Central European Economic Journal

Przemysław Ryś[1], Robert Ślepaczuk[2,3]

Machine Learning Methods in Algorithmic Trading Strategy Optimization – Design and Time Efficiency

1 Quantitative Finance Research Group, Faculty of Economic Sciences, University of Warsaw and Labyrinth HF project; Corresponding author: przrys@gmail.com.

2 Quantitative Finance Research Group, Faculty of Economic Sciences, University of Warsaw and Labyrinth HF project, ORCID ID: 0000-0001-5227-2014.

3 The views presented in this text are those of the authors and do not necessarily represent those of Circus Consulting Group nor LHF project.

Open Access. © 2018 P. Ryś, R. Ślepaczuk, published by Sciendo. BY-NC-ND This work is licensed under the Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 License. https://doi.org/ 10.1515/ceej-2018-0021

Abstract: The main aim of this paper was to formulate and analyse the machine learning methods, fitted to the strategy parameters optimization specificity. The most important problems are the sensitivity of a strategy performance to little parameter changes and numerous local extrema distributed over the solution space in an irregular way. The methods were designed for the purpose of significant shortening of the computation time, without a substantial loss of strategy quality. The efficiency of methods was compared for three different pairs of assets in case of moving averages crossover system. The problem was presented for three sets of two assets' portfolios. In the first case, a strategy was trading on the SPX and DAX index futures; in the second, on the AAPL and MSFT stocks; and finally, in the third case, on the HGF and CBF commodities futures. The methods operated on the in-sample data, containing 16 years of daily prices between 1998 and 2013 and was validated on the out-of-sample period between 2014 and 2017. The major hypothesis verified in this paper is that machine learning methods select strategies with evaluation criterion near the highest one, but in significantly lower execution time than the brute force method (Exhaustive Search).

Keywords: Algorithmic trading, investment strategy, machine learning, optimization, investment strategy, differential evolutionary method, cross-validation, overfitting.

JEL Codes: C61, G11, G17

1 Introduction

The last years witnessed a huge growth of the machine learning popularity and its quick development. The newly established algorithms were used to solve many difficult problems from various fields of science and to produce solutions facilitating many areas of life. Therefore, the application of such methods to improve the process of strategy adjustment seemed to be a natural choice.

The main aim of this study was to formulate and analyse the machine learning methods, fitted to the strategy parameters' optimization specificity. The most important problems are the sensitivity of a strategy performance to little parameter changes and numerous local extrema distributed over the solution space in an irregular way. The methods were designed for the purpose of significant shortening of the computation time, without a substantial loss of a strategy quality. The efficiency of methods was compared for three different pairs of assets in case of moving averages crossover system. Considered algorithms – the Extended Hill Climbing, Grid Method and Differential Evolution Method are based on the well-known machine learning methods or intuitive ideas based on observation of previous steps in order to improve the next ones.

The machine learning methods, discussed in this paper were designed to select the strategy parameters in order to maximize strategy performance, measured by the specified optimization criterion. The methods operated on the in-sample data, containing 16 years of daily prices, and their results were verified on 4 years of out-of-sample data. In the first case, a strategy was trading on the *SPX* and *DAX* index futures, in the second on the *AAPL* and *MSFT* stocks and finally, in the third case on the *HGF* and *CBF* commodities futures.

The major hypothesis verified in this paper is that results of the machine learning methods are the same or only slightly worse than the ones near the highest evaluation criterion, obtained by the Exhaustive Search (brute force approach), but the time required for their execution is significantly lower than computation time of checking all the points from the solution space. The additional research question is that the strategies obtained by the machine learning methods are associated with a lower risk of overfitting than the strategies resulted from the Exhaustive Search procedure.

The distributions of optimization criteria and the computation time of 1000 executions of different methods were compared and presented along with the Exhaustive Search results. The adjustment quality was assessed on in-sample data and additional out of sample data in order to test the overfitting tendency. Let us emphasise that the purpose of this paper is not to design the most profitable strategy, but to compare the efficiency of different machine learning methods and the Exhaustive Search (brute force). Tests in the out-of-sample period were performed to assess the overfitting problem. The simulations for different sets of assets was executed in the same framework implemented for the purpose of this research.

The basic machine learning methods have serious disadvantages. For instance, the well-known Hill Climbing returns the local extremum, without guarantee of reaching the global one. That algorithm is inadequate for the global search problem, but it could be used as a main component of more complex and efficient methods of global optimization.

Since the machine learning methods proved their value, by solving plenty of complicated problems, hence, it was reasonable to expect the satisfying results of such methods used for the strategy optimization. The initial intuition was that the machine learning methods would return the results a bit worse than the optimal one, but in disproportionately shorter time, than checking all the possibilities in order to get the best ones (the Exhaustive Search).

Moreover, it was expected that the machine learning methods were less likely to overfit strategy than the Exhaustive Search. The discussed methods were based on an assumption that conditional expected value of the optimization criterion is usually higher for the points surrounded by those with high value of this criterion. Therefore, the low regularity of the solution space could be a real obstacle for the methods' performance. There was no reason to assume even a moderate level of the space regularity, so the machine learning methods probably could not find the optimal points, if they were not in the high-valued neighbourhood. That property could lead to reducing overfitting risk, because usually, the parameter vector surrounded by those with similar strategy performance have a bigger chance to be profitable in the future, than those from a less stable place.

The structure of this paper is composed as follows. The second chapter contains the literature review. In the third part, machine learning methods used in this paper are explained, as well as the trading assumptions and basic terms. The fourth chapter is devoted to data description, when the fifth contains efficiency tests of considered machine learning methods, with special focus on the optimization criterion and computation time distributions . The summary of results and conclusions are included in the last part.

2 Literature review

The machine learning methods have been developed for decades, even before that term was coined in the fifties (Samuel, 1959). Nevertheless, the increased interest in that field was observed in recent years due to the technical possibility to apply the artificial intelligence in the various fields of science and life. The phenomenon of learning from the computational viewpoint was discussed by Valiant (1984). The human's natural ability to learn and adapt was presented in terms of the information's selection and automatic adjustment process, resulting in the algorithm's modifications.

This approach is followed by plenty of the modern machine learning methods and it is close to the general ideas of the classic statistical modelling, where including new dataset leads to changes in the model properties. The traditional statistical and econometric models usually assume that data is produced by the stochastic process from the specified class. The fitting procedure is aimed at finding the process accurate to actual data when the machine learning methods are often based on the iterated improvements without specified model form. The differences between these two approaches called data models and algorithmic models respectively, are widely discussed in Breiman (2001). The field of machine learning contains plenty of various algorithms and methods, used to solve a wide range of problems. Some methods have strong mathematical foundations, for instance, methods based on Markov Chain Monte Carlo (Neal, 1993), when others, such as the Hill Climbing or evolutionary methods, are based on heuristic approach (Juels and Wattenbergy, 1994). The commonly used methods and algorithms with application in scientific problems are discussed by Hastie *et al.* (2013) and Hastie *et al.* (2001).

The algorithmic strategies are widely used in the financial markets, but most of them are not discussed in papers, due to exclusive character. Nevertheless, some types of the quantitative strategies are widely known, and therefore, discussed in books and papers. The strategy based on the technical analysis indicators, such as the simple moving average crossover method considered in this paper is analysed for specified cases in Gunasekarage and Power (2001). Since machine learning methods have started to gain popularity, as a tool to solve problems in various fields, numerous attempts to use it for trading strategies occurred. Beyond the commercial usage, many academic papers describing strategies, with logic based on a machine learning have been published. For example group of researchers at Sanković *et al.* (2015), presented the strategy, based on the technical analysis and least squares support vector machines. In contrast to this paper, they used machine learning methods as a part of a system generating trading signals, not as a part of system optimization process.

The more recent research was conducted by Ritter (2017), who used Q-learning with the proper rewarding function to handle the risk-averse case and tested strategy in the simulated trading. Dunis and Nathani (2007) presented the quantitative strategies, based on the neural networks such as the Multilayer Perceptron (MLP), Higher Order Neural Networks (HONN) and on the K-Nearest Neighbours method. The authors proved that methods can be effectively used for generating excess returns from trading on gold and silver. The comparison between the performance of machine learning methods and the linear models of ARMA type not only lead to construct a better strategy but additionally showed the presence of nonlinearities in the considered time series.

The application of the machine learning methods in order to predict future prices nowadays becomes more and more popular. Shen, Jiang and Zhang (2012) presented the forecasting model for stock market indexes, based on Support Vector Machines, and tested the trading system based on the produced predictions. Similar approach was followed by Choundhry and Kumkum (2008), where they introduced the hybrid machine learning system, combining support vector machines with genetic algorithm in order to predict the stock prices. The machine learning methods were used for predicting by Patel *et al.* (2015) in a more recent research as well. The paper is focused on methods of data pre-processing for purposes of further forecasting. Therefore, many books and papers discuss the general aspects and methods, such as walk-forward optimization (Kirkpatrick and Dahlquist, 2011 or Pardo, 2011).

