (**UR**) | (**LR**)

The process required to build the associated decision tree would start with the first element of the sequence, number 15, and the computation of its (mod 1), associated with the initial symbol of the grammar. The result of this operation would be the selection of the rule <bool>. As it is not a terminal rule, we would consider the second element of the sequence, 161, and compute (mod 5) to identify the appropriate function. Given that the result is 1, the first left production of symbol **bool** (**Or** <bool><bool>) would serve the root of the functional tree. Since **Or** is a non-terminal function with two input arguments, the process would require the identification of its two children. The first of them would be based on the next element, 179, which in (mod 5) corresponds to 4 and, therefore, to the rule (**>** <exp><exp>). Given that $>$ also requires two arguments, and the fact that the element that follows it in the sequence is 174, its first child node would correspond to (**UR**), a terminal element. The process of growing the tree in preorder would continue until either all the rules are expanded, or all the elements in the vector are used. If that were the case, either a wrapping mechanism would be used, or the individual would be discarded.

The result of the process would be a tree representation like the one illustrated in figure 1, which would correspond to an investment rule whose S-expression would be:

(Or (<UR MX1) (And (>Roc12 Roc3) (<Min M2)))

The assessment of trading rules is based on a fitness function that, in this case, is defined as net investment return. That is, the sum of all transaction profits/losses adjusted for transaction costs $\Delta r = r$. Following previous

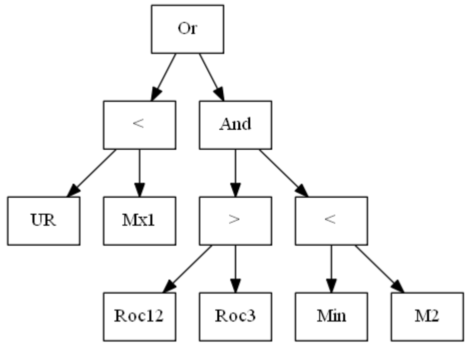


Figure 1: Trading Rule with Tree Representation.

studies like [2] and [35], we will use continuous compound yield, which is formally defined as:

$$r = \sum_{t=1}^{T} r_t \cdot I_b(t) + \sum_{t=1}^{T} r_f(t) \cdot I_s(t) + n \cdot ln\Big(\frac{1-c}{1+c}\Big) \qquad (1)$$

where:

- $r_t = ln(P_t) - ln(P_{t-1})$ is the continuous composite yield; P_t is the index at time t and r_t represents the market return accrued during the days that the investor is in the market.

- $I_b(t)$ variable that takes the value "1" in case that the recommendation is being in the market, and "0" otherwise. Conversely, $I_s(t)$ represents the opposite.

- r_f represents the risk-free return accrued by cash positions, and it is formally defined as:

$$r_{f_{(t)}} = ln\frac{(1 + r_f, monthly)}{\delta} \qquad (2)$$

Where $(1+r_f, monthly)$ denotes the monthly interest rate on the money market, and δ the number of trading days.

The third element of the expression represents transaction costs. There, n represents the number of transactions (open positions are assumed to be closed on the last trading day of the period), and c is the one-way transaction cost expressed as a price fraction (in this study it is assumed to be 0.25%).

Like other techniques within EC, GE relies on a number of operators to drive the evolution process. In this case, one-point crossover, tournament selection and uniform mutation. On the one hand, these are generally implemented very efficiently, as chromosomes are often encoded as variable-length binary strings that represent a series of integers. On the other hand, they often produce a large proportion of individuals that end up being invalid.

The problem of invalid individuals can be addressed in a number of ways, like wrapping the chromosome and interpreting it in a circular fashion in mutation, or applying crossover on the codon limit to avoid arbitrary positions [33, 36]. The implementation used in the analysis uses two standard GE repair strategies: duplication and truncation.

The first one, applicable in case that the original string of integers is too short, starts with the selection of a sequence of the missing n integers from within the same individual, A. This sequence, delimited by two indices, i and j, n positions apart, is then appended at the end of the vector. Therefore, the resulting individual will be the result of the concatenation operation $A_{1..n}||A_{i..j}$. In this regard, it should be noted that the selection of the lower end, i, is random.

In relation to truncation, this strategy determines the number of integers from the chromosome vector required by the initialisation of the tree, and eliminates all the rest. In case that only r elements were required to represent individual A, $A_{1..n}$ would be used, and $A_{r..n}$ would be truncated.

Another challenge posed by the representation system is bloating. Individuals tend to grow very rapidly with generations, and this has a negative impact on both interpretability and performance due to overfitting. In order to control this problem, the setup suggested in this study uses non-parametric parsimony pressure. This mechanism, implemented in the selection operator, punishes complexity breaking ties in terms of fitness selecting the simplest rule in competition.

3.2. Adaptive Ensemble Approach

In traditional systems, a single investment rule is generated and subsequently used throughout the whole test period. We label this approach *Static* strategy. On the opposite end of the spectrum we would have the alternative of generating trading rules that would only be used for a single time period. This could be implemented by means of a sliding window that would move one period at a time. That way, we would obtain as many trading rules as time periods. If we decided to consider for each time step only the recommendation of the rule based on the most recent information, we would obtain an approach that we will refer to as *Naif*.

This *Naif* strategy requires training multiple rules, but it would not not be considered a traditional ensemble. However, once that we have several rules available for every time period, we could start setting up ensembles that would generate recommendations based on voting mechanisms. For instance, we could define a system that would make investment recommendations according to simple majority voting based on the output of the s most recent trading rules. In case more that 50% of the s rules supported being invested in the market, the system would recommend it.

