Applied Soft Computing 64 (2018) 596–613

Contents lists available at ScienceDirect

Applied Soft Computing

journal homepage: www.elsevier.com/locate/asoc

Combining Support Vector Machine with Genetic Algorithms to optimize investments in Forex markets with high leverage

Bernardo Jubert de Almeida, Rui Ferreira Neves*, Nuno Horta

Instituto de Telecomunicações, Instituto Superior Técnico – Universidade de Lisboa, Lisboa, Portugal

ARTICLE INFO

Article history:
Received 28 September 2016
Received in revised form
18 December 2017
Accepted 30 December 2017
Available online 3 January 2018

Keywords:
Genetic Algorithms
Support Vector Machine
Leverage
Forex

ABSTRACT

This work proposes a new approach, based on Genetic Algorithms and Support Vector Machine to trade in the forex market. In this work, a new algorithm capable of generating technical rules to make investments with a given amount of leverage depending on the certainty of the prediction is presented. To forecast those predictions, a combination of a Support Vector Machine (SVM) algorithm – to identify and classify the market in three different stages –, and a Dynamic Genetic Algorithm – to optimize trading rules in each type of market, is used. The optimization of the trading rules is based on several technical indicators. Forex data for the EUR/USD currency pair, in a timeframe between the years of 2003 and 2016, is used as training and test data. The proposed architecture for the machine learning system, as well as the implementation and study of the proposed system is described in detail. The use of an hybrid system, combining a SVM and a GA with dynamic approaches such as hyper-mutation and adaptability approaches by training three different GA's for each type of market, provide a new approach for FOREX trading, where it is possible to classify trends using price sequences and therefore using the same classification for optimizing investment strategies with the most appropriate GA. Finally, the work shows promising results during the test period between the 2nd of January of 2015 until the 2nd of March of 2016, where the *Return on Investment* obtained is 83%.

1. Introduction

The Foreign Exchange (FX) Market is a global market for currency trading; it is considered the most liquid financial market in the world. According to the Bank for International Settlements; trading in Forex markets averaged $5.3 trillion per day in April 2013 [1]. Another advantage of this kind of markets; is that; transactions are available 24 hours a day; every day from Monday to Friday

This financial problematic is a widely studied subject, and constitutes a rather motivating problem to researchers on the *Machine Learning* field of studies. In this work, more precisely, it is intended to develop an investment strategy, using different *Leverage ratios*, applied to Foreign Exchange Markets, or more specifically on FOREX markets. The work is motivated by the study of techniques within the area of Evolutionary Computation, that are capable of learning past events in order to predict future ones, and generate investment rules accordingly, thus, potentiating the quality and the profits of an investment made by a human. Said that, it is intended to study different approaches using Genetic Algorithms (GA) to optimize investment rules and Support Vector Machine (SVM) to classify the different types of markets. To obtain high returns in this type of markets is essential to apply a good strategy of leverage [2]. Financial leverage is a very common tool used in financial markets, and its purpose is to increase the return of an investment, being the losses equally great.

Traditional genetic algorithms are good for static problems, i.e., problems that do not change through time and remain in the same environment. Real world problems are constantly changing, making species to adapt, and the ones who adapt faster survive, but, what happens when the environment changes? Is it better to restart the algorithm? This problem can be solve using Dynamic Genetic Algorithms that combine a set of tools to predict and better adapt to new environments, giving new solutions without restarting the whole process. As it was stated before, this work's main focus is Foreign Exchange markets using a Genetic Algorithm to explore the possibilities of financial leverage relying on SVM to market identification on Forex. *The work developed in this paper, had the following contributions*: Creates a classifier model based on a Support Vector Machine algorithm, that classifies three types of markets to identify the environment used for three different Genetic Algorithms that

* Corresponding author.
E-mail addresses: bernardo.jubert@gmail.com (B. Jubert de Almeida), rfmfn@lx.it.pt, rui.neves@tecnico.ulisboa.pt (R. Ferreira Neves), nuno.horta@lx.it.pt (N. Horta).

https://doi.org/10.1016/j.asoc.2017.12.047
1568-4946/

are trained in each one of the environments; Produces Dynamic Genetic Algorithm that introduces adaptability approaches and takes advantage of a memory system to save the populations of each GA, together with the environment information; Introduces a new leverage solution based on optimized values from the Genetic Algorithm chromosome, together with the study of the evolution of the price values of Forex market.

Finally, this paper is composed by five sections. Section 1 – Introduction, introduces the work that is developed in this work, with a brief explanation about Evolutionary Computation. Section 2 – Related Work –, explains some fundamental concepts related to financial markets used, and existing solutions in this field of studies. Following, Section 3 – Proposed System Architecture –, describes the proposed architecture of the developed system, and the development of the theoretical model used on the matter; Section 4 – System Validation – Describes the evaluation procedure that evaluates de developed strategy. Finally, Section 5 – Conclusion – summarizes the work done and draws the conclusion.

2. Background and state-of-the-art

This section provides, fundamental concepts about Foreign Exchange Markets, Technical Indicators, focusing on existing solutions regarding several areas within Machine Learning and Evolutionary Computation to better understand the problem and the proposed solution.

2.1. Financial concepts

Forex is a largely well known Foreign Exchange Index, and the approach developed in this work is applied to this specific market. Forex has seven major currencies that are traded on a daily basis, 24 h per day, from Monday to Friday, and more specifically in this work, the currency used is the EUR/USD. The currencies are called *Cross Currency Pairs*, being the currency from the left side called the base currency (EUR) or the selling currency and the one from the right side the counter currency (USD) or the buying currency. Currency quotes can be divided into two parts, the first one is the *ask price* for buying base currency and *bid price* to sell base currency.

2.2. Technical analysis

There are two types of financial analysis, *Fundamental Analysis and Technical Analysis.* In this work the objective is to focus only on the second approach. Technical Analysis is the tool required to analyse future prices of financial assets relying on past information based on mathematical transformations of price quotes. There are two main types of technical indicators:

- Trend Following – are used to understand trends i.e., they work better to understand the present trend in order to learn about the asset's behaviour, *trend following* indicators are of great importance to identify entry or exit points, i.e., these indicators identify if the cycle of a trend has begun (entry point when asset's price is rising), or ended (exit point when asset's price starts to decrease);
- Momentum Oscillators – predict sudden changes on the asset's behaviour, meaning that *Momentum Oscillators* can keep track with the speed of the price movement variation and are able to measure the strength and speed of the price trend direction. *Momentum Oscillators* can have major importance to decide the amount of leverage used in the investment. Table 1 shows the Technical indicators that will be used in this work, both *Trend Following* and *Momentum Oscillators*.

2.3. Concept of leverage

Leverage is used to profit from fluctuations in exchange rates between two different currencies (currency pairs). Leverage is a loan that is provided to the investor by a broker, i.e., companies that are specialized in financial trading lend money to the investors to increase their investment. Take a small investment of 1000 euros, if the leverage applied is 100:1, then the investment made is 100,000 euros. The amounts of leverage, that brokers are allowed to provide are 100:1, 50:1 and 10:1, although the first three ratios are extremely risky to apply. This can result on two possible outcomes, the first is, if the investment goes well and the profit is for example 3%, equivalent to 3000 euros, the trader made an investment of 1000 euros and took profits that could only be achieved with leverage. Without leverage, the profits would be 30 euros. The second outcome is more pessimistic, due to the fact that using leverage comes with a greater risk and can turn against investors, i.e., if the investment made was the same 100,000 euros, and loses the same 3%, the trader has to pay to the broker 3000 euros. To overcome this kind of risks brokers, have "Stop Limits" to avoid this type of actions to occur. Fluctuations in Foreign Exchange Markets usually do not cross over 1% during intraday trading, permitting brokers to provide this much leverage.

2.4. Approaches on Forex – overview

Works on Foreign Exchange, showed some promising results, using different approaches in Evolutionary Computation (EC), and many different strategies were used to support the optimization algorithms. In this section, different approaches are divided by EC strategies and Machine Learning algorithms, starting with Neural Networks, Genetic Algorithms, then Support Vector Machine Algorithms (SVM). Regarding wider approach that were not based only on Genetic Algorithms (GA) and SVM's, Yao and Tan [3], reported empirical evidence that a neural network (NN) model is applicable to the prediction of foreign exchange rates, using time series and different technical indicators to capture the underlying "rules" of the movement in currency exchange rates. Evans et al., [4] focused on predicting intra-day market exchanges, and their research focused on a hybrid solution between *NN* and *GA*. Deng and Sakurai [5] applied a genetic algorithm to generate trading rules based on a single technical indicator (RSI), and multiple timeframes. Hirabayashi et al. [2], proposed to automatically generate trading rules based on technical indicators, focusing on calculating the most appropriate trade timing, instead of predicting trading prices. Myszkowski and Bicz [6] approached the Forex market with two decision trees, that are responsible for taking the decisions of opening long or short positions on EUR/USD currency pair. Sermpinis et al. [7], introduced a hybrid *Rolling Genetic Algorithm-Support Vector Regression*(*RG-SVR*) model to find optimal parameter solution. *RG-SVR* method genetically searches over a feature space and combines the optimal feature subsets for each exchange rate. Additionally, this work implements a *GA-SVM* and a *GA-SVR* hybrid systems that are evaluated alongside the *RG-SVR* approach. The results obtained in the trading performance, make a guideline for the work developed in this paper.

2.4.1. Works on Genetic Algorithms – stationary problems

Besides the referred works on Forex it is also important to underline some static problems about Genetic Algorithm works that were made in the financial problematic. Starting with the Evaluation Function or Fitness Function, e.g., Gorgulho et al. [8], uses the technical indicator *Return on Investment* (*ROI*) to evaluate each individual within the population. Other interesting method proposed by Aranha & Iba [9] was a multi-objective approach that was composed by two functions, a cumulative return and *Sharpe*

Table 1
Technical Indicators used.

	Method	Definition and Technical Rule	Type of indicator
RSI – Relative Strength Index	$\frac{\lvert R \rvert}{\lvert F \rvert} \times 100$	R – Sum of the absolute value of rising width in the past n days. F – Sum of the absolute value of rising and falling width in the past n days. Aims to buy when currency is sold too much, i.e., the price *i* slow, and to sell when it's bought too much, i.e., the price is high. The value of n used is between 9 and 14 days.	Momentum Oscillator
ROC – Rate of Change	$\frac{Xt-Xt-n}{Xt-n} \times 100$	Ratio between the current closing price and the price of the last *n* time periods. This ratio measures how fast, the price of the asset is changing. When the price is rising to quickly indicates that there is a great possibility of overbought conditions. When the price is falling to rapidly it is possible that the conditions are oversold.	*Momentum Oscillator*
MACD – Moving Average Convergence/Divergence	$\mathrm{MACDt}(s, l) = \mathrm{EMA}(s)_t - \mathrm{EMA}(l)_t$	MACD is the difference between two EMA's where *n* is the number of periods considered for the trigger signal; *s* is the number of periods considered for the shorter EMA (12 weeks) and *t* is the number of periods considered for the longer EMA (26 weeks).	Trend/Momentum Oscillator
	$Triggert\,(n) = EMA\,(n)\ of\ MACDt\,(s, l)$ $Histogram = MACD\,(s, l) - Triggert\,(n)$	MACD indicates a trend turning point. When the line of the MACD crosses the 0 value to the positive side indicates a "Buying Signal", when the line of the MACD crosses the 0 value to the negative side indicates a "Selling Signal".	
MA – Moving Average	$\frac{\sum_{i=1}^{n} X(i)}{n}$	Technique to short-term variation of price. Simple mean value of the past n days' prices, where $X\,(i)$ is the assets price on a specific day. Aims to buy when MA is increasing and/or the MA line crosses price line in a descending way. Aims to sell when MA is decreasing and/or the MA line crosses price line in an ascending way.	Trend Following
EMA – Exponential Moving Average	$\mathrm{EMA_t} = \mathrm{EMA_{t-1}}\,(\mathrm{n}) \times \left(1 - \frac{2}{n+1}\right) + \mathrm{X_t} \times \frac{2}{n+1}$	EMA averages asset's price and assigns more weight (exponentially) to the latest data. The technical rule to EMA is the same as the Moving Average.	Trend Following

ratio. Férnandez-Blanco et al. [10] uses a dynamic method, relying on the premise that the fitness landscape may change over time, thus applying different indicators in different time periods. As for the genetic operators, *Hirabayashi* et al. [2] and Yuan used *Tournament Selection Method* as a *selection* operator. Evans et al. [4], and Mendes et al. [11], relied on the *Roulette Selection Method*, to find the probability of a certain individual being selected, depending on the individual's fitness compared to the fitness values of the population. Many researchers have implemented different strategies to maintain diversity in a population. The first method shows the implementation of mutation operators with a certain mutation rate, e.g., Zhang and Ren [12], Hirabayashi et al. [2] and Gorgulho et al., [13]. Fernández – Blanco et al. [10] used a different approach to maintain diversity that relies on a small set of individuals that are randomly generated every generation, called "Immigrants", choosing this approach to get the trade off between fast computation and diversity. There are, also, several ways to perform crossover, e.g., Gorgulho et al. [14] compared three different methods, *Single Arithmetic Recombination*, *Whole Arithmetic Recombination* and *One-Cut Point*, concluding that the best results were from the One-Cut Point solution, in order to generate two distinct offspring. Hirabayashi et al. [2] used a Two-Cut Point technique to perform the crossover.