Differential evolution, which is one of the methods considered in this paper, was designed by Storn and Price (1997) and discussed in further papers, such as Price *et al.* (2006). The algorithm was proposed for solving complicated problems with irregular solution space. It was used to solve non-convex portfolio optimization problems in Ardia *et al.* (2010) and the problem of minimizing CVAR for the large-scale portfolio in Ardia *et al.* (2011a). The method proved to be an efficient and effective way to optimize complex problems.

3 Machine learning methods and their benchmark

3.1 Basic terms and methodological issues

All the statistics used in the optimization criterion were determined by the equity line and could be easily calculated based on it. Therefore, the calculations of net profits and losses (PnLs) were the most complex component of the strategy evaluation procedure. The system was based on the technology called *Rcpp*, allowing to use efficient *C*++ programs inside *R* project. More precisely, the main function for calculating net PnL was implemented in *Rcpp* framework in order to accelerate computations based on loops. Other parts of system were designed in R due to vectorization possibility and high-performance of build in functions.

3.1.1 Statistics and risk metrics used in the paper

- **Annualized rate of return** – relative change of an asset value, normalized according to time. The annualized rate of return calculated for the asset of value V_t in specified period (t_1, t_2) is defined by the following formula:

$$ARC(V)_{t_1}^{t_2} = \left(\frac{V_{t_2}}{V_{t_1}}\right)^{\frac{1}{D(t_1,t_2)}} - 1 \quad (1)$$

where: $D(t_1, t_2)$ stands for the time between t_1 and t_2 in years.

- **Maximum drawdown** – the maximum percentage loss of value of the value process. For process V_t and period $[t_1, t_2]$, the maximum drawdown is defined by the following formula:

$$MDD(V)_{t_1}^{t_2} = sup_{(x,y)\in\{[t_1,t_2]^2 : x \le y\}} \frac{V_x - V_y}{V_x} \quad (2)$$

- **Annualized standard deviation** – the empirical standard deviation normalized, according to the time. For specified time series R_t, the annualized standard deviation in the period $[t_1, t_n]$ is calculated by using the formula:

$$ASD(V)_{t_1}^{t_n} = \sqrt{\frac{1}{n}\sum_{t=t_1}^{t_n} (R_t - \overline{R})^2 * \frac{1}{D(t_1, t_n)}} \quad (3)$$

where: $\overline{R} = \frac{1}{n}\sum_{t=t_1}^{t_n} R_t$

and $D(t_1, t_2)$ is the time between t_1 and t_2 in years.

- **Moving average** - for the specified time series S_t and the wages vector $w = (w_0, w_1, ...)$ the moving average is a time series given by the formula:

$$MA(S)_t^w = \sum_{i=0}^{t} S_{t-i} w_i \,. \quad (4)$$

3.1.2 Assumptions

The main problem was to find the best investment strategy in the specified class of strategies, following the two simple moving averages crossover approach. The behaviour of each strategy was fully determined by a vector of parameters from four-dimensional space Ψ, each standing for different moving average window width. More specifically, every strategy from Ψ was parametrized by a vector $p = (p_1, p_2, p_3, p_4) \in \Psi$, such that the trading signals

U_t^i for $i \in \{1, 2\}$ were given by the following formula:

$$U_t^i = sign(MA_1^i(S^i)_{t-1} - MA_2^i(S^i)_{t-1}) \quad (5)$$

where: $MA_j^i(S^i)$ denoted the simple moving average of length p_{2i+j-2} for price series S^i, which means that p_1 and p_2 stands for lengths of moving averages used for the first asset, and analogously p_3 and p_4 refer to the second asset. Additionally, we took convention that strategy component is called *momentum* if the first moving average length is smaller than the second, and analogously *contrarian* if the first moving average length is higher.

Optimization criterion was based on the typical descriptive statistics used by traders – the annualized returns (ARC), the annualized standard deviation (ASD) and maximum drawdown (MDD). The criterion was determined by the following formula:

$$OC = \frac{ARC * |ARC|}{ASD * MDD} \tag{6}$$

The construction of the optimization criterion *OC* reflected the preference of moderately profitable strategies with low risk, over the high-profitable ones, associated with much higher risk. That approach was designed to find more safe and stable strategies, in order to generate profits in the future. This criterion additionally punished strategies with high *MDD*, which means that the strategies with lower returns and constant increasing trend are more likely to be selected than the more profitable ones, associated with periods of significant losses.

Conditions on the financial markets were different during the tested time period, from *1998* to *2017*. For the sake of simplicity, the constant transactional percentage costs were assumed. In the simulated trading, every trade required bearing the fee equal to 0.25% of its value. Additionally, we assumed leverage on the level of *40%*, which means that every considered strategy invested 20% of the total account balance on each asset (40% in total). The rebalancing took place once each 5 trading days.

The available strategies were fully determined by four parameters, standing for moving averages widths. Consequently, the strategy was optimized on the parameters (solution) space Ψ composed of vectors of four numbers from the set {1, 5, 10, ... , 100} (i.e., $\Psi = \{1, 5, 10, \ldots, 100\}^4$).

3.2 Specificity of the problem

The problem of selecting the best parameters of a trading strategy could be parametrized and reformulated in terms of optimization. The optimization criterion (*OC*) is, as specified before, calculated based on annualized returns and risk measures. The parameter space and the reward function had some important properties, that need to be included in the machine learning methods' design in order to meet the problem specific requirements and reach better efficiency.

Solution space (Ψ) is discrete; thus, the application of algorithms, based on the steps of decreasing size, was strictly limited. Moreover, the function being optimized had no simple analytical formula. In consequence, there was no way to apply popular gradient-based methods. The *OC* could be obtained from the statistics of the equity line for the specified strategy parameters. The calculations were not very complicated, but they required relatively long time to be executed. The high time complexity was caused by the calculation of profits and losses for every trading day. Therefore, the main difficulty was caused rather by the time expensive criterion evaluations, than by the big number of possible parameter combinations (*194 481*).

Additionally, the performance of automatic strategies is usually sensitive to the parameters' change; therefore, even subtle difference could severely affect the results. In consequence, one can expect multiple local extrema scattered over the parameter space and big differences in criterion value of the points near each other. High sensitivity of the optimization criterion (objective function) to the parameters was crucial for the machine learning efficiency and led to the selection of more complex methods, adjusted to the problem specificity. Although the optimization criterion was unstable, some level of regularity was necessary for machine learning methods to work. Machine learning algorithms selected the points (candidate solutions), surrounded by other with the high criterion value, which could positively affect results' stability and reduce overfitting risk.

The machine learning methods presented in this paper are based on well-known concepts. The main effort was to design methods based on these algorithms, but able to run on an atypical problem, hard to be solved by the basic ones. Although the presented methods could result in lower overfitting risk, the paper was focused on the improvement of the strategy selection procedure in terms of time, and hence no features aimed at reducing the overfitting risk would be discussed.

3.3 Extended Hill Climbing (EHC)

The basic **Hill Climbing** is a local search method, based on a very intuitive approach – going always in the way that improves the situation. The method operates on a graph, composed of nodes (points) with optimization criterion value and edges, which determine the relation of being neighbours. The basic Hill Climbing algorithm is a simple loop, starting at the specified point and repeatedly changing current point to its neighbour with higher optimization criterion value, as long as an improvement was possible by making the step forward. The classic Hill Climbing procedure checks all the neighbours and selects the one with the highest value. The method traverses the parameter space with only one rule of always going up, using no information from the past, except the current position. The accurate parallel for that algorithm is 'trying to find the top of Mount Everest in a thick fog while suffering from amnesia' (Russell and Nowig, 2003). The method can find only the local extremum, thus it is local search. That is the significant limitation of this method's use, because there was no reason to expect that method will end search in one of the best solutions. Neighbours are often defined as points with the specified distance between them. In this case, the algorithm traverses the parameter space using steps of the same size. The Hill Climbing is well-adjusted for problems with great regularity, such as convex ones, where exactly one local maximum exists (Skiena, 2008). It is no reason to assume that the considered problem has that property, therefore, the main effort in this method improvement was made by adjusting method to spaces with many local maxima.