The approach that we suggest involves using an ensemble of trading rules obtained using the mentioned sliding window. Given a fixed window size, w, a trading rule is evolved using GE and the rule is used in subsequent periods to generate investment recommendations. If we move the sliding window one time-step forward, we can generate a new rule based on a new training sample that overlaps with the previous one for all the elements but the last one, the most recent (the first element of the initial window gets dropped). Once again, that new rule will be used to generate predictions for the future. If we only consider the most recent one to cast a recommendation for the next period and we repeat the process over time, we obtain the mentioned *Naif* strategy.

If we analyze this procedure, illustrated in Figure 2, we see that for any given moment of time t we can create an ensemble combining the s rules generated using the s most recent training periods. This means that the first model of the set would have been obtained using the portion of the dataset between the time instants $t-s-w+1$ and $t-s$, while the last model would be based on the time period that covers $t-w$ and $t-1$. For instance, given that in the figure the number of s voting rules is 5 and w is 10, the voting rules for $t = 22$ are those in rows 8 to 12.

The investment recommendation of the aforementioned ensemble for time t would require combining the individual recommendations of s for that point in time. The simplest solution to achieve this would be casting a majority vote among the s recommendations. In case most models recommend "1" (buy), the ensemble of models would advise to be in the market, otherwise, if the dominant recommendation were "0" (sell), the ensemble would suggest holding a cash position, or being out of the market. A dynamic version of

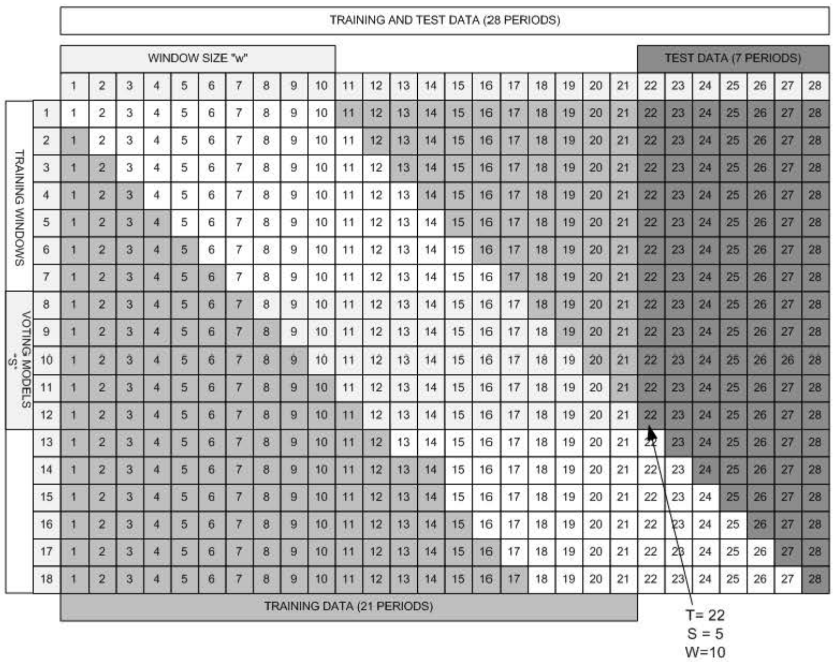


Figure 2: In medium gray color the training set, in dark grey the test set, in white the training windows used, and in light gray the windows used in the set for time t.

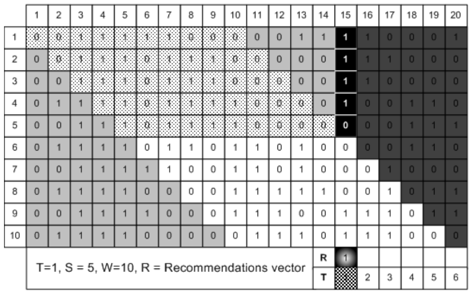
this idea would require a different ensemble per time period. The difference between any ensemble and its predecessor would be the addition of a new rule based on more recent data, and the elimination of the oldest one to keep the s constant and updated. We might label this trading system *Majority*. It is worth mentioning that the described *Majority* approach gives the same weight to all the recommendations, but nothing would prevent us from using a weighted voting scheme instead.

While both *Naif* and *Majority* offer the possibility of being adaptive, they have a potential flaw in common. The individual GE-based investment rules are optimized for relatively long periods of time. Given that the fitness function includes trading fees, one of the drivers of the evolution process is controlling the number of transactions. Once we combine several strategies, this implicit control mechanism might not be as effective and we bear the risk of trading more that necessary. This might result in profitability being severely eroded by commissions.

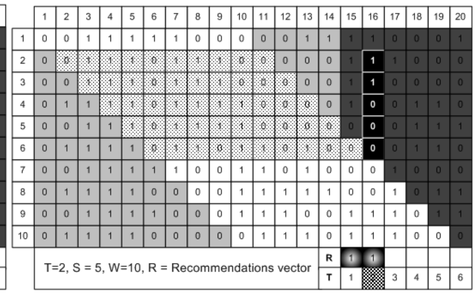
The solution that we suggest is providing the described *Majority* ensemble with some degree of inertia that biases the recommendations towards keeping the same position. Initially, the first recommendation of the ensemble would be obtained by a simple majority. From there on, only if there is a strong consensus among the rules of the ensemble regarding the need to switch current position and, therefore, get in or get out of the market, the ensemble would recommend it. The strength of this consensus would be controlled with a parameter that we might name inertia factor, i. Only if the number of models recommending a change in current position is greater or equal than the mentioned threshold, the system will do it. If that were not the case, the recommendation would be keeping current position.