2.4.2. Works on Genetic Algorithms – non-stationary problems

To address non-stationary problems, it is necessary to implement Dynamic Genetic Algorithms, thus is important to develop dynamic problem generators to create dynamic test environments [15]. Yang [16] and Yang & Yao [15] developed a XOR DOP Generator that can create DOP's and detect changes in the environment through the XOR operator, shifting the population of an algorithm to a new location in the fitness landscape [17]; Contrarily to stationary problems, that only need few performance measures, e.g., convergence speed and success rate of reaching optimality [17], DOPs have over 20 measures that were studied by Nguyen et al. [18], thus dividing it in two types of measures: Optimality-based performance measures and *Behaviour-Based* performance measures. Branke [19] implemented a direct memory approach, to store the good solutions throughout time, and examined in what circumstances memory was useful. Yang and Yao [20] investigated associative memory to store the best solutions as well as environment information to improve the algorithms adaptability to the environment. DOPs require diversity to adapt more efficiently and to better adapt to environmental changes. Park et al. [21] implemented an algorithm with two populations to provide additional diversity to the main population through *crossbreeding*. Yang [22] developed a *Memory-Based Immigrants* approach, to maintain the diversity and guide immigrants towards the new envi-

Table 2
State-of-the-Art resumed.

Work	Date	Heuristic	Approach	Evaluation Function	Result Evaluation	Financial Application	Period	Best Return
[8]	2011	GA	Artificial Immune System	ROI	Comparison with B&H and Random	Stock Market	2003–2009	62.95%
[2]	2009	GA	Immigration Method	Profit Gained	Comparison NN, no leverage, B&H	FOREX – JPY/USD, AUD/JPY, EUR/JPY	2005–2008	38%
[9]	2007	GA	Seeding Legacy Size	Sharpe Ratio	Comparison to Index	NASDAQ and, NIKKEI	2000–2006	57%
[10]	2008	GA	–	Dynamic Method	Comparison to Random and B&H	Stock Market	–	50%
[16]	2003	DGA	Adaptive Selection, Dual GA	Royal Road Function, One Max Problem, Deceptive Function	Comparison Simple GA, Dual GA and Primal Dual GA	–	–	–
[15]	2005	DGA	PBIL, Dual PBIL	Deceptive Function	SGA, PIBL, Dual BPIL	–	–	–
[20]	2008	DGA	Multi-population; Memory scheme to PIBL; Associative memory;	Dynamic fitness function	ISGA, SPBIL, SPBILi, MPBIL, MPBILi	–	–	–
[21]	2008	DGA	Multi-population scheme with reserve populations	Dynamic fitness function	DPGA, IMGA, PDGA, SGA	–	–	–
[22]	2008	DGA	Elitism, Random immigrants, Memory-based immigrants	Sum of the profits of the selected items	SGA, MEGA, MSGA, MIGA, EIGA	–	–	–
[4]	2008	NN and GA	Elitism, Termination Criteria of 15 generations unchanged	Mean absolute error, Sharpe Ratio, Annualized Return, Correlation Coefficient	Monte Carlo Simulation, Random Method, comparison between actual data and predicted data	FOREX – GBP/USD, EUR/GBP, EUR/USD	2010–2012	72.5%
[5]	2013	GA	Elitism Multiple time-frame	RSI indicator	B&H, S&H, SVM, GA-simple time-frame, GA multiple time-frame	FOREX –EUR/USD	2011	1387 pips
[7]	2015	RG-SVR	Sliding window approach, RBF ν-SVR	Annualized Return	GA-SVM, GA-SVR, ARBF-PSO, MLP, HONN, RNN, PSN	FOREX- EUR/USD, EUR/GBP, EUR/JPY	1999–2012	–

ronment, using memory scheme to adapt the GA faster to the new search space by reusing past information. Cobb & Grenfenstette [23] proposed the hyper-mutation method to raise the mutation rate temporarily, in order to improve adaptive performance to the GA; Yang & Tinos [24] developed the hyper-selection method that aimed to raise the selection pressure temporarily to improve the quality of individuals. Simões & Costa [25] introduced a predictive model based on linear regressions and a Markov model that memorizes past information of when the changes in the environment occurred, to estimate when the next change will occur. In addition to the research done, it is shown on Table 2 some of the most relevant approaches that were studied to develop this work.

2.5. Introduction to this works approach

This work proposes an hybrid system between a SVM classifier and a Genetic Algorithm with dynamic approaches. Most works seen in the literature show different approaches like systems with standard Genetic Algorithms, and do not use this approach of combining a classifier such as an SVM and a Genetic Algorithm. The authors of "A. Hirabayashi, C. Aranha, and H. Iba, Optimization of the Trading Rule in Foreign Exchange using Genetic Algorithm" propose a GA to optimize forex trading using leverage, being that, the same objective of this work. For example, it is used a Standard Genetic Algorithm and a leverage strategy between 1 and 5 depending on a correlation function. Also, the authors have investigated other currencies such as the Japanese Yen and the Australian Dollar.

The research developed in this work provides different approaches in the EUR/USD investment where a Support Vector Machine is used to classify market trends, so that the solutions can perform well in different types of markets. Additionally, this work uses three types of Genetic Algorithm depending on the type of market the SVM classifies. Inside the Genetic Algorithm approach, it is used several dynamic approaches to improve the performance of a Genetic Algorithm, and it is introduced a new leverage solution based on optimized values from the Genetic Algorithm chromosome, together with the study of the evolution of the price values of Forex market.

3. Proposed system architecture

The proposed solution intends to develop an algorithm to trade *FX* currency pairs, using an *SVM* to categorize the type of market and a *Genetic Algorithm* to optimize the investment solution in order to trade with several levels of leverage. This section describes the system's architecture, more precisely the different layers of the system application and the way they were implemented. The set of implemented technologies will also be described, according to the research made in the previous section.

3.1. Overall architecture description

The system architecture has to be implemented into three separated layers, in order to make them independent, so that the user does not have to be concerned about the different layers at the same time. The system is divided into the *User Interface Layer*, *Forex Data Layer* (*FX Data Layer*) and *Optimization Layer*. *FX Data Layer* is divided into *Price Processing Module* and *Technical Rules Module*, and the *Optimization Layer* is divided into *SVM Module* and *GA Module*. Fig. 1 shows the overall description of the proposed system and it is important to explain the general idea behind this approach. The basic concept is to use an *SVM* classifier (*SVM Module*) to classify three types of markets (class "1" for Uptrend markets, class "0" for Sideways markets and class "-1" for Downtrend markets), in order to train three different *Genetic Algorithms*, for each type of market. The idea behind the proposed system is to use the *SVM* as a classifier and also use this environment information to complete the memory approach used on the Dynamic GA, with three different GAs, where each one of them invests in the FX market, depending on the type of environment selected, e.g., if the *SVM* classifies the market as an Uptrend (class type "1"), then, the module uses this information to use the *GA* that was trained for Uptrend markets. Another important aspect that is important to refer is that the *SVM Classifier* was experimented for two types of *features* used, one with *Technical Indicators* and the other with *Forex Price Sequences*, that are explained in detail in the *SVM Module* section. These two approaches are put against each other, in order to study the consequences of using price sequences instead of technical indicators, that store information about the price sequences. It is important to explain that the technical indicators were calculated in a separate module so they can be used by the *SVM* and *GA Module*. The calculations of the technical indicators are explained in the *Technical Rules Module*. As for the three GAs, that are exactly coded the same way (only difference is that each GA is trained for different types of markets), the fitness function used is supported by a voting system that uses technical indicators. Joining an SVM classifier with a Dynamic Genetic Algorithm, making a hybrid system to predict the FX market is one of this work's objectives, and adaptability approaches were introduced in the algorithm in order to improve its dynamical capabilities.

Architecture layers have to be independent from each other, to facilitate the implementation of different methods during the process. The algorithm will be put against three classic theories, *Random Walk* and *B&H* and *S&H*. The first theory is based on the though of buying and selling assets randomly, and the second is a traditional strategy in the financial market that states that on the long run the asset's price will return profit with "long" trades. The last approach is the same as the second, being the only difference that instead of adopting long positions, the trader adopts a short position. Additionally, the proposed system is also put against a Static Genetic Algorithm that presents the usual Genetic Operators that are usually seen in the literature, but this algorithm does not present any dynamic characteristic, being the objective, to compare the results between a Static and a Dynamic approach.

The mathematical model behind this architecture is composed by the *SVM model* and the *GA model*, The *SVM model* uses a *SVM Algorithm* inspired on the "*Widest Street Approach*" that classifies a sequence of prices (the mathematical logic can be applied for both features compared in this work) to create the training model, i.e., the training data was fed with supervised sequences of prices that had the information, if the market was bearish, bullish or sideways. The objective of having this training data is to refine the *SVM Algorithm* in the matter of boundary decisions, where to classify the samples it will only be necessary to make the dot product of an unknown vector in a high-dimensional space like $\phi\left(\vec{x_i}\right)\cdot\phi\left(\vec{u}\right)$, as a function of vectors in the original space. The advantage of this approach is that it is not necessary to know the transformation $\phi\left(\vec{x_i}\right)\cdot\phi\left(\vec{u}\right)$, because:

$$\phi\left(\vec{x_i}\right)\cdot\phi\left(\vec{u}\right) = K\left(x_i, \vec{u}\right) \quad (1)$$

where, the K function is called a kernel and that is all that is needed to maximize the distance to make decision boundaries to classify price sequences. The kernels used in this work are explained on Section 3.4.1 – Support Vector Machine Module.

After the *SVM model* is trained with training data set of price sequences, it can classify if a given sequence of prices is an *Uptrend*, *Downtrend* or *Sideways*. It is important to remember that training data is fully independent from the test data.

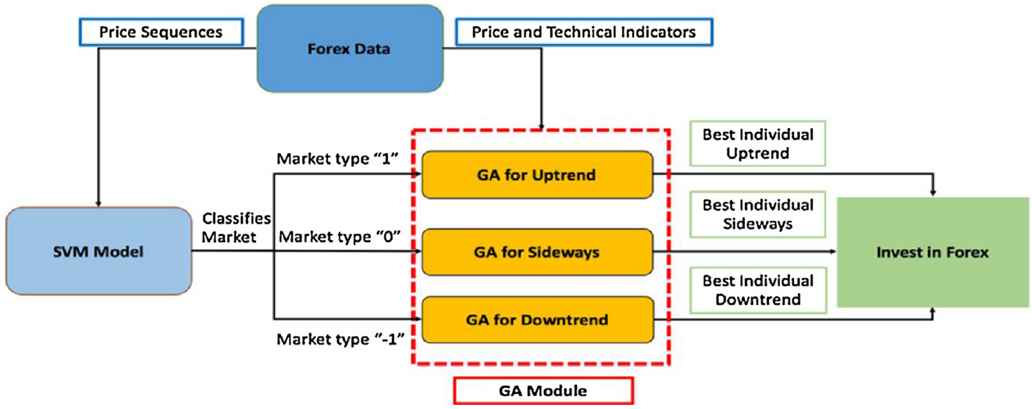

Fig. 1. Overall Architecture for Proposed System.

Depending on the type of market the price sequences enter in a different *GA* algorithm as it was explained before. The objective during the training set is to optimize F(X) of every genetic algorithm for each type market, where the fitness function F(X) is given:

$$F(X) = ROI(X) = \frac{Returns(X) - Investment(X)}{Investment(X)} \tag{2}$$

Where $X = (x_1, x_2, \ldots, x_n) = RSI, SMA, ROC, EMA, MACD, MA, WMA$, and F(X) is the *Return on Investment* for a specified time. The objective is to maximize F(X) throughout different price sequences, to obtain the best return on investment in each type of market.