The Extended Hill Climbing (EHC) is composed of the independent Hill Climbing executions, called walks. These walks are starting at different random points and the best result among them is returned at the end. Every single walk procedure checks the neighbours of the current point and goes ahead when the first improvement is found. It is a substantially different approach than followed by the classic one, reducing computation time, because it does not require calculating optimization criteria for all the neighbours. Another new feature is the use of a few different neighbours' structures. More specifically, algorithm checks the neighbours differing by exactly one parameter, which implies that a walk is on the perpendicular multidimensional grid, similar to the chess rook. At the very beginning and after making every move, the algorithm checks points with distance at specified, relatively high levels. When no improvement is possible, the method checks points with lower distance from the current one. This procedure is repeated until no improvement is possible; either the next move is made or the specified minimum stepsize is exceeded. The stepsize series is defined as

$$\{F, \left\lceil\frac{F}{k}\right\rceil, \left\lceil\frac{F}{k^2}\right\rceil, \ldots, 1\},$$

where F is a starting step, in this paper, equal to 5 and k is equal to 2. The initial preference of big steps resulted in fast crossing the space and the possibility of walking by small steps allowed the algorithm to finally search the small neighbourhood in order to find solution with optimization criterion as high as possible. This feature is a crucial element of the method – algorithm uses steps of various sizes; thus, a supervisor does not need to select step size as cautiously as for method with fixed step size. What is more, method always uses the step of minimal size at the end; the neighbourhood of the best point is precisely explored.

The number of walks required to get satisfying results is stochastic. Thus, declaring the fixed number could result in low stability of results – the difference between optimization criteria obtained in the independent optimizations could be significant. The extended method set the number of walks in a dynamic way, dependent on the efficiency of previous walks. The algorithm starts twice as many new walks, if the previous set of walks improved the optimization criterion, which suggests that there is still a possibility to improve results. That solution guarantees the higher results' stability, at the expense of the time stability. The time required for execution could be much higher, when the method starts in a different starting point, but on the other hand, 'bad' starting points should not affect the final results. In this paper, the initial number of walks is equal to 10.

Algorithm pseudocode.

Metaparameters: *initialIterationsNumber, k, initialStep*. Let us clarify, that *unitVector*$_i$ denotes the i-th unit vector of space $\mathbb{R}^d$ that is, vector of zeroes with 1 on i-th coordinate – *unitVector*$_i$ = e_i = (0,..., 0, 1, 0,..., 0). Result of an operation $x + c \times e_i$ is simply vector x with i-th coordinate increased by c (decreased by -c for a negative c).

Listing 1. Algorithm pseudocode for Extended Hill Climbing method.

```
Set iterationsNumber = initialIterationsNumber / k.
Set bestValue = -Inf.
Set bestPoint = NULL.

While TRUE {
Set iterationsNumber = Round(iterationsNumber * k).
Set bestValuePackage = -Inf.
Set bestPointPackage = NULL.

// Current walks set (package) of size equal to it-
erationsNumber.
For j = 1 to iterationsNumber {
Draw starting point x from uniform distribution over
the parameters space.
Set bestValueWalk = optimizedFunction(x).
Set bestPointWalk = x.
Set step = initialStep.
```

```
 // Single walk.
 While TRUE {
 Set previousBestValueWalk = bestValueWalk.

 // Checking neighbours.
 For every parameter space dimension i {
 Set neighbourUp = x + currentStep * unitVector_i.
 Set currentFunctionValue = optimizedFunction(neigh-
bourUp).

 If optimizedFunction(neighbourUp) > currentMaxValue
and
 neighbourUp is element of parameters space. {
 Set bestValueWalk = currentFunctionValue.
 Set bestPointWalk = neighbourUp.
 Break for loop.
 }

 Set neighbourDown = x + currentStep * unitVector_i.
 Set currentFunctionValue = optimizedFunction(neigh-
bourDown).

 If currentFunctionValue > currentMaxValue and
 neighbourDown is element of parameters space. {
 Set bestValueWalk = currentFunctionValue.
 Set bestPointWalk = neighbourUp.
 Break for loop.
 }
 }

 // Change stepsize if no better point found.
 If bestValueWalk == previousBestValueWalk
 Set step = RoundDown(step / k).

 If step == 0
 Stop walk by breaking current while loop.
 }

 // Update best value and point in package if needed.
 If bestValueWalk > bestValuePackage {
 Set bestValuePackage = bestValueWalk.
 Set bestPointPackage = bestPointWalk.
 }
 }

 // Stop algorithm if no improvement in the previous
package.
 If bestValuePackage > bestValue {
 Set bestValue = bestValuePackage.
 Set bestPoint = bestPointPackage
 } else {
 Break while loop and return bestPoint and bestValue.
 }
 }
```

3.4 Grid Method (GM)

The second machine learning method, called *the Grid Method (GM)* is designed to operate on a limited space of discrete parameters, called grids. The method is composed of simple exhaustive searches, finding the best points from the parameters subgrids. The subgrids with a decreasing interspace are considered in the consecutive steps of the method. Firstly, the subgrids of full range and the relatively high interspace between parameters are considered and some of the best feasible solutions are used as a starting point for new independent procedures. Every starting point becomes the centre of a new subgrid, with a lower interspace between parameters. After predefined number of iterations, the interspace between parameters is minimal and then the solution with the biggest criterion among all subgrids is returned. This method is purely deterministic and need the initial subgrid of parameters' space to reflect the properties of whole space, such that the best global solutions will be around the best solutions from the initial subgrid. Otherwise, the method cannot return satisfactory results. Therefore, the high-value solution, surrounded by the worse ones might not be found by this method.

The search could be improved by setting different meta-parameters, such as the number of starting points or the interspace between parameters in the initial grid. There is a natural trade-off between the method's accuracy and the computation time due to the fact that the computation time was approximately proportional to the number of evaluations. Setting the meta-parameters allows to balance between method precision and time in an easy and effective way. Another big advantage is the deterministic nature of the method. There is no uncertainty about the method's results or computation time, which could be observed for random methods, such as the *Extended Hill Climbing*. Moreover, the computation time could be estimated before execution due to the predefined number of evaluated points. The method is easy to parallelize, as well as the previous one – the procedures for separated grids could be executed at the same time on different CPUs. The discussed method is prepared for the purpose of this paper – it is not an extension of the existing machine learning methods; however, it follows the basic idea, so it could be the method in use, following a similar approach.

Throughout the paper, the number of starting points is always equal to *6* and the interspace between initial subgrid parameters is equal to the initial step of the previous method – *5*. The grid size is always 5x5x5x5, centered at the best point from the previous search procedure step. Interspace of a grid in the next step is *2* times shorter than that in the previous one (rounded up if needed).

Algorithm pseudocode

Algorithm is provided for parameter space mapped into $\{0,1,\ldots,N\}^k$ for the sake of simplicity. Therefore, *parameterSpaceWidth* denotes N and *dim* stands for the parameter space dimension k. Metaparameters: *numberOfSubgrids, gridSize, firstInterspace, initialInterspace, minimumInterspace, k.*

Listing 2. Algorithm pseudocode for Grid method.

```
Set bestValues = {-Inf, -Inf, ..., -Inf} as vector
of size numberOfGrids.
Set bestPoints = {NA, ..., NA} as list containing dim
vectors
of size numberOfSubgrids.

// Check points from the initial grid.

Set gridSize = parameterSpaceWidth / firstInterspace
+ 1.

For j1 = 0 to gridSize - 1 {
 For j2 = 0 to gridSize - 1 {
 ...
 For jdim = 0 to gridSize - 1 {
 // Check point from a current grid.
 Set currentPoint = firstInterspace *
 (j1 * unitVector_1 + j2 * unitVector_2 + ... +
 jdim * unitVector_dim).

 Set currentValue = optimizedFunction(currentPoint).

 If currentValue > bestValues[numberOfSubgrids] and
 currentPoint is element of parameters space {

 // Overwrite the lowest value from bestPoints and
sort.
 Set bestValues[numberOfSubgrids] = currentValue.
 Set bestPoints[numberOfSubgrids] = currentPoint.

 Sort descending bestValues.
 Permute bestPoints accordingly.
 }
 }
 ...
 }
}

// Check subgrids centered at the bestPoints.
Set interspace = initialInterspace.
While interspace >= minimumInterspace {
 For i = 1 to numberOfSubgrids {
 Set center = bestPoints[i].
 For j1 = -(gridSize - 1) / 2 to (gridSize - 1) / 2 {
 For j2 = -(gridSize - 1) / 2 to (gridSize - 1) / 2 {
 ...
 For jdim = -(gridSize - 1) / 2 to (gridSize - 1) /
2 {
 // Check point from a current grid.
 Set currentPoint = center + interspace *
 (j1 * unitVector_1 + j2 * unitVector_2 + ... +
 jdim * unitVector_dim).

 Set currentValue = optimizedFunction(currentPoint).

 If currentValue > bestValues[i] and
 currentPoint is element of parameters space {
 Set bestValues[i] = currentValue.
 Set bestPoints[i] = currentPoint.
 }
 }
 ...
 }
 }
 }

 Set interspace = RoundUp(interspace / k).
}

Return Max(bestValues) and corresponding point from
bestPoints.
```

3.5 Differential Evolution Method (DEM)

The Differential Evolution is the optimization method inspired by the biological phenomenon of evolution. The considered machine learning method follows the approach of taking the random sample (population) from the solution space (Ψ), disturbing the parameters (mutation of the population characteristics) and creating a new sample from the most profitable strategies (reproduction). The steps are repeated – the new population is created, with the disturbed characteristics similar to the best ones from the previous step. This approach is similar to the rule of *the Grid Method*. Both methods repeatedly check all the strategies from a specified sample and create the next one, using the information learned before.