The process is illustrated in Figure 3. There, we present an example where we use an ensemble of $s = 5$ models optimized on training samples defined by a sliding window of size $w = 5$ periods and an inertia factor $i = 4/5$. The recommendation for T1, described in panel 3a, is the result of the simple majority. Given that the first four rules (1-4) recommend being in the market and only the fifth one recommend a cash position, the recommendation of the ensemble would be being fully invested. Regarding T2, in panel 3b, the five most recent trading rules are 2-6, and the majority of them recommend being out of the market. However, a support of 3/5 does not meet the 4/5 threshold criterion and, therefore, is not enough to change current position. For that reason, the ensemble would recommend staying in the market. Conversely, at T3, in 3c, 4/5 of the components of the ensemble (trading rules 3-7), recommend switching from "1" to "0". As the support is greater or equal to the inertia factor of 4/5, the ensemble would recommend leaving the market. The dynamics in T4-T6, pictured in 3d-3f are similar. Only if the support of the five rules within each ensemble for a change in current position meets the required threshold, the system recommends it.

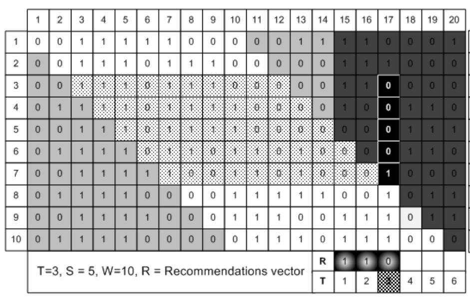
Regarding computational cost, the key element of the core algorithm used to obtain the rules is the evaluation of the fitness function, which is significantly more expensive than the application of the genetic operators. As a result, the complexity of training a single trading rule using GE could characterized as of linear order $O(g \times n)$, where g is the number of generations and n is the size of the population. Given that only the *Static* approach trains a single rule, this cost should be multiplied by the number of trading rules to be generated. Both the *Naif* and the *Inertia* system generate a new rule per time period and, therefore, the complexity would be $O(g \times n \times t)$, where t is the number of time periods for which recommendations are issued. Since the approaches that rely on ensembles and sliding windows, like *Inertia*, *Majority*, or those based on similar weighted voting schemes, require training s rules to



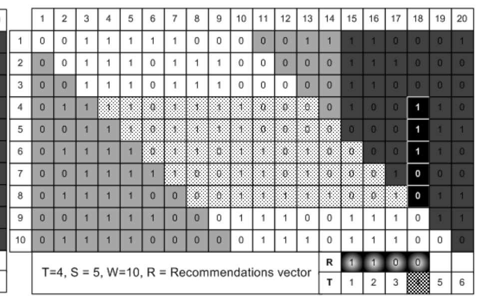
(a) T1. Majority vote: 1. Recommendation:1



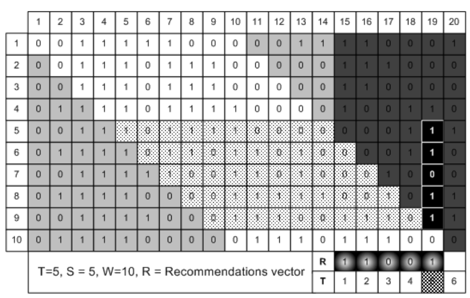
(b) T2. Majority: 0. Support: 3/5. Current: 1. Recommendation: 1



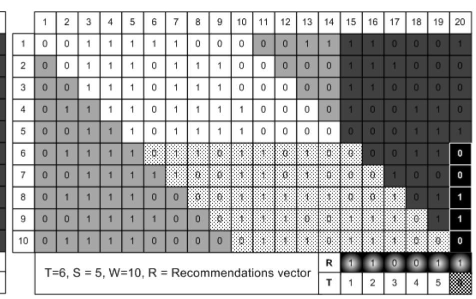
(c) T3. Majority: 0. Support: 4/5. Current: 1. Recommendation: 0



(d) T4. Majority: 1. Support: 3/5. Current: 0. Recommendation: 0



(e) T5. Majority: 1. Support: 4/5. Current: 0. Recommendation: 1



(f) T6. Majority: 0. Support: 3/5. Current: 1. Recommendation: 1

Figure 3: Illustration of ensemble behavior for periods **T** 1 to 6. Ensemble **S** of 5 models with a threshold of 4/5. Training periods in darker color. Test periods in lighter color. Investment recommendations: "1" stay in the market, "0" stay in cash. Partial recommendations of the elements of the ensemble in black rectangles. Ensemble recommendation **R** below in bold.

generate the first recommendation, the previous estimate should be updated to $O(g \times n \times (t + s - 1))$. The cost of the weighting scheme itself could be deemed irrelevant. As reference, the average time required to evolve a single rule using the parametrization used in the experimental analysis on an Intel Core I7 2630QM with 16 Gbytes DDR3 using 8 threads was 1.196 seconds. Based on that, *Static* approach, it would be easy to extrapolate. The *Naif* strategy would require roughly to multiply that figure by the number of periods to be predicted and so on.

4. Experimental analysis

In this section we describe the experimental setup, including elements like the data set, the experimental protocol and the parametrization, followed by the presentation and discussion of the results.

4.1. Data sets and experimental protocol

The experimental validation, of the described approach, will be made assessing its performance on historical data vs other comparable GE-based strategies, more specifically the *Static*, *Naif* and *Majority* described in section 3.2, together with two ensemble-based alternatives, *Weighted1* and *Weighted2*, that rely on weighted voting schemes.

Regarding the experimental sample, the study will consider two datasets that cover 13-year worth of daily data, from 2005 to 2017. The first one corresponds to the Standard & Poor's 500 index, obtained from Datastream, and the second one, the daily risk-free return, obtained from the Federal Reserve Bank of Atlanta and available at *https://fred.stlouisfed.org/series/TB3MS*.