The three *GAs* use the same mathematical logic throughout the different logic operators of *selection*, *crossover* and *mutation*. The *SVM model* decides which type of market is more appropriate to "activate" the respective *GA* and therefore used to maximize its F(X). The numerical example on Table 3, describes how training data with a 1000 price sequence behaves in the hybrid system. The window size represents the size of the price sequence and the number of features used in the *SVM* Algorithm, i.e., in this example it is shown that the windows size is always 100 prices, but it is possible to use different sizes of price sequences, depending on the timeframe intended to analyse. For this example, the SVM classifies the last 100 prices, and then the respective *GA Algorithm* is "activated" to be used on the next price, e.g., for price *t* the SVM classifies *t-100* prices and then, the GA for price *t* is used as it can be seen on Table 3 for different entry points on a 1000 price sequence training data. Note that every price is a possible entry point, being only showed in Table 3, a small group of entry points to make the example more perceptible.

Using the training method shown above, it is possible to train three different *GAs* to optimize three different populations (one fitness function F(X) for each *GA*) that will be further used in the test set accordingly, e.g., if the market classifies a specific period as an *Uptrend* market, then the trained *Uptrend GA* is activated and used to optimize F(X), as it is shown by a numerical example of a test set in Table 4.

The interaction between the SVM and the GAs is applied on the test set as well, and it is important to say that it is possible to shift from different types of GA at any entry point because the SVM can classify a different type of market at a given price. This approach allows the hybrid system to shift in a more dynamic way, than a static GA. Finally, it is important to refer the mathematical model of the *SVM Algorithm* described can be applied whether the features used are price sequence or technical indicator values. This premise is important to understand that the same logic was applied when comparing the results of using price sequences or technical indicator values as features, to obtain the best type of market classification.

3.2. *User interface layer*

This layer is the upper layer and provides the program interface for the user. The first step was to work with a Python IDE called *Spyder* [26] to develop the necessary features to make it possible to implement a graphical interface. This interface has input data, where the user decides what currency pair wants to invest, in this work, more specifically, the EUR/USD, and also chooses the GA parameters, i.e., how many technical indicators and which ones wants to insert in the *GA*. The interface returns the output data where it is included which strategy, technical indicators and technical rule will be applied, and the best level of leverage to be used. It can, also, show a comparison with the *B&H* and *S&H* strategy, so it is possible to analyse the two different approaches.

3.3. *Forex data layer*

3.3.1. *Price processing module*

The simulations use EUR/USD data since the year 2003 until 2016. The price points used for this work were downloaded from the Dukascopy website [27], to obtain the price sequences used in the price processing module. This layer is also responsible to calculate each technical indicator, to create the input to insert in the GA. Retrieving data is a very important step and has to be made very carefully so the data retrieved does not become damaged. Since the GA needs past information to make predictions about the future, it is only natural to conclude that the more information it retrieves the better. Finally, the system has to be robust and since the index is updated on a daily basis, it has to be possible to adapt and correct the data if it ever becomes corrupted. To provision the work's database (DB), the software relied on a python library called *Pandas* [28], in order to improve the robustness of data and to reduce the access time when a request to the DB was made. *Pandas* library was designed to be faster and more organized as a DB when compared to simple array structures or *Python* types such as "lists" or "matrixes". The technical indicators were calculated using the *FX data*, in order to be used in the optimization layer. The Technical Indicators were calculated through a *Python* library called *ta-lib* [13,29], making it possible to only calculate them once, in order to make the software faster and less heavy on computation. The technical indicators used in this work and its parameters are shown in Table 5.

3.3.2. *Technical rules module*

One of the main difficulties is to choose the right strategy to invest, whether is the right mix of technical indicators, its thresholds or even the parameters that are chosen to calculate the technical indicators, small variations in one of these decisions can change the rule to buy or sell in instants. The first assumption to

Table 3
Training data set numerical example.

SVM Window	Action between modules	Price Sequence(t)	Type of Market	GA Type	F(X)
$[t_1; t_{100}]$	SVM classifies price sequence to define the type of market to activate one of the GAs	*GA for* t_{101}	Uptrend	Uptrend GA	1.18
$[t_{101}; t_{200}]$	SVM classifies price sequence to define the type of market to activate one of the GAs	*GA for* t_{201}	Uptrend	Uptrend GA	1.31
$[t_{201}; t_{300}]$	SVM classifies price sequence to define the type of market to activate one of the GAs	*GA for* t_{301}	Downtrend	Downtrend GA	1.20
$[t_{311}; t_{410}]$	SVM classifies price sequence to define the type of market to activate one of the GAs	*GA for* t_{411}	Downtrend	Downtrend GA	1.28
$[t_{411}; t_{510}]$	SVM classifies price sequence to define the type of market to activate one of the GAs	*GA for* t_{511}	Sideways	Sideways GA	1.24
$[t_{531}; t_{630}]$	SVM classifies price sequence to define the type of market to activate one of the GAs	*GA for* t_{631}	Sideways	Sideways GA	1.32
$[t_{601}; t_{700}]$	SVM classifies price sequence to define the type of market to activate one of the GAs	*GA for* t_{701}	Sideways	Sideways GA	1.37
$[t_{771}; t_{870}]$	SVM classifies price sequence to define the type of market to activate one of the GAs	*GA for* t_{871}	Uptrend	Uptrend GA	1.32
$[t_{801}; t_{900}]$	SVM classifies price sequence to define the type of market to activate one of the GAs	*GA for* t_{901}	Uptrend	Uptrend GA	1.43
$[t_{901}; t_{1000}]$	SVM classifies price sequence to define the type of market to activate one of the GAs	*GA for* t_{1001}	Downtrend	Downtrend GA	1.56

Table 4
Test data set numerical example.

SVM Window	Action between modules	Price Sequence(t)	Type of Market	GA Type	F(X)
$[t_{1101}; t_{1200}]$	SVM classifies price sequence to define the type of market to activate one of the GAs	*GA for* t_{1201}	Uptrend	Uptrend GA	1.27
$[t_{1221}; t_{1320}]$	SVM classifies price sequence to define the type of market to activate one of the GAs	*GA for* t_{1321}	Sideways	Sideways GA	1.37
$[t_{1351}; t_{1450}]$	SVM classifies price sequence to define the type of market to activate one of the GAs	*GA for* t_{1451}	Downtrend	Downtrend GA	1.35

Table 5
Parameter intervals used for each technical indicator.

Technical Indicators (TI)	Intervals for TI parameters
RSI (Momentum Oscillator)	Period: [5, 30]
ROC (Momentum Oscillator)	Period: [15, 30]
SMA (Trend Following)	Period: [10, 100]
EMA (Trend Following)	Period: [10, 100]
WMA (Trend Following)	Period: [10, 100]
MACD (Trend Following)	Fast Period: [10,20] Slow Period: [20, 35] Signal Period: [5,10]

Table 6
Decision Rules used for voting system.

Technical Indicator	Trading Rule	Decision
RSI	• RSI < 30 • RSI > 70	• Buy Rule • Sell Rule
ROC	• ROC > 0 • ROC < 0	• Buy Rule • Sell Rule
SMA	• SMA < price • SMA > price	• Buy Rule • Sell Rule
EMA	• EMA < price • EMA > price	• Buy Rule • Sell Rule
WMA	• WMA < price • WMA > price	• Buy Rule • Sell Rule
MACD	• MACD > 0 • MACD < 0	• Buy Rule • Sell Rule

make is quite simple, there is no better indicator, only a good combination of them will return a good decision rule. The technical rule applied was a voting system where a combined set of technical indicators voted, depending on specific rules that are explained in Table 6.

3.4. Optimization layer

As it was said before, this layer is decomposed into two different modules. The first one, is the *SVM module*, its objective is to identify different market strategies, i.e., the *SVM algorithm* classifies the currency pair historical data in three groups, *Bullish Market* (uptrends), *Bearish Market* (downtrends), and *Sideways* (No trend). This way, the *GA* trains its populations to specific types of markets, depending on the trend. This feature is expected to return a better performance to the *GA* and increase its success rates.

3.4.1. Support Vector Machine module

This module is composed by several components, and the first step is to describe the features used in the *SVM algorithm*. The project relied on *scikit learn* framework in order to assure reliability in terms of computation and stability, being that the focus of this work is not the implementation of an *SVM* itself. Although the *SVM* classifier is the same, there were two possible approaches to choose the type of features used. The first one was the use of technical indicators as features and the second one was the use of price sequences, in order to classify the type of market. In the first case the features used were the six technical indicators calculated in the *FX Data Layer*, in the second case, sequences of 100 prices were used and each one of the prices worked has a feature of the *SVM*. Table 7 shows an example of the two types of features used in each scenario.

The *SVM algorithm* takes an *X* array of size *n_samples* and *n_features* like: *X* = [*n_samples*, *n_features*] of training samples and a *Y* array of class labels of *size n_samples* like: Y = [*n_samples*]. In this work's scenario the labels correspond to the market types, where a *bullish market* represents a “1”, a *sideways* market represents a

Table 7
Features used for the SVM Model.

Feature type	Features example
Technical indicators	Features = ["RSI", "ROC", "EMA", "SMA", "WMA", "MACD"]
Sequence of Prices	Features = [Price1, ..., Price100]

Table 8
Hyper- parameters of the SVM Model.

Hyper-Parameters	Values
C	[1,10,100,1000]
Gamma	[0.0001, 0.001, 0.01, 0.1, 1]
Kernel	• Rbf • Polynomial, n = 2,3,4 • Linear

"0" and a *bearish market* represents a "−1". The classification of the market is then calculated using all the classifications of the predictions, i.e., The classifier predicted 100 price sequences in a sliding window, returning 100 samples that were classified as "0", "1" or "−1". The calculations are made with a *Weighted Moving Average (WMA)*, that analyse the average along a certain period, making the decision of which type of market is based on the result of the *WMA*. If the result of the WMA is bigger than 0.5, the market is *Bullish*, if the WMA is lower than −0.5 the market is *Bearish*, otherwise is *Sideways*. Moreover, the WMA computes an average where the last prices have more importance, i.e., weight that influences the moving average.

The *X* and *Y* are then divided into *X_train* and *Y_train* for the training set and *X_test* and *Y_test* for the test set. First, let's focus on the training set, where the *SVM* "learns" from the data in order to classify the market in the test set. To optimize the performance of the *SVM* training model it is necessary to tune its *hyper-parameters*. A *Grid Search* algorithm was implemented to choose the best parameters for the *SVM* classifier, where an exhaustive search of combinations was made through a specified subset of the *hyper-parameter* space. The *SVM* takes three *hyper-parameters*, where *C parameter* is the parameter for the cost function, which controls the influence of each individual support vector and involves trading error penalty for stability, i.e. *C parameter* controls the cost of misclassification on the training data, and it can influence the variance of the solution (large *C* value makes the cost of misclassification high, *hard margin* and low *C* value makes the cost of misclassification low, *soft margin*). *Gamma* is the *RBF kernel* parameter that handles non-linear classification, i.e., it influences the trade-off between the size of the street (from the SVM's "Widest Street Approach) and the margin for misclassification of a label. The *hyper-parameters* values used in this work are shown in Table 8.

The training solution is then saved to a file in order to make the classification of the market faster, and to avoid an overkill solution where the *SVM* model is trained every time the software is run. After the training set is complete the algorithm applies the *cross-validation* method of *K-Fold* to assure that the trained solution is not biased. The method divides the DB in *K* parts and trains for some of them and then tests the others with the trained model. It is important to state that the training periods used for the K-fold are independent from the testing period used, otherwise the solution would be overfitted. Fig. 2 shows a schematic representation of a 5-Fold strategy example.

The algorithm takes sequences of 100 prices, classifies them and then sends the answer to the *GA module*, so that the *GA* can know which environment it is on. Fig. 3 describes a pseudo-code that shows how the *SVM model* was trained, and then used in the *GA* later on. The pseudo code shows the several steps that were taken to create the classifier, and shows the two approaches that were tried as features in this work. Both approaches were tested with the same logic and code.