The R implementation of the procedure *JDEoptim* from *DEoptimR* package of version 1.0–8 with default meta parameters (Conceicao, 2016) is used as a main component of the method. The implemented version of algorithm is consistent with jDE method from Brest *et al.* (2006). The function *DEoptim* from package *DEoptim* (Mullen *et al.*, 2016) is implemented in Rcpp in a more efficient way, and moreover, provides bigger elasticity than the *JDEoptim*; therefore, we suggest to use it in practical applications. Nevertheless, in this paper, the method *JDEoptim* was used, because it was fully implemented in R like all other considered methods. Using faster function implemented in Rcpp would give this result unfair advantage over the other ones and could significantly affect the conclusions.

The differential evolution operates on the continuous spaces of real numbers, therefore, it is inadequate for the selection of integer parameters. However, the discrete space can be intrapolated on the continuous one by several methods. The optimization criterion function $OC: \Psi = \{1, 5, \dots, 100\}^4 \rightarrow \mathbb{R}$ was extended to $\widetilde{OC}$ operating on the continuous real space $[0, 100]^4$ in the following way:

$$\widetilde{OC}(x_1, x_2, x_3, x_4) = OC(5max(\lfloor\frac{x_1}{5}\rfloor,1), 5max(\lfloor\frac{x_2}{5}\rfloor,1),\dots,5max(\lfloor\frac{x_4}{5}\rfloor),1)$$

The extended function $\widetilde{OC}$ simply returns the value of OC for the rounded values of parameters with additional assumption that parameters equal to 0 are changed to 1.

3.6 The Exhaustive Search (ES)

The strategies selected by different methods were analysed and compared with the optimal strategy, maximizing the optimization criterion in the in-sample period. The optimal strategy was found in every case by the Exhaustive Search (brute-force) algorithm checking all possible combinations of parameters in order to select one with the highest criterion value. Following this approach always leads to get the highest possible criterion value, but it requires plenty of time. The main purpose of using machine learning methods instead of the Exhaustive Search was to get significantly lower computation time without the loss of quality of results. Therefore, the difference in computation time reflects the value of information learned in previous steps for further search procedure efficiency. Moreover, the Exhaustive Search will be treated as a benchmark due to its simplicity, intuitive character and widespread use.

4 Data description

The main goal of machine learning methods was to find the parameters' vector $\psi \in \Psi = \{1, 5, \dots, 100\}^4$ (i.e. vector of four parameters, each from the set {1, 5, ... , 100}) in order to select the self-financing strategy maximizing *OC* within the framework of the assumptions.

Every considered portfolio was composed of two securities of the same kind. The first pair contained the futures contracts on two important and highly correlated market indexes – American *S&P500 Index (SPX)* and German *Deutscher Aktienindex (DAX)*. The next considered pair was composed of two big American high-tech companies stocks – *Apple Inc. (AAPL)* and *Microsoft Corp. (MSFT)*. These companies are major representatives of the IT sector and American economy, but there was a real difference, between their dynamics of growth. The last considered assets were two commodities' futures contracts – *High-Grade Copper Futures (HG.F)* and *Crude Oil Brent Futures (CB.F)*.

The machine learning methods searched for strategy optimal in the in-sample period from the beginning of 1998 to the end of 2013. Strategies were validated on the out-of-sample data from the beginning of 2014 to the end of 2017. All strategies operated on daily data, taking position each trading day. The length of in-sample period was big enough to make sure that the different market trends were included for all-time series. On the other hand, the out-of-sample length allowed to properly validate strategies and assess overfitting level.

The most rapid growth of value was observed for *AAPL* stock, whose price increased on average by around *35%* annually. The disproportion between the profitability of *AAPL* and other assets did not diminish, after adjusting for risk and applying terms of IR measure. The standard deviation of *AAPL* returns was higher than for any other asset, however, in a way disproportionately lower than the returns. Therefore, one can say that risk was fully compensated by enormously high returns. Additionally, *AAPL* had the lowest maximum drawdown (*43.80%*) among all the considered price series (around *60–70%*).

The returns of *AAPL* was still high, but noticeably lower in the out-of-sample period, although the IR measure was higher than in the in-sample period. The returns of *AAPL* was lower than in the earlier period but associated with much lower risk. Microsoft share price increased faster than the other asset prices in the out-of-sample period. The IR was higher than for any time series in the out-of-sample. It is worth noting, that the commodities' (*HG.F* and *CB.F*) return was negative in the out-of-sample period. The graphs of the prices normalized by the initial value would be presented together with strategies' equity lines in the further part of this paper.

Tab. 1: The descriptive statistics of the considered assets

	In-sample						Out-of-sample					
	SPX	**DAX**	**AAPL**	**MSFT**	**HGF**	**CBF**	**SPX**	**DAX**	**AAPL**	**MSFT**	**HGF**	**CBF**
%ARC	3.92	4.79	35.22	6.32	9.41	12.14	9.67	8.07	22.68	25.70	-0.62	-11.01
%ASD	20.39	24.97	46.69	33.06	28.48	34.54	11.94	18.37	22.27	21.43	19.25	33.07
IR	0.19	0.19	0.75	0.19	0.33	0.35	0.81	0.44	1.02	1.2	-0.03	-0.33
%MDD	56.78	72.68	43.80	71.65	68.37	73.48	14.16	29.27	30.45	18.05	42.47	75.83

%ARC - annualized rate of return (%), %ASD - annualized standard deviation (%), %MDD - maximum drawdown of capital (%), IR - information ratio calculated as %ARC / %ASD, OC - optimization criterion calculated as 100 * (%ARC * %ARC) / (%ASD * %MDD), SPX - S&P500 Index, DAX - Deutscher Aktienindex, AAPL - Apple Inc. stock, MSFT - Microsoft Corp. stock, HGF - High Grade Copper Futures, CBF - Crude Oil Brent Futures. The statistics have been calculated for in-sample period from the beginning of 1998 to the end of 2013 and for out-of-sample period from the beginning of 2013 to the end of 2017, on daily data.

5 Efficiency tests for different methods

Methods described before were tested on the three pairs of assets by running whole optimization process on the data from in-sample period. Extended Hill Climbing and Differential Evolution Method were executed *1000* times for every pair due to their random nature. The strategies with median optimization criterion were treated as final results for comparison purposes and were called *median strategies*. Lengths of moving averages for first asset are denoted by k1 and k2 and analogously k1.2, k2.2 for the second one.

5.1 S&P500 Index (SPX) and Deutscher Aktienindex (DAX)

All statistics and graphs referring to the methods' performance on that pair of assets were denoted by the acronym **SPXDAX**.

5.2 In-sample methods efficiency

All machine learning methods had the same selected median strategy, different than the one resulted from Exhaustive Search procedure. Nevertheless, all the methods used contrarian approach on *SPX* and momentum component operating on *DAX*. Both the optimal strategies resulted from *ES* and median strategy of machine learning methods generated only

Tab. 2: The median strategies parameters and statistics resulted from all the methods for SPXDAX

	In-sample				Out-of-sample			
	ES	**EHC**	**GM**	**DEM**	**ES**	**EHC**	**GM**	**DEM**
k1	60.00	100.00	100.00	100.00	60.00	100.00	100.00	100.00
k2	45.00	35.00	35.00	35.00	45.00	35.00	35.00	35.00
k1.2	65.00	45.00	45.00	45.00	65.00	45.00	45.00	45.00
k2.2	75.00	85.00	85.00	85.00	75.00	85.00	85.00	85.00
%ARC	4.27	3.92	3.92	3.92	-0.03	-0.62	-0.62	-0.62
%ASD	5.17	4.63	4.63	4.63	4.02	3.74	3.74	3.74
IR	0.83	0.85	0.85	0.85	-0.01	-0.17	-0.17	-0.17
%MDD	4.53	4.30	4.30	4.30	7.20	6.34	6.34	6.34
OC	77.79	77.16	77.16	77.16	0.00	-1.62	-1.62	-1.62

%ARC - annualized rate of return (%), %ASD - annualized standard deviation (%), %MDD - maximum drawdown of capital (%), IR - information ratio calculated as %ARC / %ASD, OC - optimization criterion calculated as 100 * (%ARC * %ARC) / (%ASD * %MDD), k1, k2, k1.2, k2.2 - strategy parameters, width of the moving averages' windows. The statistics of the equity lines have been calculated for the strategy working on daily frequency, investing 20% of capital in position on each asset with rebalancing every 5 trading days. Trading from the beginning of 1998 to the end of 2013 has been simulated, with the assumption of fee equal to 0.25% of the position value.