The process will start extracting two subsamples out of the original dataset. The first one, from 2005 to 2012 will be used for parametrization purposes. There, we will set the basic parameters for GE and, based on preliminary experiments, we will choose the number of models to be included in the ensemble, s, and the inertia factor, i. Once these values are determined, we will run main experiments based on the second subsample that covers the period from 2009 to 2017. It is worth noting that the partial overlapping of the two subsamples has to do with the size of the sliding window used to evolve the rules. We will use 3 years worth of daily data, hence the evaluation periods for parametrization and benchmarking purposes are non-overlapping. The comparison of the six methods will be based on net returns on 2013, 2014, 2015, 2016 and 2017.

Given that GE is a stochastic method, the experiments will be run 30 times. The statistical significance of the results will be formally tested using the protocol that follows:

First, the normality of returns will be assessed using the Kolmogorov-Smirnov test applying the Lilliefors correction. In case that it is rejected, the Wilcoxon's non-parametric test sign ranges will be applied. Otherwise, homoscedasticity will be evaluated using Levene's test. At that point, depending on whether we can reject homoskedasticity or not, we will rely on a standard t-test or the Welch test.

It is worth noting that, even though we will evaluate the approach using daily returns, the algorithm is basically agnostic to time scale. As long the technical indicators and operators used to generate the rules are appropriately selected and parametrized to capture relevant information at the preferred time frame, the algorithm will offer good performance. Operating intraday would require some adaptations to tackle the complexity of issues like managing market opening and closing times, but there should not be relevant differences in terms of the algorithm between using daily stock returns or, for instance, monthly ones.

4.2. Parametrization

The parametrization of the core algorithm was based on some preliminary experiments and values commonly found in the literature. We started from them, and then explored several configurations out-of-sample in the neighborhood.

As a result of process, we used a population size of 500 individuals, 50 generations and implemented elitism. Every generation the best strategy is selected and copied to the next one. Individuals were initialized using geometric series. For this sake, we used a minimum initial complexity value of 5 and 0.85 as the growth probability of the population initialization algorithm.

Regarding the main genetic operators, the key elements were defined as follows:

- Crossover: one-point crossover is performed with a probability of 0.85 on individuals selected by tournament. Given two individuals, A and B, the operator selects two random indices i and j, one per vector. Then, it swaps the string of genes from A_i to A_{end} with the genes from B_j to B_{end}, generating two new individuals. Finally, these are then subject to a truncation mechanism that eliminates the integers in the vector genotypes that have not been used.

- Duplication: given an individual selected by tournament, with a probability of 0.05, the operator selects a sequence from the individual in the range defined by two random indices, and appends it at the end. Finally, the process ends with truncation.

- Mutation: uniform mutation is applied with a probability of 0.1 on individuals chosen by tournament selection. The operator randomly modifies genes within a specified range (-128, 127) with a probability 0.05, allowing a circular wrapping of the genes vector up to 16 times. As with the other operators, mutated individual are truncated before their introduction in the new population.

In order to improve variability in the population, both during initialization and mutation, should the same individual appear more than once, there will be up to 100 attempts to replace it with a new one.

Regarding the ensemble, we initially performed some exploratory parametrization experiments using year 2012. The combinations that we considered included ensembles of 3, 5, 9, 13, 15, 17, 21 and 25 models and inertia factors of 2/3, 3/4 and 4/5. For all configurations, the experiments were ran 30 times. The best results were obtained with 3 models and unanimity (for that ensemble size both 3/4, 4/5 and unanimity are the same). To make sure that these results were reasonably stable over time, we extended the analysis, computing the average of yearly returns for all the periods ending at the end of 2012 from 2008 onward. These results are summarized in table 2. As we can see, regardless of whether we consider periods of one or five years, the configuration seems to be appropriate. That is also the case for 2, 3 and 4-year periods.

4.3. Experimental results

The results of the experiments are reported in Table 3. There, we provide the main descriptive statistics for the returns obtained in the test samples over 30 experiments for five years. In addition to the ensemble approach with *Inertia* that we discussed, we report the performance of five benchmarks: the standard *Static* approach, where a single trading rule is obtained from the three years closest to the test period and it is subsequently used for the whole test year; the one that uses a different model per time step, *Naif*; the results

Table 2: Parametrization tests. Mean annual return over 30 experiments for the different combinations of number of models in the ensembles and the inertia factor over the specified periods.

Models	Inert. factor	2008-12	2009-12	2010-12	2011-12	2012
3	2/3	-0.0356	0.0097	0.0032	0.0443	0.1734
3	3/4	-0.0128	0.0446	0.0356	0.0742	0.2043
3	4/5	-0.0128	0.0446	0.0356	0.0742	0.2043
5	2/3	-0.0307	0.0279	0.0133	0.0423	0.1955
5	3/4	-0.0307	0.0279	0.0133	0.0423	0.1955
5	4/5	-0.0359	0.0390	0.0287	0.0600	0.1667
9	2/3	-0.0312	0.0354	0.0231	0.0511	0.1742
9	3/4	-0.0430	0.0366	0.0243	0.0539	0.1523
9	4/5	-0.0615	0.0357	0.0271	0.0585	0.1463
13	2/3	-0.0488	0.0389	0.0278	0.0644	0.1500
13	3/4	-0.0692	0.0223	0.0102	0.0416	0.1453
13	4/5	-0.0836	0.0144	0.0001	0.0238	0.1416
17	2/3	-0.0714	0.0218	0.0113	0.0337	0.1560
17	3/4	-0.0682	0.0280	0.0194	0.0449	0.1566
17	4/5	-0.0705	0.0241	0.0142	0.0361	0.1485
21	2/3	-0.0585	0.0329	0.0281	0.0505	0.1512
21	3/4	-0.0656	0.0260	0.0183	0.0379	0.1303
21	4/5	-0.0681	0.0267	0.0190	0.0431	0.1237
25	2/3	-0.0642	0.0356	0.0306	0.0482	0.1265
25	3/4	-0.0739	0.0294	0.0215	0.0452	0.1260
25	4/5	-0.0891	0.0149	0.0045	0.0326	0.1403

of an ensemble that relies on simple majority to make the recommendations, *Majority*, and two variations on the latter that use weighted voting schemes, *Weighted1* and *Weighted2*.