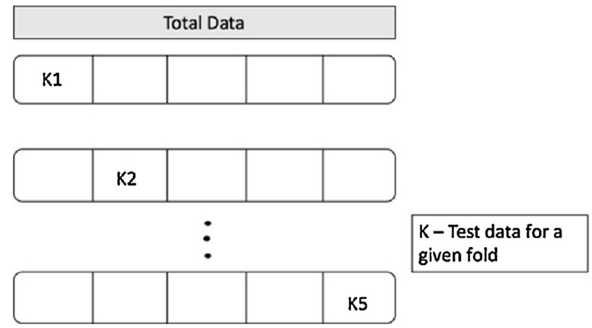

Fig. 2. K- Fold strategy example.

Table 9
Definition of the Metrics.

Metrics	Definition
Precision	$\frac{True\ Positives}{True\ Positives + False\ Positives}$
Recall	$\frac{True\ Positives}{True\ Positives + False\ Negatives}$
Accuracy	$\frac{True\ values}{Sample\ Space}$

Admitting that only the correct evaluations are important as a metric of quality seems to be incomplete. For the SVM's case studies, it is necessary to evaluate the quality of a classification using the metrics, such as precision, recall and accuracy. Before describing the measures, it is necessary to explain what are the *True Positives*, *False Positives*, the *True* Negatives and the *False* Negatives. While *True Positives* are the number of items correctly labelled as belonging to a given class, *False Positives* label a number of items as a given class when in fact the item does not belong to that class. *True Negatives*, are the items that are truly rejected when labelled to a given class, e.g., an item that is class "1" gets rejected when the hypothesis is to label it as class "0". *False Negatives*, are items that were not labelled as belonging to a given class but should have been. *Precision* is a measure for result relevancy, i.e., the fraction of selected items that are relevant and by its definition is the fraction between *True Positives* and the sum of *True Positives* and *False Positives*, as is described in Table 9 *Recall* is also a measure of relevancy but more specifically the fraction of truly relevant results returned, i.e., the measure of how many relevant items are selected as a *True Positive*, and is defined as the number of *True Positives* over the number of *True Positives* plus the number of *False Negatives*. *Accuracy* is the measure between the number of items that were labelled correctly (True Values) and the sum of all the items.

3.4.2. Genetic Algorithm module

In this module, the training set, takes advantage of three types of chromosomes. Although they are structurally the same, they were trained depending on the type of market the *SVM Module* classified.

3.4.2.1. Chromosome representation.

The chromosome structure is an array of floats where each gene of the chromosome is the parameter for each technical indicator and its weight accordingly. Each gene is transformed to an integer for each parameter of a technical indicator, this way making the construction of the chromosome structure faster than if the chromosome was a combination between integer and float array. The values to fill each chromosome, i.e., to create each individual are set with a random function with interval constraints depending on each technical indicator. The weights are a random function that returns a float

```
features =["Price1", "Price2", ... , "Price100"] #Approach of features type 1
features=["RSI", "SMA", "WMA", "EMA", "MACD", "ROC"] #Approach of features type 2

Database:
    DB= Prices_from_EUR_USD #Sequence of Prices from EUR/USD
    DB =Technical_indicators#Calculated in the Technical Rule Module


X= array(DB[features]) #array with the features chosen from the selected DB
Y=DB["labels"] # Labels from a DB that is classified by the user to train the SVM Model
for DB:

    split X and Y in train_set and test_set

    Parameters=[["rbf", "poly", "linear"], [0, 0.01, 0.001, 0.0001], [1, 10, 100, 1000]] #hyper parameters [kernel, gamma, C]

    find_best_parameters:
        do grid_search(Parameters)

    classifier=train_DB(X, Y) #function where the model learns and creates a classifying model for the SVM

    apply 5-Fold(X) #Cross Validation Method
```

Fig. 3. Pseudo-code for SVM Model.

RSI Weight	SMA Weight	EMA Weight	WMA Weight	MACD Weight	ROC Weight	Leverage level
RSI parameter	SMA Parameter	EMA Parameter	WMA Parameter	MACD Parameter	ROC parameter	

Fig. 4. Chromosome Representation.

between zero and one. The chromosome's objective is to optimize two sub-strategies, while the first one optimizes the importance of each technical indicator, meaning that the GA searches for the optimal solution in terms of finding the optimal weights for each TI, the second sub-strategy optimizes the parameters of each technical indicator, i.e., parameters need to be dynamic in order to adapt to each sequence of prices, e.g. a RSI with a 14 period price sequence can tell the trader to sell the currency, when in fact the trader should have bought it if the RSI period was 25. The truth of the rule derives from each price sequence, and thus, being important to optimize the TI parameters also. The structure of the chromosome is shown in Fig. 4. Red section indicates the weights for the chosen TIs, and the green section indicates the parameters for the second sub-strategy that is optimizing each TI. Finally, the leverage level of the red section optimizes the amount of leverage that is used.

The chromosome built in this work uses 15 different genes, where 8 are technical indicator values, and the other 6, the technical indicator weights, and a last gene with the possible values of leverage levels. The amount of possible combinations that combine a full chromosome are quite wide. An estimation of the computational complexity to this work's problem should be higher 2,05 × 1020, that are the number of possible combinations to create a chromosome, where each gene range is multiplied. This amount of possible combinations can provide to this system, a wide number of possibilities to find an optimal solution to invest in EUR/USD.

3.4.2.2. Evaluation function. The fitness function used was the *Return on Investment* (ROI), calculated as the *Returns-Investment/Investment*. In addiction to the evaluation function of the GA, it is important to refer another evaluation metric that is used to analyse the algorithm exposure to risk, named *Drawdown*. This second evaluation metric is given by *Local Max. price – Local Min. price* of the investment return.

The *Genetic Algorithm* function is single objective and tracks to maximize the approaches chosen above. To calculate both objectives, the fitness function relied on the *Technical Rules Module* to make the decisions according to the TI. The trading rule decides on a voting system, of whom ever won by majority would perform the trade, i.e. if the TIs voted for "Buying" Rule with a *K* % majority over the "Selling" Rule, then the *GA* would perform a "Buying" rule instead of a "Selling" rule, and vice-versa. If a majority was not reached, then the *GA* entered the "Do nothing Rule" and waited until the next entering point. The following equations show, how the voting takes place, being *K* the parameter that defines the majority thresh hold.

$$Voter\ Buy = \sum_{i=0}^{n=6} w_l \tag{3}$$

$$Voter\ Sell = \sum_{i=0}^{n=6} w_l \tag{4}$$

$$Voter\ Buy > K * Voter\ Sell \tag{5}$$

$$Voter\ Sell > K * Voter\ Buy \tag{6}$$

After the voting system takes part, the investment module inside the evaluation function opens a "Long" or "Short" position depending on the voting, if there is no "Opened" position, then the voting system decides to enter the market if the voting rules a buying or selling position. If the voting rules a sideways then it does not enter the market. When there is an "opened" position, then the position maintains opened until the voting system rules otherwise, i.e., if the position is long, then it stays long until the voting system rules a sideways or a short position.

3.4.2.3. Selection. After defining the chromosome representation, it is necessary to define the genetic operators used in the algorithm. The selection method chosen was the *Tournament Selection* method to define which individuals were to be selected for the next generation. The *tournament selection* method involves running several tournaments between individuals among the population, and the winner of each tournament is selected to perform *crossover*. The bigger the tournament size the less chances a weaker individual has to perform crossover. Initially the tournament size was set to 3 and selection probability is 0.5. Only when the *hyper-selection* [24] was activated that the tournament size was set to 5 to increase the selection pressure in a population. *Hyper-selection* was chosen to be applied in this work, because some periods of the *FX market* can be more difficult than others because of an ill-defined market tendency, thus, making the *returns on investment* (ROI) harder to increase its value. Said that, the selection pressure is increased temporarily to decrease the probability of weaker individuals to perform *Crossover*. This can lead to an over fitting, so it is important to keep in mind that the *Hyper-Selection* should only be temporarily.

3.4.2.4. Crossover. The method that was most suitable to solve this problem were the *Two-Cut-Point crossover*, where two chromosomes exchange two genes between themselves. This algorithm was chosen over the *One-Cut-Point* because the *Two-Cut-Point*, slowed the early convergence and thus avoiding faster over fitting in the final solution. This *crossover* method consists on picking two random points of the parent chromosomes, then, the genes between the two points, and the ones outside the two points are swapped, originating *offsprings*.

3.4.2.5. Mutation. Individuals may go through mutation in the end of each generation and in this work the mutation that one individual performs is derived from a *Gaussian Distribution*, although its limits are imposed by the gene mutated must be higher than zero to be considered valid. The mean and standard deviation values are linked to the parameter of each gene. Being the mean the parameter and standard deviation 10% of the parameter value. *Mutation* is a random process that depends on the mutation rate and on the probability of each individual to get a mutation. Also in these studies, one introduced *Hyper-Mutation* [23], and according to changes in the environment the mutation parameter will go up temporarily in order to increase the diversity in a given population.

3.4.2.6. SVM and Dynamic Genetic Algorithm together. Like stated above, this is one of the key issues for this work, where it is implemented a *memory system*, relying on a *SVM* classifier to identify which type of market and then store the information along side an individual that was specifically trained for that situation. This memory system works as an *Associative Memory* for the Dynamic Problem of the GA. The goal is to train three different populations for the three types of markets, and then used them during the investment sessions, taking advantage of knowing the type of market they are in, i.e. knowing the environment can increase the performance of each population, and the memory approach proved to be efficient, thus, it is not needed to have a new population every time the environment changes.

This solution provides three approaches to solve Dynamic Problems in the Genetic Algorithm, which are:

- *Associative Memory* – The *SVM* provides the information of the markets(environment) that is saved along side the best individuals of each population, providing a solution where the GA uses this information to adapt to each market (environment) according to the continuous change of FX price sequences.
- *Hyper-Mutation* – This method provides a temporarily raise up of the mutation rate, in order to introduce more diversity in the GA, to avoid local maximums. This method makes the GA more adaptable to find new optimal solutions instead of being stuck in a solution that may have been good for a given price sequence, but has gotten worse every time that price sequence is updated.
- *Hype-Selection* – This third approach also provides a temporarily condition to the GA but this time is to increase the selection pressure, where the quality of the solution is obligated to increase. This approach is very helpful in the way that it makes possible to increase the quality of the solution without becoming over fitted because it is only a temporarily solution.

3.4.2.7. Implementing leverage. Leverage is explained in section 2.3 and as it was said before it can bring great returns but also great amounts of risk. To limit the risk exposure, one made the decision to limit the leverage multiplier to a maximum value of 10.

This financial tool was applied as the following procedure:

- If the trading rule and the *SVM* classifier are combined, i.e., if the trading rule votes a "Buying Rule" and the *SVM* classifies the market as *Bullish*, or if the trading rule votes a "Selling Rule" and the *SVM* classifies the market as *Bearish*, then leverage is activated; The second step to activate leverage is defined by the *Moving Average* of the "High" values combined with the *Moving Average* of the "Low" values, of each closing price e.g. If the *Moving Average* of the "High" and "Low" values cross the respective "High" and "Low" values to an uptrend when the trading rule votes a "Buying Rule", or to a downtrend when the trading rule votes "Selling Rule", then leverage is applied. To better understand the idea, Fig. 5 shows a chart with entry points where leverage is activated;
- The amount of leverage used, is part of the chromosome structure and when the leverage is activated the amount of leverage goes between 2 and 10;

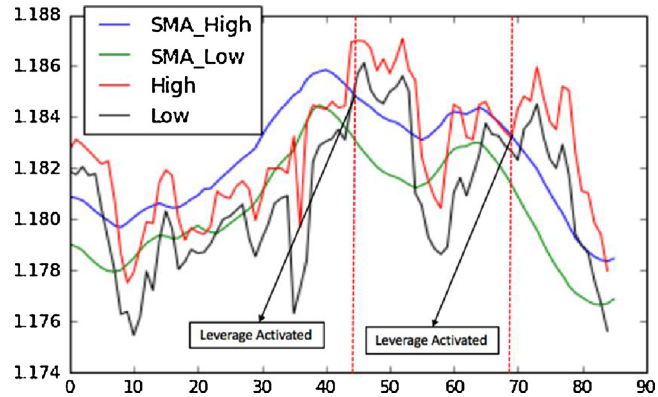

Fig. 5. Entry Points where Leverage is activated.

Table 10
Case study configuratios.