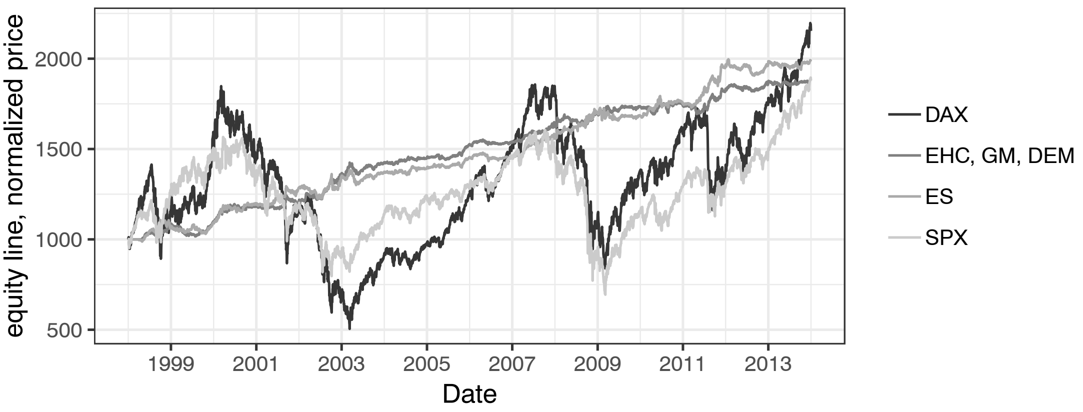

Fig. 1: The equity lines of the strategies selected by all the methods for SPXDAX - in-sample

SPX - S&P500 Index, DAX - Deutscher Aktienindex, ES, EHC, GD, DEM - equity line of the median strategy resulted from respectively Exhaustive Search, Extended Hill Climbing, Grid Method and Differential Evolution Method. Prices of both assets have been normalized in order to have initial value equal to 1000. The equity line has been calculated for the strategy working on daily frequency, investing 20% of capital in position on each asset with rebalancing every 5 trading days. Trading from the beginning of 1998 to the end of 2013 has been simulated, with the assumption of fee equal to 0.25% of the position value.

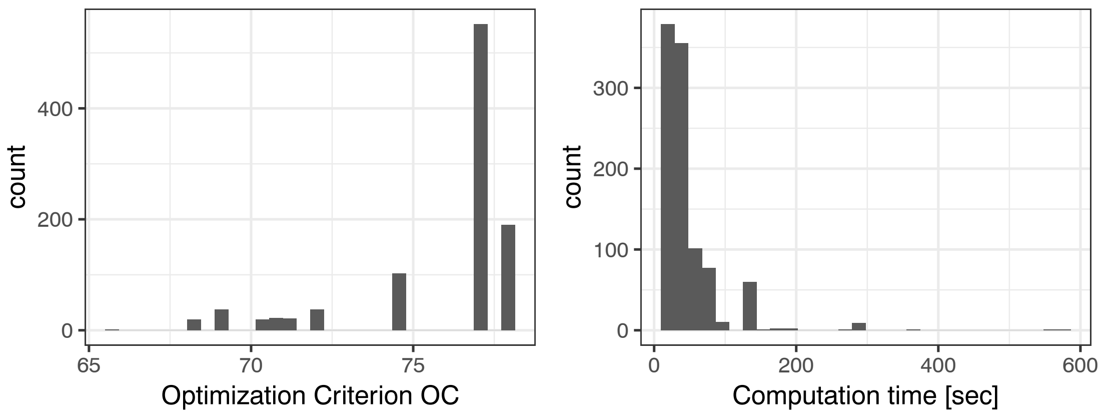

Fig. 2: The histograms of the reached optimization criterion and the execution time of EHC for SPXDAX – in-sample

OC - optimization criterion calculated as 100 * (%ARC * %ARC) / (%ASD * %MDD). The optimization criterion have been calculated from the sample of 1000 independent algorithm executions. The strategies have been working on the daily frequency, investing 20% of capital in position on each asset with rebalancing every 5 trading days. Trading from the beginning of 1998 to the end of 2013 has been simulated, with assumption of fee equal to 0.25% of the position value.

moderate profits during the whole in-sample period. On the other hand, the strategy met the requirements of stability and safety. Strategies of that kind were preferred over the more profitable ones due to using low leverage and including two risk measures in the construction of the optimization criterion. The maximum drawdown was especially low, despite the relatively long time horizon. The resulting strategies never lost more than *5%* of the available money during whole 16 years of the in-sample period, comparing with more than *50%* on the basis instruments (SPX and DAX indexes).

The strategy components could hedge each other in order to reduce the portfolio risk and obtain more smooth equity line (stable profits). Both strategies followed two opposite approaches in trading on two similar assets. The strategy contained the contrarian part, operating on *SPX* and the momentum one trading on *DAX*. The Exhaustive Search reached the global maxima, but the total calculation time was equal to 609.37 minutes.

The empirical distributions of the reached criterion and computation time of *1000* independent *EHC* and respectively *DEM* executions are presented on Fig. 2 and Fig. 3. Most of the independent procedures for both methods returned the same strategy with the second best optimization criterion. No selected solution had significantly worse performance and considerable number

Tab. 3: The summary of the reached optimization criterion and the execution time of methods for SPXDAX – in-sample

	ES		EHC		GM		DEM	
	OC	Time [sec]	OC	Time [sec]	OC	Time [sec]	OC	Time [sec]
Minimum	77.79	35562.17	65.58	11.87	77.16	128.04	71.93	13.11
1st Quantile	77.79	35562.17	74.39	13.93	77.16	128.04	77.16	24.84
Median	77.79	35562.17	77.16	30.97	77.16	128.04	77.16	31.15
Mean	77.79	35562.17	75.94	43.1	77.16	128.04	77.34	42.73
2nd Quantile	77.79	35562.17	77.16	65.32	77.16	128.04	77.79	61.5
Max	77.79	35562.17	77.79	569.39	77.16	128.04	77.79	141.08
Standard deviation	0.00	0.00	2.61	48.77	0.00	0.00	0.36	24.06

OC - optimization criterion calculated as 100 * (%ARC * %ARC) / (%ASD * %MDD). The equity lines have been calculated for the strategy working on daily frequency, investing 20% of capital in position on each asset with rebalancing every 5 trading days. Trading from the beginning of 1998 to the end of 2013 has been simulated, with the assumption of fee equal to 0.25% of the position value.

of them reached the highest criterion as well. Therefore, the results proved both high efficiency and stability of the methods (Tab. 3). The median of the Extended Hill Climbing procedure computation time is equal to *30.97* seconds, when the Exhaustive Search took more than *10* hours. The computation time improvement is indisputable. Due to the dynamic stopping rule, the execution time was highly varied across the sample. Some runs lasted around *11* seconds, when the others took around *9* minutes. Nevertheless, the observed low level of stability did not affect the time advantage over the Exhaustive Search, because all the procedures lasted incomparably less.

The Grid Method resulted in the second best strategy, exactly the same as median strategy from *the Extended Hill Climbing*. The calculation time was longer than for the previous method. On the other hand, the execution lasted *128.04* seconds, which was still much less than for *the Exhaustive Search*. The method had some advantages over the previous, machine learning method as well. The optimization criterion was similar, but there was no uncertainty either about results or time, while the *EHC* and *DEM* results was random.

Most of the *DEM* executions selected exactly the same strategy, as two previous methods. The median execution time for *DEM* equalled to *31.15* seconds and the time required to execute the procedure had lower standard deviation than for *EHC*. The *Differential Evolution Method* gave strategies similar to the optimal ones but in relatively short and stable time.

5.2.1 Out-of-sample results

As expected, the out-of-sample strategy performance was worse than the in-sample period. The strategies obtained by the Exhaustive Search and all the considered machine learning methods were ineffective in the out-of-sample period and resulted in return close to zero at the end of a time horizon.