The last three consider the same trading rules as *Inertia*. The difference is the way that the system combines the recommendations to generate the output of the ensemble. *Weighted1* and *Weighted2* prioritize to different degrees the recommendations of the most recent models in the ensemble and, like *Majority*, do not implement any inertia element. The former uses the weights $[0.1, 0.15, 0.2, 0.25, 0.3]$, where the first position represents the weight attributed to the recommendation of the oldest trading rule in the ensemble, and the latter $[0.05, 0.1, 0.15, 0.25, 0.45]$, which emphasizes significantly more the importance the of the most recent rules.

The best average performance in terms of net returns is obtained by the ensemble with the inertia component. If we consider average yearly returns over the 5-year period, this approach obtained 4.77% vs. 2.55% of the *Static* approach and the 2.09% of the ensembles that make recommendations based on majority vote. The other two ensembles, *Weighted1* and *Weighted2*, obtained slightly worse results, specially the latter, and the worst performing system, with an average yearly net loss of 3.96%, entailed using a sliding window to train rules that are used only once. The *Inertia* strategy also offered advantages in terms of reliability, as the average of the yearly return variances 0.0007 vs. the second most stable approach, *Static*, with 0.001. The strategies based on weighted voting turned out to be the worst in terms of this indicator.

There is, however, some variability once we analyze the results year by year. While *Inertia* dominates the rest in three out of five years, the *Static* one beats the rest in 2015 and 2016, which were also the years where the market offered the worst performance. Even though the relative performance of *Static*, *Naif*, *Majority*, *Weighted1* and *Weighted2* changes over time, *Inertia* dominates consistently the last four, and the first weighting scheme outperforms systematically beats the second.

The statistical significance of the median return differences vs. *Inertia* was formally tested using Wilcoxon's Test. As we can see, most of the differences are significant at 1%. The only exceptions are the differences of *Inertia* vs. the rest of the ensembles in 2013. That year, the difference vs the simple voting mechanism was significant at 5%, while we could not reject the null hypothesis of equality vs the two weighted voting approaches.

In order to gain some additional insight into the mechanism that might

Table 3: Net return. Includes transactions costs. Main descriptive statistics over 30 runs. Test results.

	Strategy	Mean		Median	Var.	Max.	Min.
2013	Inertia	0.1423		0.1400	0.0001	0.1691	0.1298
	Static	0.0381	**	0.0409	0.0001	0.0714	0.0051
	Naif	0.1311	**	0.1323	0.0003	0.1715	0.0909
	Majority	0.1367	*	0.1363	0.0002	0.1557	0.0964
	Weighted1	0.1387		0.1363	0.0001	0.1616	0.1190
	Weighted2	0.1372		0.1387	0.0001	0.1582	0.1088
2014	Inertia	0.0947		0.0938	0.0006	0.1048	0.0749
	Static	0.0606	**	0.0827	0.0015	0.1042	0.0079
	Naif	0.0547	**	0.0526	0.0002	0.0813	0.0281
	Majority	0.0852	**	0.0876	0.0001	0.0969	0.0471
	Weighted1	0.0803	**	0.0825	0.0001	0.0959	0.0490
	Weighted2	0.0782	**	0.0784	0.0001	0.0961	0.0548
2015	Inertia	-0.0511		-0.0468	0.0006	-0.0214	-0.1327
	Static	-0.0160	**	-0.0123	0.0001	-0.0123	-0.0600
	Naif	-0.1687	**	-0.1685	0.0024	-0.0574	-0.2564
	Majority	-0.0903	**	-0.0817	0.0017	-0.0303	-0.1738
	Weighted1	-0.0959	**	-0.0807	0.0023	-0.0379	-0.2052
	Weighted2	-0.1266	**	-0.1270	0.0016	-0.0488	-0.1958
2016	Inertia	-0.0089		-0.0083	0.0011	0.0579	-0.0731
	Static	0.0241	**	0.0043	0.0016	0.0889	-0.0223
	Naif	-0.1455	**	-0.1384	0.0022	-0.0794	-0.2582
	Majority	-0.0448	**	-0.0383	0.0012	0.0075	-0.1231
	Weighted1	-0.0489	**	-0.0372	0.0026	0.0560	-0.1418
	Weighted2	-0.0828	**	-0.0778	0.0023	0.0043	-0.1922
2017	Inertia	0.0615		0.0621	0.0009	0.1076	0.0065
	Static	0.0204	**	0.0092	0.0016	0.1668	0.0091
	Naif	-0.0718	**	-0.0755	0.0011	0.0295	-0.1316
	Majority	0.0176	**	0.0294	0.0022	0.1094	-0.0753
	Weighted1	0.0260	**	0.0209	0.0021	0.1480	-0.0661
	Weighted2	0.0025	**	0.0057	0.0023	0.0943	-0.1136
Mean	Inertia	0.0477		0.0481	0.0007	0.0836	0.0011
	Static	0.0255		0.0250	0.0010	0.0832	-0.0120
	Naif	-0.0396		-0.0419	0.0012	0.0311	-0.1054
	Majority	0.0209		0.0267	0.0011	0.0678	-0.0458
	Weighted1	0.0200		0.0244	0.0014	0.0847	-0.0490
	Weighted2	0.0017		0.0036	0.0013	0.0608	-0.0676

** Significant vs. Inertia at 1% * Significant vs. Inertia at 5%.

explain the differential performance, we looked into the number of transactions required by the strategies. The data provided in Table 4 makes apparent the discrepancies in this regard among the six approaches.