Parameters	Value
Market	Forex – EUR/USD
Training with sliding Window Period	01/01/03–01/01/2015
Real Test Period	02/01/2015–02/03/2016
Initial Investment	100 000 €
Short Selling and Long Buying	True
Leverage	[2,10] in the proposed approach
Number of executions	50

4. System validation

In this section, the system validation is presented, where a case study is ilustrated with the results obtained from the experiences of developing a hybrid system with an *SVM* and a *GA*. The *Backtesting* approach is based on a *Sliding Window* for training sets between 2003 and 1st of January of 2015, and then a real test period where the approach had never seen the data before between 2nd of January of 2015, until 2nd of March of 2016. The dataset is from the forex broker Dukascopy and is available for download in demo accounts [27]. Table 10 shows the case study configuration.

4.1. Case study – description

In order to include the most possible data for different types of trends in this work, the EUR/USD *Forex market* was set between 1st

Table 11
Best parameter configuration for the SVM Model.

SVM Parameters	Values
Model: SVM Classifier	–
Kernel	RBF
C	10
Gamma	0.001
Price Sequence Length	100
Number of executions	50

of January of 2003 and 1st of January of 2015. This approach is compared against *Random Walk*, *B&H*, *S&H*, and a *Static GA* approach.

- ***Buy & Hold/Sell & Hold*** – This two classical approaches take the believe that the market always takes a trend and it is not possible to predict market fluctuations relying on past data, so, whether the market is bullish or bearish, traders take the position and holds until he changes his opinion.
- ***Random Walk*** – This theory relies on the fact that the market is random, and thus, making random investments as a consequence. This approach decides in a totally random way, if it takes a short or long position and if a position is opened or closed.

Moreover, the *Proposed System* is compared with the work done by Hirabayashi et al. [2], due to the fact that it is based on Forex Market, although this work is based on Japanese Yen(JPY) against US Dollar and EUR, thus making the behaviour of this specific market different from the EUR/USD, it is an important benchmark to compare the behaviour of the returns and in the implementation of leverage. Also, the *Proposed System* is compared with the work done by Sermpinis et al. [7], in terms of *Drawdown*, since there are no charts showing the evolution of ROI. For both works, actual results on ROI are not reasonable to compare, because the Testing Period is different, and the type of market is not always the same.

Finally, real life transactions are a bit more difficult to calculate in this work, because each broker decides its own cost transaction. The transaction costs are usually low comparing to other types of markets, thus the results do not diverge considerably when applying an average transaction cost of 0.02% of the transaction made.

4.2. Parameters setup

The parameters are divided between the two modules, the *SVM* and the *GA*. In this approach it is allowed to perform long and short selling, including leverage, and also, this approach takes advantage of technical analysis only. As for the SVM module, the configuration parameters for the *SVM classifier* are shown in Table 11. The *SVM* parameters were selected out of 50 experiments set, where these were the values that presented the best average performance.

The results obtained for the parameters, regarding the choice of the kernel used and its hyper-parameters, show that the solution does not indicate over fitting in the SVM model, because the *C* and Gamma obtained are not too "optimized", i.e., in the C interval of [1,10,100,1000], the value obtained is not too high, as for the Gamma interval of [0.1, 0.01, 0.001, 0.0001] the value 0.001, although is a bit closer to the limit is still a reliable value to use. When comparing this solution against a *Linear* kernel that has *C* parameter equal to 1000, this solution for parameter setup is the most appropriate to design the *SVM* model.

As for the *GA* configuration, Table 12 shows the different methods applied in this process, as well as, the value parameters for the proposed methods. A sliding window approach was used to perform training and test sets. The stopping criteria used in this work is inspired on Hirabayashi approach [2], where the generations increased limitless as long as the *GA* finds a better solution after each generation, if there is no better solution at the end of 30 generations then the *GA* stopped there.

Table 12
Configuration of the GA.

Parameter	Value
Population Size	100
Mutation Rate	• Hyper-Mutation: 0.4 • Static Mutation: 0.2
Generations	Stopping Criteria
Selection parameter	• Tournament size (TS): 3 • Hyper-selection: TS - 5
Crossover	• Two- Cut – Point • Crossover prob.: 0.8
Random Immigrants	50%

In order to better understand the tuning procedure of the GA, Table 13 shows the different configurations used to perform the parameter tuning method of the GA.

To better understand what were the best parameters for tuning the GAs, several tests were made regarding variation in population, mutation rate and number of generations. The baseline configuration is 100 Generations, 100 individuals in a population and 20% of Mutation Rate (MR). The Proposed Stopping Criteria (PSC) was introduced with the objective of increasing the adaptability to new solutions in the search space, rather than having a fixed number of generations. This solution is inspired on [2], where the generations increased limitless as long as the GA finds a better solution after each generation. In this work the generations can be limitless as long as the GA finds a solution with better fitness, but the number of generations without finding a better solution is set to 30 generations instead of 10. After a new solution is found, the number of generations is reset to zero and the process is repeated. The Stopping Criteria has the following procedure:

- Calculate fitness for best individual of present generation;
- If fitness is higher than the fitness from the previous generation, reset the number of generations to zero
- If fitness from the present generation is not higher than the previous one for 30 generations, then convergence is obtained.

Table 14 presents the results obtained for each configuration, where it is possible to conclude that configuration 2xPop-PSC slightly outperforms the PSC, but the performance of the algorithm is worse. The size of population was chosen in order to achieve a trade-off between the best solution and the computation speed of the algorithm. The results are aligned with the theoretical assumptions, because it is only natural that a population that starts with the double of chromosomes has twice the chances of finding a good solution for the problem. As it can be seen by Table 14, the configuration 2XMR provides promising results to use hyper-mutation temporarily, since this approach reaches higher Returns on Investment but presents worse Drawdown and a higher Standard Deviation. An interesting result is the 2xMR, where the average result is slightly worse than the PSC and 2xPop-PSC, but it returned better Max. Returns compared to the PSC and 2xPop-PSC. This may be linked to the fact that the Mutation Rate is higher, thus inducing the probability of discovering new solutions in the search space, that could lead to better results, when compared to the PSC or 2xPop-PSC configuration. Although this configuration may lead to better maximum solutions, it is not advisable to use this configuration as a baseline because, as it is possible to see by its Average Returns, the configuration is slightly worse when the diversity introduced is too high, i.e., too much diversity may cause the algorithm to lose its best chromosome because it suffered a

Table 13
Configuration of the GA for the tuning procedure.

Number of Generations	Size of Population	Mutation Rate(%)	Configuration
50	100	20	1/2 Gen
100	100	20	Base
200	100	20	2xGen
Proposed Stopping Criteria	100	20	PSC
100	50	20	1/2 Pop
100	200	20	2xPop
Proposed Stopping Criteria	50	20	1/2 Pop-PSC
Proposed Stopping Criteria	200	20	2xPop-PSC
Proposed Stopping Criteria	100	10	1/2MR
Proposed Stopping Criteria	100	40	2xMR

Table 14
Results obtained for each configuration.

Method	PSC	Base	$\frac{1}{2}$ Gen	2xGen	$\frac{1}{2}$Pop	2xPop	$\frac{1}{2}$Pop-PSC	2xPop-PSC	$\frac{1}{2}$MR	2xMR
Avg. ROI (%)	43.8	32.2	27.1	34.0	29.7	34.9	40.2	44.6	37.2	38.1
Max. ROI (%)	83.1	74.3	64.6	79.2	72.8	75.1	82.9	83.7	69.4	89.3
Min. ROI (%)	−6.1	−14.2	−13.3	−8.2	−15.0	−4.3	−9.3	−3.2	−2.4	−15.9
Std. Dev (%)	30.2	26.7	24.2	25.8	27.1	26.1	29.7	28.4	23.8	36.2
Drawdown (%)	10.3	9.9	10.6	9.6	11.2	9.3	10.2	9.1	7.9	14.3

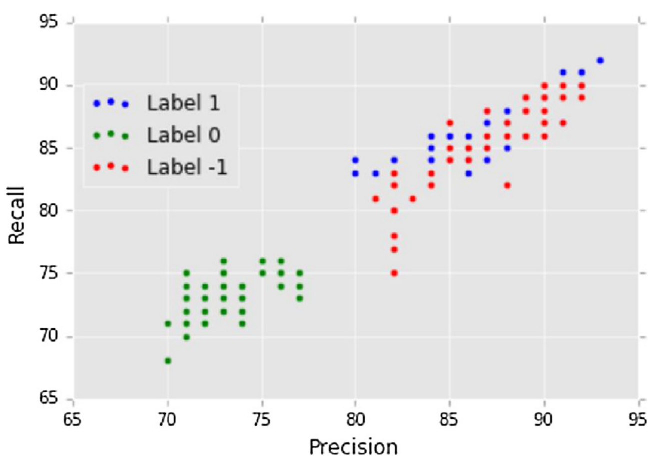

Fig. 6. Average metric results of each label with Price Sequence approach.

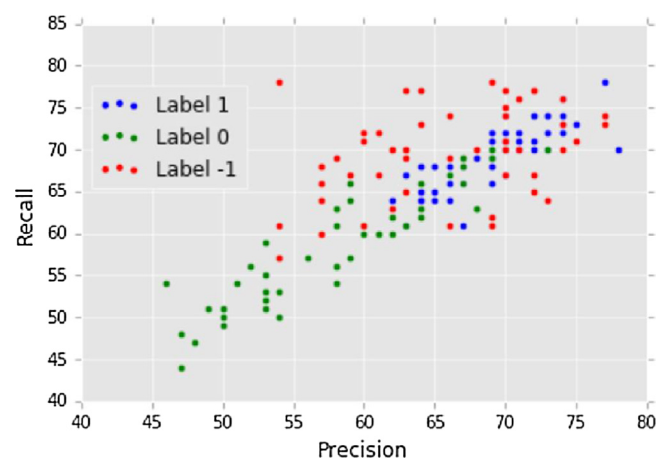

Fig. 7. Average metric results of each label for Technical Indicator approach.

mutation, nonetheless it is a good sign that it is possible to use hyper-mutation, temporarily, to discover better solutions in the search space, i.e., introducing diversity in extreme cases can reveal good solutions as a dynamic approach.

4.3. Case study – performance analysis

4.3.1. SVM model

The *SVM* algorithm was trained for 50 runs, and the data used to "teach" the model, goes from 2003 until 2015 with the sliding window approach in order to maximize the solution as much as possible. The experiment collected 4000 samples to train the *SVM model*. To use more than 4000 samples would require more computational power, and the results would not present a considerable improvement.

Fig. 6, describes a scatter plot for the results obtained by the metrics used in this paper during the test period for the *Price Sequence Approach* returned better results when compared to the *Technical Indicator Approach*, it is possible to see that the *Price Sequence Approach* can classify the three types of markets with better results on *Precision*, *Recall* and *Accuracy* [30,31], showing that this method is more reliable to use, further on, in the *Proposed System*. Although the solution shows a good performance, results indicate that label "0", i.e., the label that classifies sideways markets, it is worse than the others. A reason for this to happen, is that this label is more difficult to classify than the up or down trends, even for the human eye, because it has a bigger degree of subjectivity. Nonetheless, it is possible to see that, from the metric's results, it is clear that the *SVM model* can predict successfully the type of markets that are sent to the *GA*, showing promising results for the algorithm proposed in this work. Fig. 7 describes a scatter plot for the results obtained by the metrics used in this paper during the test period for the *Technical Indicator Approach*, showing that, the average metric results have a higher standard deviation, indicating that this solution may not be reliable. The reasons for this to happen is that this approach may need more samples than the first one to achieve good results, due to the fact that the correlation between the several TI's may not be an easy problem to solve. Finally, Tables 15 and 16 describe the results obtained for the SVM classification, for the price sequence and technical indicator approach, the average score of the metrics for each label, "1", "0", "−1", respectively, and the average score refers to the average result of the three labels together.

The first approach clearly outperforms the second one in every aspect, showing better metric results and average scores. Also shows, that the first solution proved itself to be more consistent throughout the executions, being more reliable to implement with the GA for the proposed system. Moreover, this solution is faster in computation than the second solution, when compared with the

Table 15
Results obtained for the SVM classification of the labels from the Technical Indicator approach.

Parameter	Precision	Recall	Accuracy
Avg. Score for each label (%)	[69.5, 57.7, 65.8]	[68.8, 58.6, 69.56]	65.34
Avg. Score (%)	64.3	65.65	64.90
Max. Score (%) for each label	[78, 73, 77]	[78, 70, 78]	74
Min. Score for each label (%)	[62, 46, 54]	[61, 44, 57]	53
Std. Dev. for each label (%)	[4.09, 7.23, 6.61]	[4.16, 7.13, 5.64]	5.63
Avg. CV scores (%)	[66.3, 63.4, 64.2, 66.2, 62.1]		

Table 16
Results obtained for the SVM classification of the labels from the Price Sequence approach.