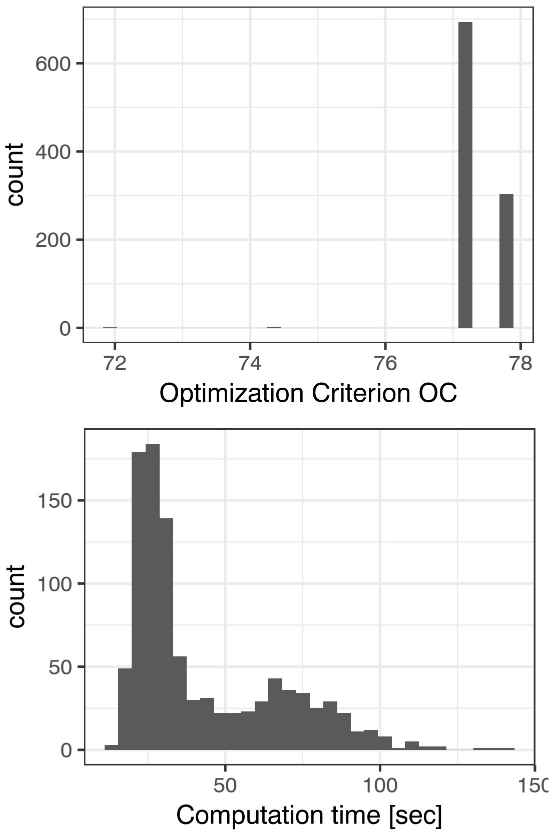

Fig. 3: The histograms of the reached optimization criterion and the execution time of DEM for SPXDAX - in-sample

OC - optimization criterion calculated as 100 * (%ARC * %ARC) / (%ASD * %MDD). The optimization criterion have been calculated for the strategy working on daily frequency, investing 20% of the capital in position on each asset with rebalancing every 5 trading days. Trading from the beginning of 1998 to the end of 2013 has been simulated, with the assumption of fee equal to 0.25% of the position value.

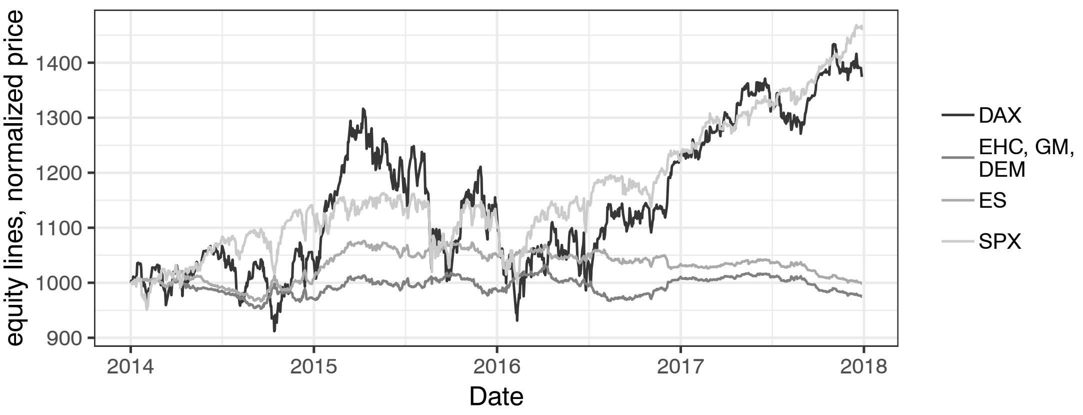

Fig. 4: The equity lines of the strategies selected by the all the methods for SPXDAX – out-of-sample

ES, EHC, GD, DEM - equity line of the median strategy resulted from respectively Exhaustive Search, Extended Hill Climbing, Grid Method and Differential Evolution Method. Prices of both assets have been normalized in order to have initial value equal to 1000. The equity line has been calculated for the strategy working on daily frequency, investing 20% of capital in position on each asset with rebalancing every 5 trading days. Trading from the beginning of 2014 to the end of 2017 has been simulated, with the assumption of fee equal to 0.25% of the position value.

5.3 Apple Inc. (AAPL) and Microsoft Corp. stock (MSFT)

The strategy was optimized for the stocks of high-tech companies *Apple Inc.* and *Microsoft Corp*. The dynamic growth of prices was a great trading opportunity, therefore the strategies were able to generate high profit in both periods. Similar to the previous pair, the high correlation between prices was observed both in the in-sample and in the out-of-sample periods, what gave the possibility to design strategies with hedging elements and obtain results associated with a lower risk. All the statistics and graphs referring to that case were denoted by *AAPL MSFT*.

5.3.1 In-sample methods efficiency

All the considered methods selected exactly the same strategy. Simple moving averages crossover approach was highly effective due to the enormously high growth of the Apple stock. That strategy had a large return in the in-sample period, *17.79%* annually, and low risk measures as well. The annualized standard deviation of returns was equal to *11.13%* when the maximum drawdown was lower than *8%*. The return generated by the strategy was high but incomparably lower than the percentage growth of *AAPL*. Nevertheless, buy-and-hold strategy following the specified rebalancing rule with leverage at level *40%* generated a return of *13.59%* with annualized standard deviation equal to *9.24%*

Tab. 4: The median strategy parameters and statistics resulted from all the methods for AAPLMSFT

	In-sample				**Out-of-sample**			
	ES	**EHC**	**GM**	**DEM**	**ES**	**EHC**	**GM**	**DEM**
k1	50.00	50.00	50.00	50.00	50.00	50.00	50.00	50.00
k2	60.00	60.00	60.00	60.00	60.00	60.00	60.00	60.00
k1.2	75.00	75.00	75.00	75.00	75.00	75.00	75.00	75.00
k2.2	40.00	40.00	40.00	40.00	40.00	40.00	40.00	40.00
%ARC	17.79	17.79	17.79	17.79	1.37	1.37	1.37	1.37
%ASD	11.13	11.13	11.13	11.13	5.68	5.68	5.68	5.68
IR	1.60	1.60	1.60	1.60	0.24	0.24	0.24	0.24
%MDD	7.71	7.71	7.71	7.71	9.54	9.54	9.54	9.54
OC	368.83	368.83	368.83	368.83	3.49	3.49	3.49	3.49

%ARC - annualized rate of return (%), %ASD - annualized standard deviation (%), %MDD - maximum drawdown of capital (%), IR - information ratio calculated as %ARC / %ASD, OC - optimization criterion calculated as 100 * (%ARC * %ARC) / (%ASD * %MDD), k1, k2, k1.2, k2.2 - strategy parameters, width of the moving averages' windows. The statistics of the equity line have been calculated for the strategy working on daily frequency, investing 20% of capital in position on each asset with rebalancing every 5 trading days. Trading from the beginning of 1998 to the end of 2013 has been simulated, with the assumption of fee equal to 0.25% of the position value.

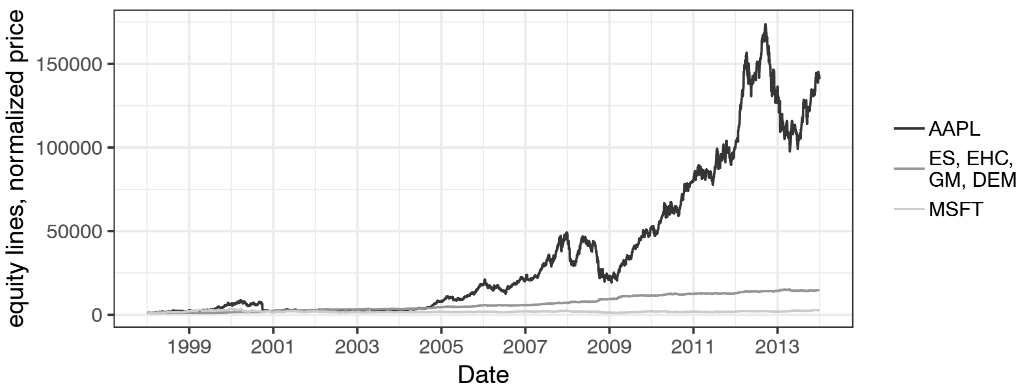

Fig. 5: The equity line of the strategy selected by all the methods for AAPLMSFT – in-sample

AAPL - Apple Inc. stock, MSFT - Microsoft Corp. stock, ES, EHC, GD, DEM - equity line of the median strategy resulted from respectively Exhaustive Search, Extended Hill Climbing, Grid Method and Differential Evolution Method. Prices of both assets have been normalized in order to have initial value equal to 1000. The equity line has been calculated for the strategy working on daily frequency, investing 20% of capital in position on each asset with rebalancing every 5 trading days. Trading from the beginning of 1998 to the end of 2013 has been simulated, with the assumption of fee equal to 0.25% of the position value.