The two that resulted in the lowest number of transactions are *Inertia* and *Static*. Depending on the period one ranks higher than the other. On the other side of the spectrum, however, the *Naif* adaptive strategy systematically results in more purchase and sell orders. There is a clear inverse relationship between this figure and profitability, and transaction costs severely undermine the performance of this approach. Among the two weighted strategies, the second one is associated with a larger number of orders. This seems reasonable as, in practice, the disproportionate weight given to the recommendation of the most recent rule is likely to result in a behavior that is more similar to that of the *Naif* strategy.

Interestingly, the *Static* approach resulted in a good number of rules that did not provide any purchase signal in 2017 and, therefore, stayed out of the market for the whole period earning the risk-free return. Another piece of evidence supporting the importance of transaction costs would be the fact that *Inertia* beat the *Static* approach in the two periods where it required fewer transactions. However, all this is more apparent once we consider gross performance.

Table 5 represents gross returns, that is, the figures reported in Table 3 plus transaction costs. If we compute average returns over the five years, we see that the *Naif* strategy, with 7.21%, offers the best results. These are followed by *Majority*, 6.666% and *Weighted2*, 6.663%. The traditional *Static* approach, which offered the second best average performance in net terms, provided the worst one in gross ones, 3.73%.

Once again *Inertia* seems to offer the most reliable results among the six alternatives. The average of the yearly return variances, 0.0004, is about half of the second most stable approach, *Majority* with 0.0008.

This experimental evidence strongly supports both the importance of using adaptive approaches, as they show great potential vs static ones, and controlling transaction costs, which are clearly a major factor. The suggested *Inertia* strategy provides a good compromise between these contradicting objectives, as it offers flexibility to adapt to structural change, though less than *Naif*, at the same time that it limits the number of transactions, even if it does it to a lower extent than *Static*.

Table 4: Number of transactions. Main descriptive statistics over 30 runs. Test results.

	Strategy	Mean	Median	Var.	Max.	Min.
2013	Inertia	4.07	4	0.13	6	4
	Static	7.20	4	48.17	32	4
	Naif	14.00	14	16.00	20	4
	Majority	5.80	6	3.96	10	4
	Weighted1	5.67	6	2.51	10	4
	Weighted2	5.07	4	1.86	10	4
2014	Inertia	4.93	5	2.96	8	2
	Static	7.07	4	25.31	14	2
	Naif	14.73	14	11.72	22	8
	Majority	8.00	8	1.66	10	6
	Weighted1	8.60	8	3.08	12	6
	Weighted2	9.13	8	4.6	14	6
2015	Inertia	8.80	8	3.20	12	4
	Static	2.40	2	1.21	6	2
	Naif	42.40	42	24.94	56	36
	Majority	17.00	18	10.69	26	10
	Weighted1	19.20	18	17.27	28	14
	Weighted2	24.33	24	21.26	40	18
2016	Inertia	11.00	12	3.79	14	8
	Static	5.53	4	13.43	18	2
	Naif	72.47	72	58.40	92	62
	Majority	29.33	28	19.95	40	18
	Weighted1	28.67	30	28.23	42	16
	Weighted2	43.13	42	36.33	58	28
2017	Inertia	8.00	8	5.24	12	4
	Static	1.40	0	13.28	12	0
	Naif	79.40	81	102.39	100	52
	Majority	31.33	32	35.40	44	20
	Weighted1	28.13	30	76.67	56	10
	Weighted2	45.93	48	111.86	78	30
Mean	Inertia	7.36	7.40	3.06	10.40	4.40
	Static	4.72	2.80	20.28	16.40	2.00
	Naif	44.60	44.60	42.69	42.00	48.40
	Majority	18.29	18.40	17.50	26.00	11.60
	Weighted1	18.05	18.40	25.55	29.60	10.00
	Weighted2	25.52	25.20	35.18	40.00	17.20

Table 5: Gross return. Main descriptive statistics over 30 runs. Test results.

	Strategy	Mean		Median	Var.	Max.	Min.
2013	Inertia	0.1525		0.1500	0.0001	0.1791	0.1398
	Static	0.0561	**	0.0509	0.0004	0.1215	0.0193
	Naif	0.1661	**	0.1654	0.0003	0,2215	0.1409
	Majority	0.1512		0.1519	0.0001	0.1676	0.1214
	Weighted1	0.1528		0.1535	0.0001	0.1766	0.1340
	Weighted2	0.1540		0.1562	0.0001	0.1732	0.1338
2014	Inertia	0.1070		0.1067	0.0001	0.1248	0.0899
	Static	0.0783		0.0927	0.0007	0.1092	0.0429
	Naif	0.0915	**	0.0890	0.0001	0.1165	0.0716
	Majority	0.1052		0.1074	0.0001	0.1184	0.0671
	Weighted1	0.1018	**	0.1027	0.0001	0.1175	0.0740
	Weighted2	0.1010	**	0.0987	0.0001	0.1161	0.0798
2015	Inertia	-0.0291		-0.0223	0.0001	-0.0014	-0.1077
	Static	-0.0100	**	-0.0073	0.0001	-0.0073	-0.0451
	Naif	-0.0627	**	-0.0627	0.0024	0.0576	-0.1379
	Majority	-0.0478	*	-0.0415	0.0014	0.0084	-0.1288
	Weighted1	-0.0479	*	-0.0295	0.0018	0.0041	-0.1434
	Weighted2	-0.0658	**	-0.0692	0.0014	0.0112	-0.1361
2016	Inertia	0.0186		0.0217	0.0010	0.0779	-0.0381
	Static	0.0379		0.0293	0.0015	0.0952	0.0073
	Naif	0.0357		0.0409	0.0018	0.1346	0.0282
	Majority	0.0285		0.0355	0.0010	0.0775	0.0481
	Weighted1	0.0228		0.0317	0.0022	0.1310	-0.0518
	Weighted2	0.0250		0.0275	0.0019	0.1243	-0.0497
2017	Inertia	0.0815		0.0818	0.0009	0.1260	0.0277
	Static	0.0239	**	0.0092	0.0017	0.1718	0.0092
	Naif	0.1296	**	-0.0708	0.0013	0.1968	0.0591
	Majority	0.0959		0.0294	0.0014	0.1694	0.0323
	Weighted1	0.0963		0.0894	0.0015	0.2030	0.0327
	Weighted2	0.1174	**	0.1110	0.0014	0.1922	0.0479
Mean	Inertia	0.0661		0.0675	0.0004	0.1012	0.0223
	Static	0.0373		0.0350	0.0009	0.0980	0.0067
	Naif	0.0721		0.0723	0.0012	0.1454	0.0324
	Majority	0.0666		0.0565	0.0008	0.1082	0.0280
	Weighted1	0.0652		0.0696	0.0011	0.1264	0.0091
	Weighted2	0.0663		0.0648	0.0010	0.1234	0.0151