Parameter	Precision	Recall	Accuracy
Avg. Score for each label (%)	[87,4, 73.2, 86,8]	[87, 1, 72, 9, 85, 2]	85.64
Avg. Score (%)	82	82	85.64
Max. Score (%) for each label	[93, 77, 92]	[92, 76, 90]	91
Min. Score for each label (%)	[80, 70, 81]	[82, 69,75]	76
Std. Dev. for each label (%)	[3.15, 2.04, 3.31]	[2.59, 1.8, 3.74]	3.934
Avg. CV scores (%)	[84.3, 82.5, 84.3, 84.1, 86.2]		

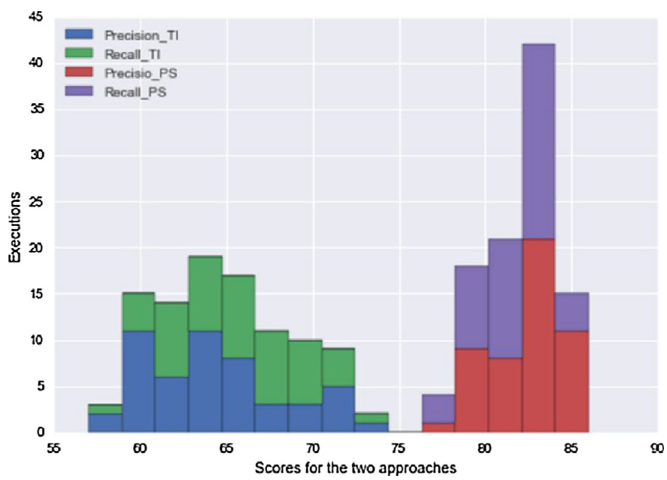

Fig. 8. Histogram showing the results for 50 executions comparing the two approaches.

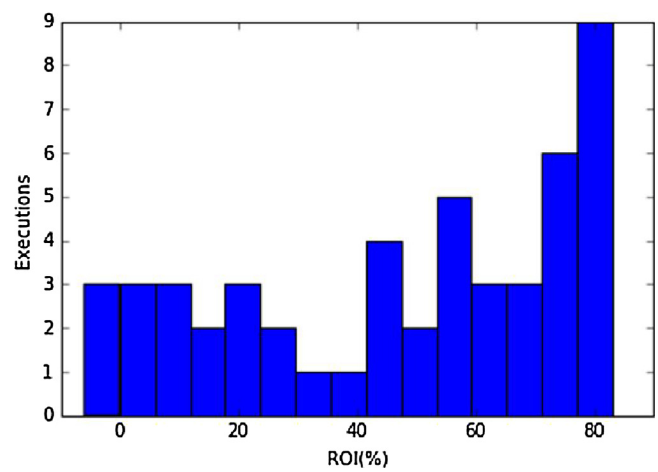

Fig. 9. Histogram with the results obtained for the test period, using leverage.

same number of samples. Fig. 8 shows a histogram that presents the results for the 50 executions, comparing the scores for average precision and recall of the 3 labels. The values obtained on *Precision* and *Recall* for the *Price Sequence* (PS) approach are positioned in the interval of [75, 85], while the values of *Precision* the *Recall* [30] for the Technical Indicator (TI) approach are on the interval of [55, 75], showing a disperse and volatile solution thus indicating that, the *Price Sequence* approach is better than the TI approach. The objective is to have both high *Precision* and *Recall* and the only approach that obtains that criteria is the *PS* approach. This criterion has to be accomplished in order to ensure that the model is not over fitted, i.e., having a good training model with good metric results and after in the test period returning a lot of false positives and false negatives.

4.3.2. *Proposed System analysis*

4.3.2.1. Dynamic Genetic Algorithm vs Static Genetic Algorithm. The following section intends to study the *Dynamic Genetic Algorithm* approach. The experiment is made within 50 executions and in the same train and test period of the *Static Genetic Algorithm* approach. Moreover, this section, intends to prove that some dynamic criterions (there are many that can be considered), shown in the literature, are compliant with this financial problematic, more specifically in this section, the following criterions are addressed and explained in detail further on:

- Predictability;
- Cyclicity;
- Visibility;
- Time-Linkage;
- Stability;
- Diversity Measures;
- Adaptability;

Fig. 9 shows a histogram with the results obtained for the test period, using leverage from the chromosome structure. As it is possible to see, the *Avg. Proposed System* shows steady growth through time and does not present considerable volatility during the evolution of the chart. The *Avg. Proposed System* obtained a *ROI* of 43% and the *Best Proposed System* obtained a score of 83%. The histogram shows that the executions may vary more than expected. This may be related with the fact that the population is not big enough, compared with the size of the chromosome, to evolve to an optimum solution that is identical in every execution, nonetheless, the 50 executions show that in 94% of the executions the *Proposed System* obtains positive ROIs. Fig. 10 shows that the *Proposed System* behaves with great stability, and as it was said before, it has a steady growth. The best solution within the 50 executions has a little bit more volatility, but it is still possible to see a steady growth, only with bigger variations. This is normal due to the fact that one line takes one executions only, and the other is the combination of the 50 executions.

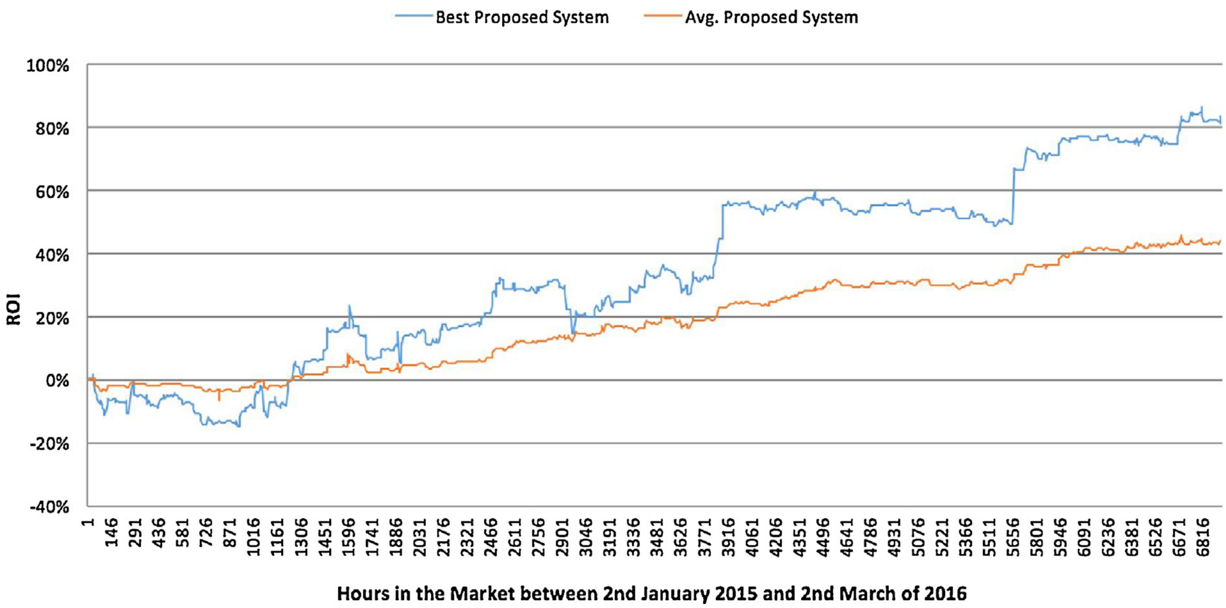

Fig. 10. Results obtained for the Best and Avg. Proposed System with the DGA approach.

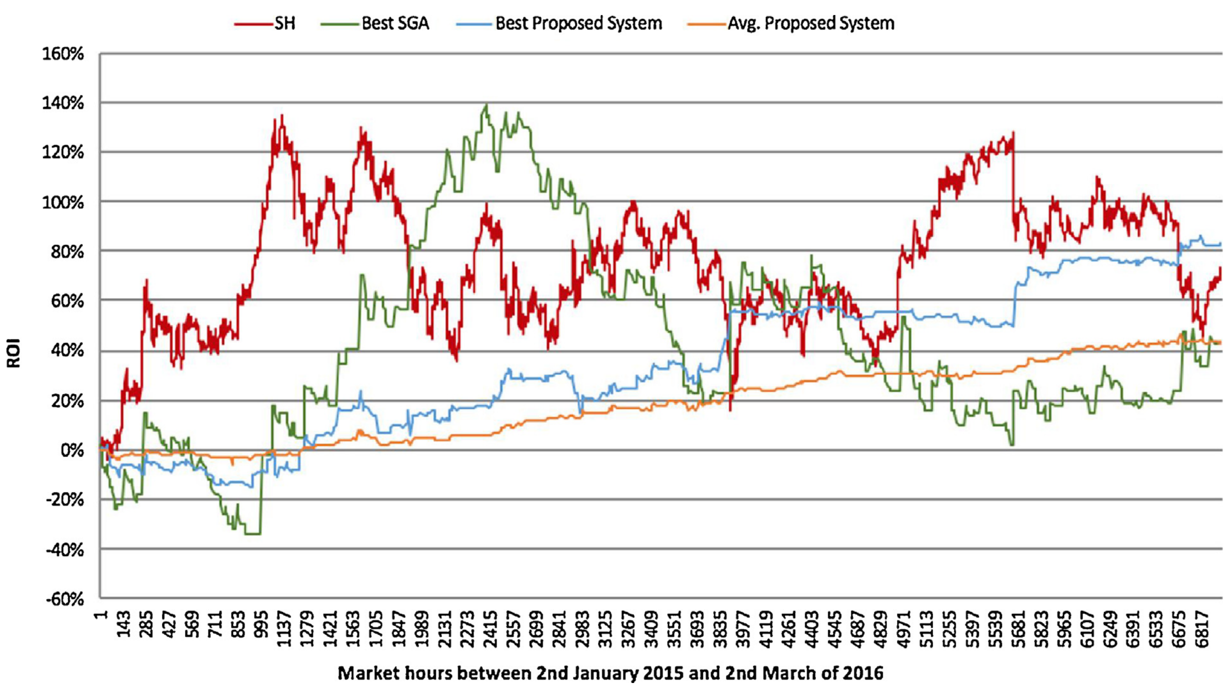

Fig. 11. Returns with leverage obtained during test period for the different approaches.

Before comparing the Static and the Dynamic approach, it is important to evaluate the criterions shown above, in order to evaluate the behaviour and how the proposed system reacts to the FX market. The following points discuss how the Proposed System reaches those criterions:

- **Predictability** – Although this work does not propose any predictive method like *Hidden Markov Models* or *Naive Bayes* (predictive methods that are usually seen in the literature), the proposed system shows that in the 50 executions, the losses seen are not considerable, even more, the solution presents a steady growth, where, in 57% of times, the Proposed System has profitable positions. By using *Adaptive* approaches (*Hyper-Selection* and *Hyper-Mutation*) and the *Memory* approach relying on the *SVM* classifier, made the *Proposed System*, quite adaptable to new environments, despite of the fact that this system does not contain any "true" predictive method. The *SVM* classifier is used to classify the present hour, based on a time-window of 100 h, then, the result of the classification is the prediction for the next hour, based on the idea that the type of market remains unchanged in the next hour.
- **Cyclicity** – The *Proposed System* takes advantage of the cyclicity in FX markets, by using the *SVM classifier*, that detects in which type of market the investor is inserted, then, the *Associative Memory* approach used in this work, stores the information (population and environment) about the different markets that are trained by the three *GAs*, in order to use them when the market changes, i.e., the *GA Module* optimizes three different *GAs* depending on the

classification of the *SVM* and then stores the information about the population and the environment trained, so that it can be used when the market repeats itself, e.g., let us imagine that the market is *Bullish* (Uptrend), in that case, the *GA* selected is the *Uptrend GA* that has a population that was specifically trained for Uptrends, if the market starts to change and the *SVM* classifies it as a *Sideways*, then the *GA Module* shifts the *GA* for a *Sideways GA*. This characteristic allows the *Optimization Module* to track *Cyclicity* through time, and can shift the populations according to the type of market, if the market is bullish and then shifts to a *Bearish* (downtrend) market and after that returns to a bullish market again, the *GA Module* can adapt and take advantage of an already trained population for that specific environment;

- **Visibility** – There are many ways of detecting changes, whether is in the environment itself or in the performance of the *GA*, the *Proposed System*, as it was referred before, can detect changes in the environment through the *SVM* model, although it is almost impossible to detect sudden changes, that were also referred in previous sections, like important news or press conferences from important politicians or central banks from the world. These sudden changes are impossible to predict when the algorithm only aims to analyse prices and the TIs. So to conclude this idea, the *Proposed System* does not detect those sudden changes, but can detect changes in the environment when its trend changes. As for the *GA* performance, the *Proposed System*, tracks the *ROI* during the investment sessions, and if the performance decreases to losses over 0.5% of the *ROI*, the *GA* closes the position and starts a new evaluation from that point without having to reset the algorithm. This measure proved to be very helpful to control the losses and the *Drawdown* in Fig. 10, showing a more stable and adaptable solution.
- **Time-Linkage** – During the several experiments and after the careful study made to this problematic, it is possible to assume that this problematic does not present a time-linkage characteristic, i.e., actions made by the selection criterion do not influence the environment. The FX market is completely independent from an action of a single investor, and the proposed system does not influence the environment whatsoever. The only theoretical way of this solution influencing the market is that if the great majority of traders used it. That option seems far way from reality, so it is automatically put aside. The only real influencers, in the short term, are the central banks, due to the fact that their news influence the great majority of traders, thus making sudden and great variations in the FX market.
- **Stability/Robustness** – The *Proposed System*, as it can be seen in Fig. 10, shows great stability with a steady growth. It is possible to see that the leverage does not influence the algorithm with great *Drawdowns*, meaning that the implementation of leverage did not destabilized the proposed system. The introduction of dynamic methods show that the solution obtained proved to be very stable, as it was intended, the objective is to have a steady growth, rather than huge variations that might have great returns but show great quantities of volatility.