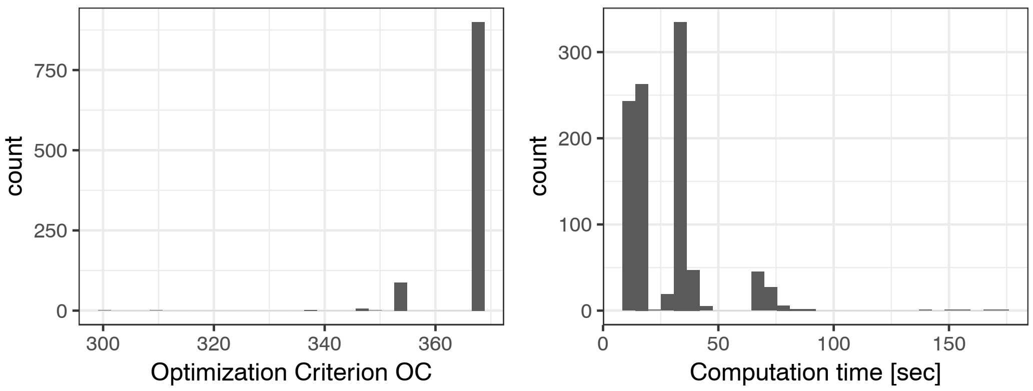

Fig. 6: The histograms of the reached optimization criterion and the execution time of EHC for AAPLMSFT – in-sample

OC - optimization criterion calculated as 100 * (%ARC * %ARC) / (%ASD * %MDD). The optimization criterion has been calculated from the sample of 1000 independent algorithm executions. The strategies have been working on the daily frequency, investing 20% of capital in position on each asset with rebalancing every 5 trading days. Trading from the beginning of 1998 to the end of 2013 has been simulated, with the assumption of fee equal to 0.25% of the position value.

and the maximum drawdown of *9.71%*. Therefore, the selected strategy is both highly profitable and associated with relatively low risk as well. The Exhaustive Search lasted 547.02 minutes in that case.

The Extended Hill Climbing method returned exactly the same results as the Exhaustive Search, but in significantly lower time. There were a number of similarities between the method performance in the current and the previous case. The Extended Hill Climbing had the high stability of results, but uncertain computation time. Despite the low time stability, the method proved to be far more efficient than the Exhaustive Search. The median of execution time was equal to *18.18* seconds (Tab. 5).

The Grid Method resulted in exactly the same strategy as both previous methods. The computation time was equal to 150.66 seconds.

The Differential Evolution reached the global maxima in almost every attempt. Moreover, the median of the execution time was higher (*22.19*) than for *the Extended Hill Climbing*, but the execution time had a lower standard deviation. Therefore, the differential evolution proved to be an efficient and a stable method in that case. The median strategy was the optimal one, the same as for all other considered methods.

Tab. 5: The summary of the reached optimization criterion and the execution time of methods for AAPLMSFT – in-sample

	ES		**EHC**		**GM**		**DEM**	
	OC	**Time [sec]**	**OC**	**Time [sec]**	**OC**	**Time [sec]**	**OC**	**Time [sec]**
Minimum	368.83	32821.18	301.34	11.96	368.83	150.66	274.97	11.82
1st Quantile	368.83	32821.18	368.83	14	368.83	150.66	368.83	19.67
Median	368.83	32821.18	368.83	18.18	368.83	150.66	368.83	22.19
Mean	368.83	32821.18	367.16	27.4	368.83	150.66	368.55	22.71
2nd Quantile	368.83	32821.18	368.83	32.97	368.83	150.66	368.83	25.27
Max	368.83	32821.18	368.83	174.06	368.83	150.66	368.83	45.8
Standard deviation	0.00	0.00	5.51	18.71	0.00	0.00	5.14	4.52

OC - optimization criterion calculated as 100 * (%ARC * %ARC) / (%ASD * %MDD). The optimization criterion has been calculated from the sample of 1000 independent algorithm executions. The strategies have been working on the daily frequency, investing 20% of capital in position on each asset with rebalancing every 5 trading days. Trading from the beginning of 1998 to the end of 2013 has been simulated, with the assumption of fee equal to 0.25% of the position value.

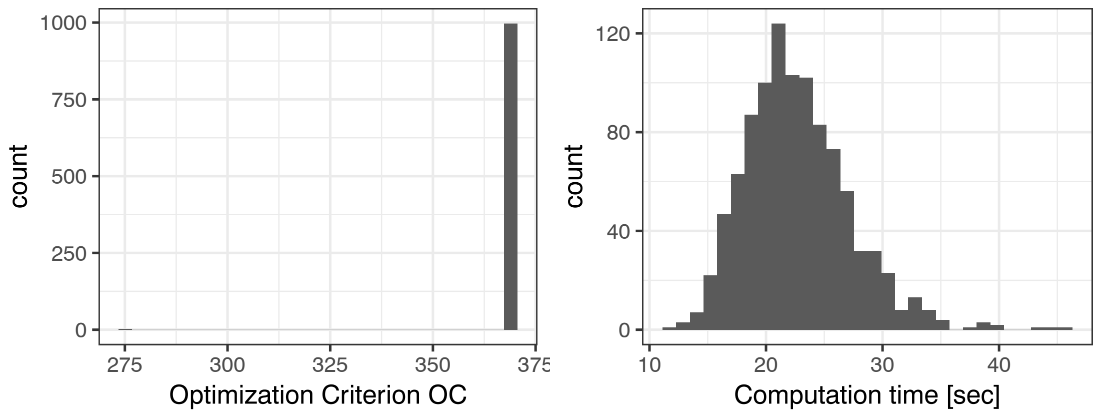

Fig. 7: The histograms of the reached optimization criterion and the execution time of DEM for AAPLMSFT – in-sample
OC - optimization criterion calculated as 100 * (%ARC * %ARC) / (%ASD * %MDD). The optimization criterion have been calculated from the sample of 1000 independent algorithm executions. The strategies have been working on the daily frequency, investing 20% of capital in position on each asset with rebalancing every 5 trading days. Trading from the beginning of 1998 to the end of 2013 has been simulated, with the assumption of fee equal to 0.25% of the position value.

5.3.2 The out-of-sample results

All the considered machine learning methods selected the same strategy as the exhaustive search procedure. That strategy was optimal on the in-sample period in terms of optimization criterion, resulted in annualized returns at the level of *17.79%* and relatively low both annualized standard deviation (%*ASD*) and maximum drawdown (%*MDD*). Performance of the selected strategy on the out-of-sample period was substantially worse than on the in-sample period. Strategy had more than *10* times lower %*ARC*, similar %*MDD* and a little lower %*ASD* (Tab. 4). A big difference between performance on the consecutive periods suggested high level of strategy overfitting, however, strategy generated positive profits in the out-of-sample period.

5.4 High Grade Copper Futures (HG.F) and Crude Oil Brent Futures (CB.F)

The last considered pair of assets was composed of the two commodities' futures contracts. The problem of finding the optimal strategy was harder, than for the previous ones. The difference between commodities' behaviour in both periods and a weaker statistical

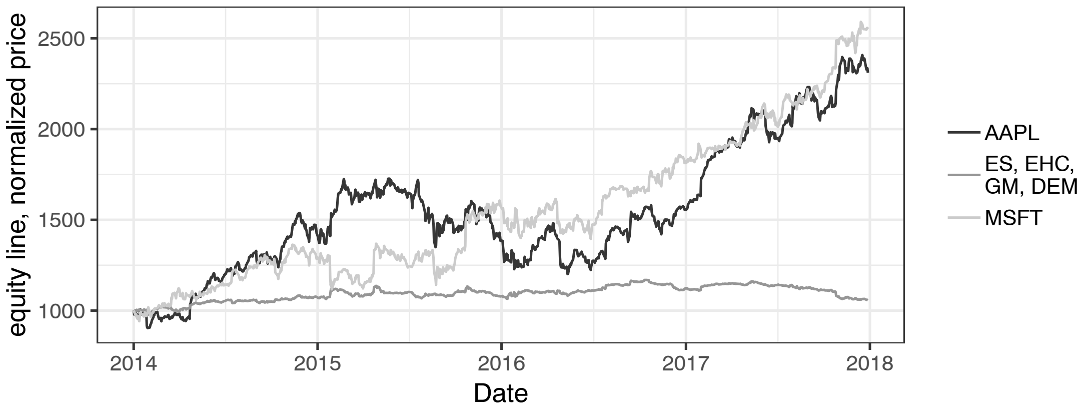

Fig. 8: The equity lines of the strategies selected by all the methods for AAPLMSFT – out-of-sample

AAPL - Apple Inc. stock, MSFT - Microsoft Corp. stock, ES, EHC, GD, DEM - equity line of the median strategy resulted from respectively Exhaustive Search, Extended Hill Climbing, Grid Method and Differential Evolution Method. Prices of both assets have been normalized in order to have initial value equal to 1000. The equity line has been calculated for the strategy working on daily frequency, investing 20% of capital in position on each asset with rebalancing every 5 trading days. Trading from the beginning of 2014 to the end of 2017 has been simulated, with the assumption of fee equal to 0.25% of the position value.