** Significant vs. Inertia at 1% * Significant vs. Inertia at 5%.

5. Summary and Conclusions

The evolution of trading rules with flexible representation using Grammatical Evolution in its standard version involves obtaining a single rule based on a training period that is subsequently used to generate recommendations over time. Given the prevalence of structural change in financial series, this poses a problem.

In this paper we suggested using an ensemble of trading rules obtained using Grammatical Evolution on a sliding window. The system has a critical component, the voting mechanism, that creates an inertia that reduces overtrading.

The experimental evaluation of the approach involved five comparable alternatives based on the same core algorithm over 5-years. They included the standard static approach, *Static*; one that uses a different model per time step, *Naif*, and three based on ensembles. Out of these three, one relies on simple majority to make the recommendations, *Majority* and two on weighted voting the emphasized the importance of recent rules to different degrees, *Weighted1* and *Weighted2*.

The results support the superiority of the ensemble with the inertia component in terms of net return, followed by the static approach and the rest of the ensembles. In relation to these, there seems to be a gradient that shows that biasing the weighing towards the most recent rules degrades performance. The version that implemented simple majority rule without inertia offered slightly better average results than those whose weight distribution was closer to being linear or exponential. The worst performing system was the *Naif* strategy. The suggested strategy also offered advantages in terms of uncertainty.

The analysis of the impact of transaction costs makes apparent the importance of limiting overtrading. There is a clear inverse relationship between the number of purchase and sell orders, and profitability. The *Naif* strategy trades much more often than the rest, and commissions erode its return very significantly.

These findings emphasize the importance of keeping a balance between the need to adapt to structural changes and the risk of updating constantly trading rules that are implicitly optimized for the longer run.

Future lines of work include expanding the experimental analysis to other financial assets or indices. Given that the core problem of overtrading has been clearly identified for GE-based rules, a limitation that is also very likely to be shared with those obtained with Genetic Programming, the search for algorithmic solutions that adapt these techniques to be both adaptable and prevent excessive transactions is likely to be fruitful.

6. Acknowledgements

The authors would like to acknowledge the financial support of the Spanish Ministry of Science, Innovation and Universities under project PGC2018-096849-B-I00 (MCFin).

References

[1] F. Allen, R. Karjalainen, Using genetic algorithms to find technical trading rules, Journal of Financial Economics 51 (1999) 245–271.

[2] C. Setzkorn, L. Dipietro, R. Purshouse, Evolving Rule-Based Trading Systems, 36th Annual Meeting of the CEA (2002).

[3] C. J. Neely, Risk-adjusted, ex ante, optimal technical trading rules in equity markets, International Review of Economics & Finance 12 (2003) 69–87.

[4] N. Navet, S. H. Chen, On predictability and profitability: Would GP induced trading rules be sensitive to the observed entropy of time series?, Studies in Computational Intelligence 100 (2008) 197–210.

[5] D. Lohpetch, D. Corne, Discovering effective technical trading rules with genetic programming: Towards robustly outperforming buy-and-hold, in: 2009 World Congress on Nature and Biologically Inspired Computing, NABIC 2009 - Proceedings, pp. 439–444.

[6] D. Lohpetch, D. Corne, Outperforming buy-and-hold with evolved technical trading rules: Daily, weekly and monthly trading, in: Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), volume 6025 LNCS, pp. 171–181.

[7] J. How, M. Ling, P. Verhoeven, Does size matter? A genetic programming approach to technical trading, Quantitative Finance 10 (2010) 131–140. ISSN 1469-7688.

[8] A. Esfahanipour, S. Mousavi, A genetic programming model to generate risk-adjusted technical trading rules in stock markets, Expert Systems with Applications 38 (2011) 8438–8445.

[9] V. Manahov, R. Hudson, H. Hoque, Return predictability and the 'wisdom of crowds': Genetic Programming trading algorithms, the Marginal Trader Hypothesis and the Hayek Hypothesis, Journal of International Financial Markets, Institutions and Money 37 (2015) 85–98.

[10] Y. Lee, T. J. Choi, C. W. Ahn, Multi-objective evolutionary approach to select security solutions, CAAI Transactions on Intelligence Technology 2 (2017) 64–67.

[11] X. Zhao, G. Xu, D. Liu, X. Zuo, Second-order de algorithm, CAAI Transactions on Intelligence Technology 2 (2017) 80–92.