The *Proposed System*, also, proved to be quite robust, because the testing period chosen, has the three types of market addressed in this paper, and especially the year of 2016, that has been a very difficult year for traders, and yet, the algorithm shows a good performance.

- **Diversity Measures** – One of this paper objective is to explore the possibilities of diversity, as it was stated before, the proposed system uses *Adaptability* methods to increase diversity, *Hyper-Mutation* and *Hyper-Selection*, provided new solutions in the search space, i.e., when these measures were activated the *Proposed Stopping Criteria* found, for 58% of times, new solutions in the search space, that obtained a better fitness than the previous ones. In this case study, when introducing hyper-mutation, i.e., when the mutation rate was increased temporarily, the GA is able to find solutions with higher ROI and at the same time avoid higher volatility because this measure is only temporarily, meaning that, when a new solution is found, the Stopping Criteria is activated and the generation number is reset to zero again. Hyper-Mutation and Hyper-Selection are only activated when the solution does not change for 15 generations, being this method, a good trade-off between diversity, and higher fitness function. Moreover, if the hyper-mutation is used during several generations, it is possible that the best solution may be lost due to a random mutation. Finally, the use of three different GAs provided three different populations that were trained for three different environments. Although this method may not be considered a *Multi-Population* method, it provides a reasonable increase in the diversity when compared to a single GA.

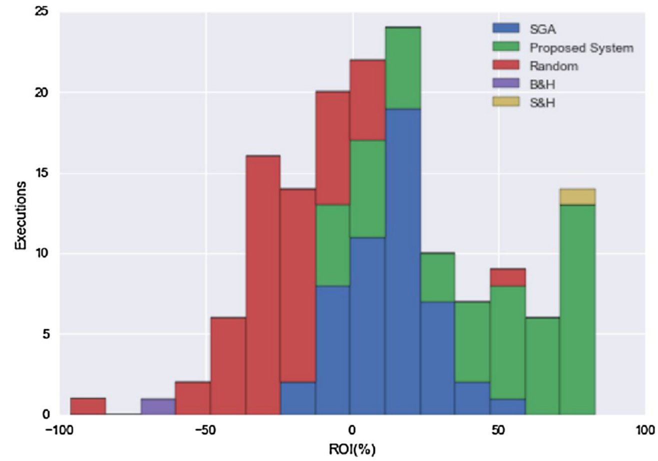

Fig. 12. Histogram with the ROI for the 50 executions.

4.3.2.2. Comparing proposed system with proposed benchmarks. The proposed solution, illustrated on Fig. 11, exhibits a better *Return on Investment* (ROI), compared to the approaches described in section 4.1. The proposed system outperforms the other strategies, although this solution is still a bit heavy on computation, and only breaks the other approaches in the end, duo to it's steady growth, it rarely looses positions with considerable losses. As it is possible to observe, the S&H and the SGA approaches reach higher returns at certain points, but they are very volatile, and the objective is to get a steady growth throughout time.

The histogram of the ROI described in Fig. 12, shows that for the proposed approach, 12% of the runs obtained negative values, 36% were values near zero or results that are somewhat similar to the other approaches, but still, for 76% of runs, the proposed system outperforms the other approaches, giving a high confidence for the proposed strategy. Applying the *Random Walk* approach is clearly worse in every run, compared to the rest of the approaches, this method shows that in 90% of times, the results were negative. For the *B&H* and the *S&H* the results clearly depended on the type of market they were in. For *B&H*, 100% of the runs, the results were below the proposed solution, and in *S&H*, 88% of the runs were also below the proposed solution. Moreover, the proposed solution is also put against a *Static GA* (*SGA*), *memoryless*, without *hyper-mutation*, and *hyper-selection* features. The results confirm the assumptions made for this work, where it is possible to conclude that dynamic approaches enhance the performance of the *GA*, especially the proposed approach of introducing the *SVM* module in the algorithm, making it as said before more stable and

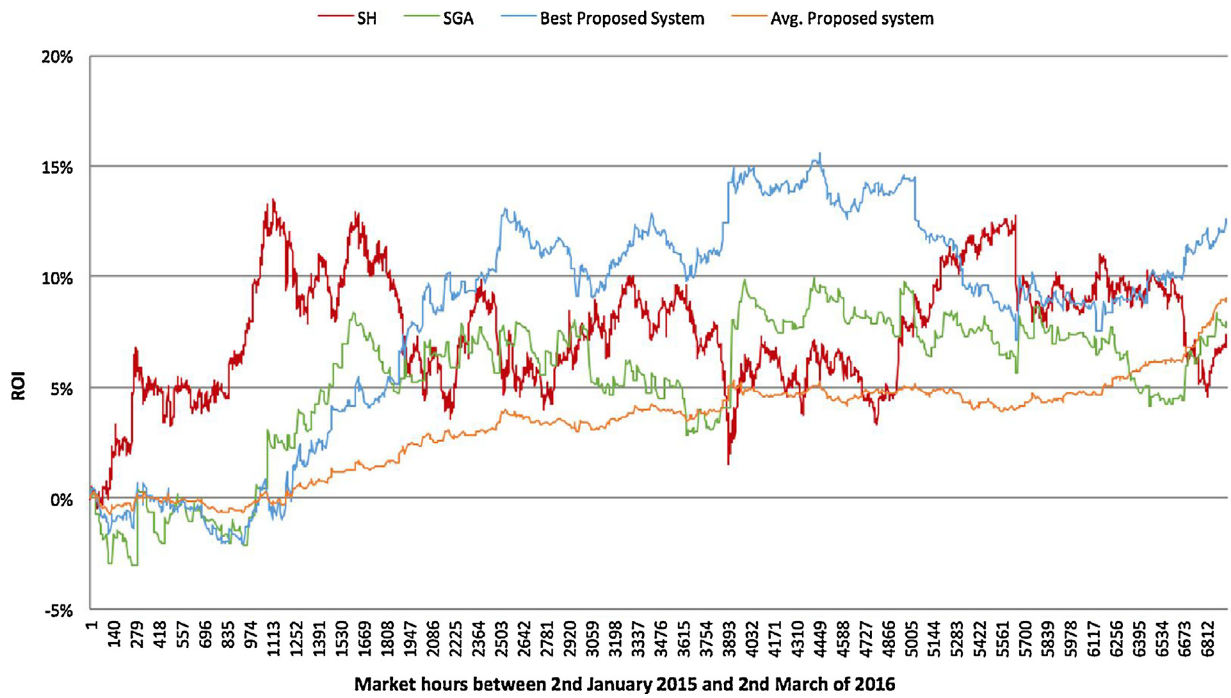

Fig. 13. Returns without leverage obtained during test period for the different approaches.

less exposed to great losses. In Fig. 12, it is possible to see that the proposed solution outperforms the *SGA* in 66% of the runs.

Regarding the *leverage* used in this approach, the proposed solution applies leverage in every run, but it is important to say that the level of leverage is chosen by the *GA*, also, it is possible to conclude that the amount of leverage used by the *GA* is a good approach, when compared with a fixed level of leverage used by the SGA, where the number of times that the *SGA*, and Random apply leverage correctly is only 16%, 2%, respectively. Table 17 presents a more detailed statistical analysis of the average results over the 50 runs.

Although the work developed by Sermpinis et al. [7] does not show a *ROI* evolution in a chart, the results provided, regarding *Drawdown* are of great importance. Table 15 shows that the Best *Proposed System* has a *Max. Drawdown* of 14%, and the results obtained for the EUR/USD in the Sermpinis work show very similar results, of 13.63 for the *GA-SVM* approach, 14.81 for *GA-SVR* approach and 14.38 for *RG-SVR* approach. Although the Test Period is not the same, the results obtained by Sermpinis et al. indicate that the Drawdown obtained for the *Best Proposed System* is a successful result.

It is also important to demonstrate that the proposed solution outperforms the other without leverage. Fig. 11 indicates that the leverage imposed by the GA chromosome, used in the *Proposed System*, does not increase the risk when compared with the proposed solution that, does not implement leverage in Fig. 13, showing the *Proposed System* does not make considerable mistakes when inserting leverage. Fig. 14 exhibits the variation of the leverage levels during the test period, showing that, the levels of leverage depended on the type of market, more specifically in this execution the Bullish Market obtained a leverage of 9, the Bearish Market a leverage level of 8 and the Sideways Market a leverage level of 2, note that, when the proposed system does not use leverage, the level is always 1. Moreover, Fig. 14 shows that for well defined markets the leverage level is higher than the rest and even more, the leverage level is higher in the situations where the FX market presents higher price variations. Finally, the leverage levels presented are not actual columns, but a very oscillatory variation of the levels it self, making it look like a histogram.

When compared the *Best Proposed System* with the leverage approach made by Hirabayashi et al. [2], it is possible to see that the *Best Proposed System* shows a steadier behaviour and less risk exposure when trading EUR/USD. Again, these cannot be direct comparisons since the work made by Hirabayashi et al. [2] does not focus on the EUR/USD market and the Test Period is different, nonetheless, the work made by Hirabayashi et al. [2] is a good benchmark to prove that the *Proposed System* provides promising results. Both the works that were compared with the *Proposed System* proved that is possible to use Evolutionary Computation in the *FX market*.

Finally, the solutions show that the different approaches might lose more money than the initial investment. This situation can be explained by the fact that the representation is only mathematical, and due to leverage it is possible to lose more money than the initial investment, as shown in “Concept of Leverage” (Section 2.3).

After comparing the proposed system against other solutions, it is important to describe the behaviour of the *GA* when it comes to choose the level of leverage for the best population. Fig. 15 describes the optimization made for the three types of markets addressed in this work, showing both the histogram (Fig. 15a) for the 50 executions and a density distribution (Fig. 15b) to better understand how the results are distributed in the search space of possible leverage levels. Moreover, the *GA* obtained the different optimal solutions for the best population, depending on the type of market, the Y label shows the results for the distributions obtained during the 50 executions and for each market, it is shown the level of leverage optimized for that specific execution.

As it is possible to see, for both *Bearish* and *Bullish* markets, that are, traditionally, easier to describe and to define, the leverage level optimized is in a great majority of time, a level 9. This result is rather promising because the *GA* can successfully introduce leverage in the higher levels, meaning that, profitable positions are being well chosen for this two types of markets. As for the *Sideways* market, results show that for this type of market, the leverage level obtained with higher density was the level 2 and 3, showing that this type of market is, as it was said before, harder to define and thus harder to make investments in those periods, also, this market shows a

Table 17
Comparison between the proposed system and the benchmark approaches.