[12] Z.-H. Zhou, Ensemble methods: foundations and algorithms, Chapman and Hall/CRC, 2012.

[13] A. Pimenta, C. A. L. Nametala, F. G. Guimarães, E. G. Carrano, An automated investing method for stock market based on multiobjective genetic grogramming, Computational Economics (2017) 1–20. ISSN 15729974.

[14] D. Opitz, R. Maclin, Popular ensemble methods: An empirical study, Journal of artificial intelligence research 11 (1999) 169–198.

[15] R. Polikar, Ensemble based systems in decision making, IEEE Circuits and systems magazine 6 (2006) 21–45.

[16] L. Rokach, Ensemble-based classifiers, Artificial Intelligence Review 33 (2010) 1–39.

[17] H. Jamalinia, S. Khalouei, V. Rezaie, S. Nejatian, K. Bagheri-Fard, H. Parvin, Diverse classifier ensemble creation based on heuristic dataset modification, Journal of Applied Statistics 45 (2018) 1209–1226.

[18] Y. Yu, M. Li, Y. Fu, Forest type identification by random forest classification combined with spot and multitemporal sar data, Journal of Forestry Research 29 (2018) 1407–1414.

[19] L. K. Hansen, P. Salamon, Neural network ensembles, IEEE transactions on pattern analysis and machine intelligence 12 (1990) 993–1001.

[20] M. P. Perrone, L. N. Cooper, When networks disagree: Ensemble methods for hybrid neural networks, in: How We Learn; How We Remember: Toward an Understanding of Brain and Neural Systems, 1995, pp. 342–358.

[21] D. W. Opitz, J. W. Shavlik, Actively searching for an effective neural network ensemble, Connection Science 8 (1996) 337–354.

[22] C. Grosan, A. Abraham, Stock market modeling using genetic programming ensembles, in: Genetic systems programming, Springer, 2006, pp. 131–146.

[23] A. Brabazon, M. O'Neill, Evolving technical trading rules for spot foreign-exchange markets using grammatical evolution, CMS 1 (2004) 311–327.

[24] I. Dempsey, M. O'Neill, A. Brabazon, Live Trading with Grammatical Evolution, GECCO 2004 Workshop Proceedings (2004) 9137–9142. Doi: https://doi.org/10.1109/CEC.2006.1688631, ISBN 0-7803-9487-9.

[25] I. Contreras, J. I. Hidalgo, L. Nunez-Letamendia, Combining Technical Analysis and Grammatical Evolution in a Trading System, in: Applications of Evolutionary Computing, EvoApplications 2013: EvoCOMNET, EvoCOMPLEX, EvoENERGY, EvoFIN, EvoGAMES, EvoIASP, EvoINDUSTRY, EvoNUM, EvoPAR, EvoRISK, EvoROBOT, EvoSTOC, volume 7835, Springer, Berlin, Heidelberg, 2013, pp. 244–253.

[26] I. Contreras, J. I. Hidalgo, L. Núñez-Letamendia, A GA Combining Technical and Fundamental Analysis for Trading the Stock Market, in: A GA combining technical and fundamental analysis for trading the stock market, Springer, Berlin, Heidelberg, 2012, pp. 174–183.

[27] H. Schmidbauer, A. Rösch, T. Sezer, V. S. Tunalioğlu, Robust trading rule selection and forecasting accuracy, Journal of Systems Science and Complexity 27 (2014) 169–180.

[28] K. Deb, S. Agrawal, A. Pratap, T. Meyarivan, A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization:

Nsga-ii, in: International Conference on Parallel Problem Solving From Nature, Springer, pp. 849–858.

[29] G. Wilson, D. Leblanc, W. Banzhaf, Stock trading using linear genetic programming with multiple time frames, GECCO '11: Proceedings of the 13th annual conference on Genetic and evolutionary computation (2011) 1667–1674. Doi: https://doi.org/10.1145/2001576.2001801, ISBN 9781450305570.

[30] S. Deng, A. Sakurai, Foreign exchange trading rules using a single technical indicator from multiple timeframes, Proceedings - 27th International Conference on Advanced Information Networking and Applications Workshops, WAINA 2013 (2013) 207–212. Doi: https://doi.org/10.1109/WAINA.2013.7,ISBN 9780769549521.

[31] J. T. N, Genetic Algorithms Based Combined Strategy Optimization of Select Technical Trading Rules, International Journal of Innovative Research in Technology 1 (2014) 137–145.

[32] J. Machado, R. Neves, N. Horta, Developing multi-time frame trading rules with a trend following strategy, using GA, in: 17th Genetic and Evolutionary Computation Conference, GECCO 2015, pp. 765–766. Doi: https://doi.org/10.1145/2739482.2764885,ISBN 9781450334884.

[33] C. Ryan, J. Collins, M. O'Neil, Grammatical Evolution: Evolving Programs for an Arbitrary Language, Springer Berlin Heidelberg (1998) 83–96.

[34] J. R. Koza, P. J. Angeline, Genetic programming: On the programming of computers by means of natural selection, volume 33, MIT press, 1992.

[35] L. Becker, M. Seshadri, GP-evolved technical trading rules can outperform buy and hold, Proceedings of the Sixth International Conference on Computational Intelligence and Natural Computing, Embassy Suites Hotel and Conference Center, Cary, North Carolina USA, September 26-30 2003 (2003).

[36] M. O'Neill, C. Ryan, M. Keijzer, M. Cattolico, Crossover in Grammatical Evolution: The Search Continues, in: Genetic Programming: 4th European Conference, EuroGP 2001 Lake Como, Italy, April 18–20,

2001 Proceedings, volume 2038, Springer Berlin Heidelberg, 2001, pp. 337–347.