Parameter	Random	B&H	S&H	SGA	Avg. Proposed System	Best Proposed System
ROI (%)	−22.3	−72.2	72.2	12.5	43.9	83.5
ROI without Leverage (%)	−5	−7.22	7.22	8.1	8.9	12.8
Profitable Positions (%)	26	31	69	39	57	63
Std. Dev. of ROI (%)	26.9	17.7	17.7	15.4	17.3	30.2
Number of days with negative ROI(%)	68.1	100	0	10.3	15.4	16.5
Max. Drawdown (%)	200	160	20	60	9	14

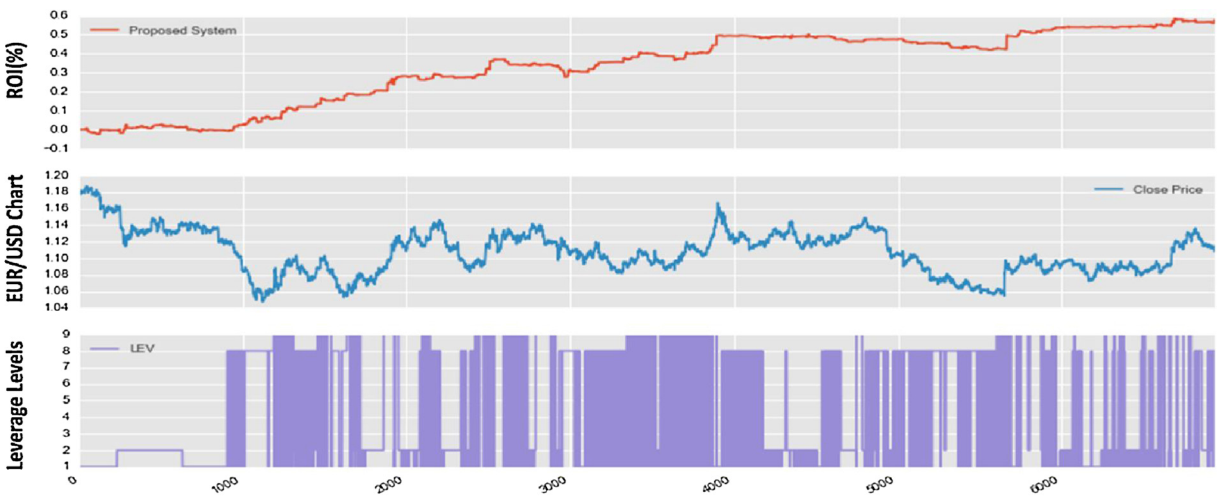

Fig. 14. Returns with leverage marks obtained during *Best Proposed System*.

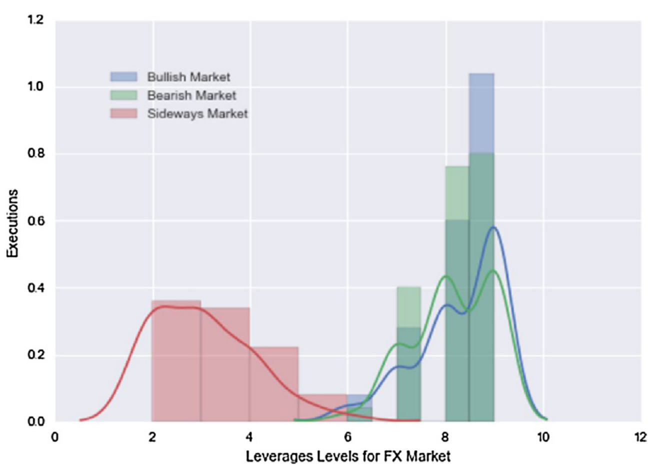

Fig. 15. a) Leverage levels obtained for the three different markets in 50 executions. b) Density Distribution of the Leverage Levels obtained during the 50 runs.

wider distribution when compared to the *Bullish* and *Bearish* market, indicating that this two types of markets are more profitable to invest than the *Sideways* market. Results are in line with the theory of *FX* markets, proving that investing in a *Sideways* market is more volatile, due to the fact that, there is no defined trend.

To conclude this section, one can see that the *Proposed System* with leverage outperforms the *Proposed System* without leverage. A very positive outcome is that both results return positive *ROIs*, meaning that the *Proposed System* is well designed and does not return profit only because it has leverage. Introducing leverage provided a great improvement in the results, showing that this tool is very important in a FX market. nonetheless, it is necessary to use it with care, since it can turn against the investor if the exposure to risk is to high, being the main reason to choose levels of leverage that are smaller than the usual used in the investment world. By using high levels of leverage in an Evolutionary system, showed that is possible to get high returns without exposing the system to high levels of risk, thus showing that the *Proposed System* is a very promising algorithm to invest in *FX* markets.

5. Conclusions

This work proposes a viable solution to automatically invest in Forex Markets, using technical indicators and price sequences to predict entry and exit points. The solution combines a SVM to classify the market with genetic algorithms to find different trading rules for each type of the market. To improve the solution more data about the FX market should be used, the bigger the populations and the more price sequences the SVM is provided to perfect the training module, the better the solution is. Also, by studying the SVM approach, it is possible to conclude by the results, that the sideways market is more difficult to describe than the others as it was seen in the previous section, where the label “0” showed poorer quality in the classification. As it was possible to see, the GA showed good adaptive response, and a good trade-off between diversity and quality of a given population. It is also true to conclude that the memory approach improved the performance of the GA, and thus, being very important so that the algorithm could adapt to different environments during the evolution of the price sequences. The results obtained confirmed the improved performance achieved by using a leveraged strategy.

For future works, it is important to keep in mind that, although the Evolutionary Computation plays a fundamental role to provide the best solutions, the investment rules used to entry or exit the market can be perfected with more elegant rules and strategies that are used by professional traders. In a more technical point of view, it is considered as important issues to explore the following ideas: Explore unsupervised learning methods to classify different types of markets; For classifying purposes, a volatility prediction to help the *SVM* to improve the market type classification is important, not only to classify but also to make investment decisions and provide information for the leverage tool, i.e., this work classified three types of markets, but it is interesting to develop the idea that within those three types there are more types of markets; Introduce new mechanisms of multi-population schemes, that help the algorithm to search in parallel different types of solutions; Studying the impact of introducing a back-end *GA* to optimize the parameters used in the *SVM*, rather than use a *Grid Search* algorithm; Comparing different *Stopping Criteria* methods, such as introducing measures of variance in the population, or in the fitness function. The Stopping Criteria can influence the search of new solutions, thus being a very important topic in the study of *Genetic Algorithms*;

Acknowledgments

This work was supported in part by Fundação para a Ciência e a Tecnologia (Project UID/EEA/50008/2013).

References

[1] S. Cirillo, S. Lloyd, P. Nordin, Evolving intraday foreign exchange trading strategies utilizing multiple instruments price series, CoRR November (2014).
[2] A. Hirabayashi, C. Aranha, H. Iba, Optimization of the trading rule in foreign exchange using genetic algorithm, in: Proceedings of the 11th Annual Conference on Genetic and Evolutionary Computation (GECCO '09), Montreal, Canada, New York USA : ACM, 2009, pp. 1529–1536.
[3] J. Yao, C.L. Tan, A case study on using neural networks to perform technical forecasting of forex, Neurocomputing 34 (September (1–4)) (2000) 79–98.
[4] C. Evans, K. Pappas, F. Xhafa, Utilizing artificial neural networks and genetic algorithms to build an algo-trading model for intraday foreign exchange speculation, Math. Comput. Modell. 58 (January (5–6)) (2014) 1249–1266.
[5] S. Deng, S. Akito, Foreign exchange trading rules using a single technical indicator from multiple timeframes, Proceedings of the 27th IEEE International Conference on Advanced Information Networking and Applications Workshops(WAINA) (2013) 207–212.
[6] M. Pb, B. Adam, Evolutionary algorithm in forex trade strategy generation, Proceedings of the IEEE International Multiconference on Computer Science and Information Technology (IMCSIT) (2010) 81–88.
[7] G. Sermpinis, C. Stasinakis, K. Theofilatos, A. Karathanasopoulos, Modeling, forecasting and trading the EUR exchange rates with hybrid rolling genetic algorithms – support vector regression forecast combinations, Eur. J. Oper. Res. 247 (3) (2015) 831–846.
[8] A. Gorgulho, R. Neves, N. Horta, Applying a GA kernel on optimizing technical analysis rules for stock picking and portfolio composition, Expert Syst. Appl. 38 (11) (2011) 14072–14085.
[9] C. Aranha, H. Iba, Modelling cost into a genetic algorithm-based portfolio optimization system by seeding and objective sharing, Proceedings of the IEEE Congress on Evolutionary Computation (CEC) (2007) 196–203.
[10] P. Fernández-Blanco, D.J. Bodas-Sagi, F.J. Soltero, J.I. Hidalgo, Technical market indicators optimization using evolutionary algorithms, in: Proceedings of the 10th Annual Conference Companion on Genetic and Evolutionary Computation (GECCO '08), Atlanta, USA, 2008, pp. 1851–1858.
[11] L. Mendes, P. Godinho, J. Dias, A forex trading system based on a genetic algorithm, J. Heuristics 18 (4) (2012) 627–656.
[12] H. Zhang, R. Ren, High frequency foreign exchange trading strategies based on genetic algorithms, Proceedings of the 2nd IEEE International Conference on Networks Security Wireless Communications and Trusted Computing (NSWCTC) (2010) 426–429.
[13] A. Gorgulho, R. Neves, N. Horta, Using GAs to balance technical indicators on stock picking for financial portfolio composition, in: Proceedings of the 11th Annual Conference Companion on Genetic and Evolutionary Computation Conference: Late Breaking Papers, Montreal, Canada, 2009, pp. 2041–2046.
[14] A. Gorgulho, N. Horta, R. Neves, Intelligent Financial Portfolio Composition Based on Evolutionary Computation Strategies, Springer-Verlag, Berlin Heidelberg, 2013.
[15] S. Yang, X. Yao, Experimental study on population-Based incremental learning algorithms for dynamic optimization problems, Soft Comput. 9 (11) (2005) 815–834.
[16] S. Yang, Non-Stationary problem optimization using the primal-dual genetic algorithm, Proceedings of the IEEE Congress on Evolutionary Computation (CEC) (2003) 2246–2253.
[17] S. Yang, Evolutionary computation for dynamic optimization problems, in: Proceedings of the Companion Publication Ofthe 2015 Annual Conference on Genetic and Evolutionary Computation (GECCO '15), Madrid, Spain, 2015, pp. 629–649.
[18] T.T. Nguyen, S. Yang, J. Branke, Evolutionary dynamic optimization: a survey of the state of the art, Swarm Evol. Comput. 6 (2012) 1–24.
[19] J. Branke, Memory enhanced evolutionary algorithms for changing optimization problems, Proceedings of the IEEE Congress on Evolutionary Computation (CEC) (1999) 1875–1882.
[20] S. Yang, X. Yao, Population-Based incremental learning with associative memory for dynamic environments, IEEE Trans. Evol. Comput. 12 (October (5)) (2008) 542–561.
[21] T. Park, R. Choe, K. Ryel, Dual-Population genetic algorithm for non-Stationary optimization, in: Proceedings of the 10th Annual Conference on Genetic and Evolutionary Computation (GECCO '08), Atlanta, USA, 2008, pp. 1025–1032.
[22] S. Yang, Genetic algorithms with memory-and elitism-Based immigrants in dynamic environments, MIT Press J. (2008) 385–416.
[23] H.G. Cobb, J.J. Grenfenstette, Genetic Algorithms for Tracking Changing Environments, Naval Research Lab, Washington DC, 1993.
[24] Y. Shengxiang, T. Renato, Hyper-Selection in dynamic environments, Proceedings of the IEEE World Congress on Computational Intelligence (2008) 3185–3192.
[25] A. Simões, E. Costa, Improving prediction in evolutionary algorithms for dynamic environments, Proceedings of the 11th Annual Conference On Genetic and Evolutionary Computation (GECCO '09) (2009) 875–882.
[26] IPython, Spyder – Documentation – Spyder 2.3 Documentation, 2009 ([Online]. Available: https://pythonhosted.org/spyder/).
[27] Dukascopy [Online]. Available: https://www.dukascopy.com/.
[28] Python data analysis library – pandas: Python data analysis library, [Online]. Available: http://pandas.pydata.org/.
[29] TA-Lib. [Online]. Available: https://mrjbq7.github.io/ta-lib/func.html.
[30] Precision-recall – scikit-learn 0.17.1 documentation, in Scikit Learn, 2010. [Online]. Available: http://scikitlearn.org/stable/auto_examples/model_selection/plot_precision_recall.html.
[31] B. Scholkopf, A. Smola, Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond, MIT press, 2001.