Tab. 6: The median strategies parameters and statistics resulted from all the methods for HGFCBF

	In-sample				**Out-of-sample**			
	ES	**EHC**	**GM**	**DEM**	**ES**	**EHC**	**GM**	**DEM**
k1	60.00	60.00	60.00	60.00	60.00	60.00	60.00	60.00
k2	75.00	75.00	75.00	75.00	75.00	75.00	75.00	75.00
k1.2	50.00	30.00	30.00	30.00	50.00	30.00	30.00	30.00
k2.2	25.00	95.00	95.00	95.00	25.00	95.00	95.00	95.00
%ARC	8.18	9.53	9.53	9.53	-1.59	6.60	6.60	7.38
%ASD	8.16	9.83	9.83	9.83	7.09	8.17	8.17	8.07
IR	1.00	0.97	0.97	0.97	-0.22	0.81	0.81	0.91
%MDD	7.51	9.52	9.52	9.52	15.86	12.16	12.16	12.16
OC	109.11	97.04	97.04	97.04	-2.24	43.81	43.81	55.49

OC - optimization criterion calculated as 100 * (%ARC * %ARC) / (%ASD * %MDD). The empirical statistics have been calculated from the sample of 1000 independent algorithm executions. The strategies have been working on the daily frequency, investing 20% of capital in position on each asset with rebalancing every 5 trading days. Trading from the beginning of 1998 to the end of 2013 has been simulated, with the assumption of fee equal to 0.25% of the position value.

relationship between them were the main reasons for the difficulties. The statistics and graphs from that case were always denoted by *HGFCBF*.

5.4.1 In-sample methods efficiency

The Exhaustive Search selected the strategy with an annualized return equal to *8.18%*, when all the machine learning methods selected median strategy with returns at level of *9.53%*. Nevertheless, that strategy was optimal in terms of optimization criterion, depending on

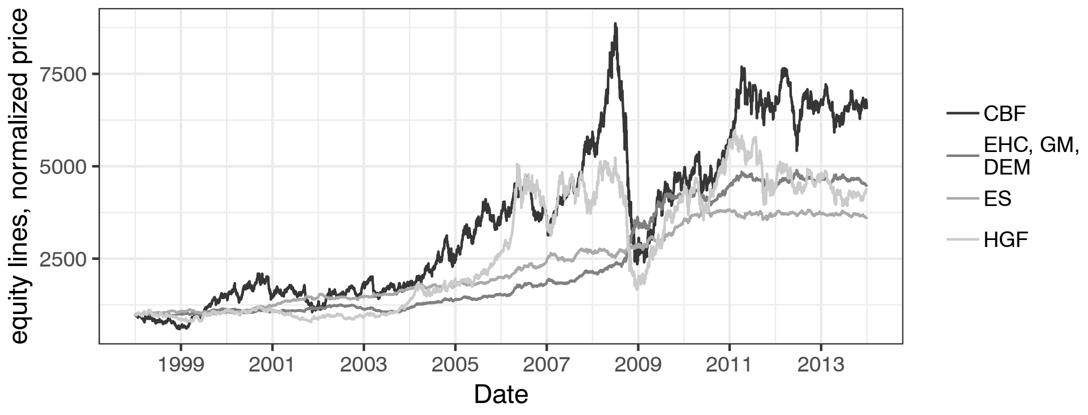

Fig. 9: The equity lines of the strategy selected by all the methods for HGFCBF – in-sample

HGF - High Grade Copper Futures, CBF - Crude Oil Brent Futures, ES, EHC, GD, DEM - equity line of the median strategy resulted from respectively Exhaustive Search, Extended Hill Climbing, Grid Method and Differential Evolution Method. Prices of the both assets have been normalized in order to have initial value equal to 1000. The equity line has been calculated for the strategy working on daily frequency, investing 20% of capital in position on each asset with rebalancing every 5 trading days. Trading from the beginning of 1998 to the end of 2013 has been simulated, with the assumption of fee equal to 0.25% of the position value.

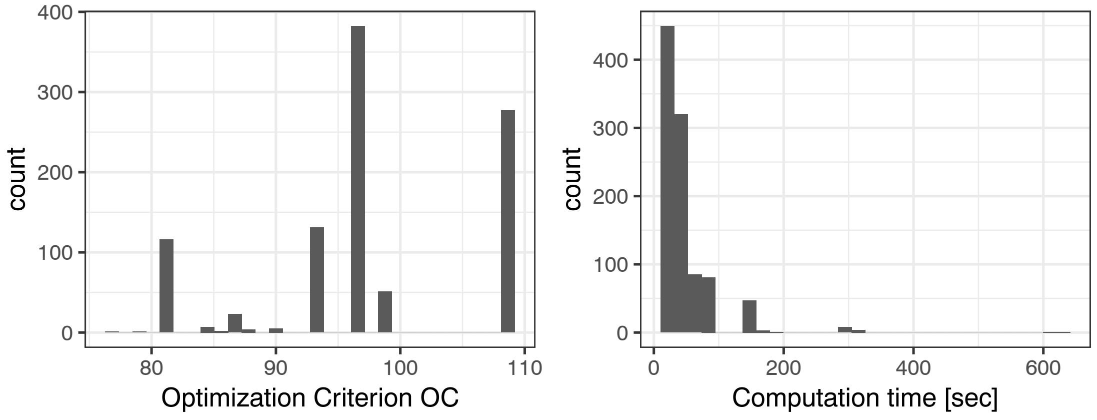

Fig. 10: The histograms of the reached optimization criterion and the execution time of EHC for HGFCBF – in-sample

OC - optimization criterion calculated as 100 * (%ARC * %ARC) / (%ASD * %MDD). The optimization criterion has been calculated from the sample of 1000 independent algorithm executions. The strategies have been working on the daily frequency, investing 20% of capital in position on each asset with rebalancing every 5 trading days. Trading from the beginning of 1998 to the end of 2013 has been simulated, with the assumption of fee equal to 0.25% of the position value.

the returns and risk measures as well. The Exhaustive Search process took 703.226 minutes.

The conclusions from computing *1000* procedures of *EHC* were the same as for the previous cases (Tab. 7). The reached optimization criterion was rather stable, in contrast to an unstable computation time. Once again, all the learning procedures took a far less time than the Exhaustive Search (median of execution time was equalled *33.07* seconds).

FThe *GM* method returned the median strategy of *EHC* and its computation time lasted 113.752 seconds. The conclusions were consistent with those discussed in the previous cases. The method obtained good results in the fixed time, when the *Extended Hill Climbing* optimization gave similar results in shorter, but more random time. The Grid Method resulted in a strategy near to optimal with a reasonable time of execution.

The median of *DEM* criteria was the same as for *the Extended Hill Climbing*. The computation time of the *Differential Evolution* had the higher stability than *EHC* and the lowest median value among all methods (*23.08* seconds).

Tab. 7: The summary of the reached optimization criterion and the execution time of the methods for HGFCBF – in-sample

	ES		EHC		GM		DEM	
	OC	**Time [sec]**	**OC**	**Time [sec]**	**OC**	**Time [sec]**	**OC**	**Time [sec]**
Minimum	109.11	42193.57	77.29	12.29	97.04	113.75	97.04	9,76
1st Quantile	109.11	42193.57	93.62	14.73	97.04	113.75	97.04	19.92
Median	109.11	42193.57	97.04	33.09	97.04	113.75	97.04	23.08
Mean	109.11	42193.57	97.82	42.69	97.04	113.75	99.01	27.34
2nd Quantile	109.11	42193.57	109.11	36.85	97.04	113.75	97.04	29.15
Max	109.11	42193.57	109.11	622.17	97.04	113.75	109.11	110.93
Standard deviation	0.00	0.00	8.59	51.28	0.00	0.00	4.46	12.5

OC - optimization criterion calculated as 100 * (%ARC * %ARC) / (%ASD * %MDD). The optimization criterion has been calculated from the sample of 1000 independent algorithm executions. The strategies have been working on the daily frequency, investing 20% of capital in position on each asset with rebalancing every 5 trading days. Trading from the beginning of 1998 to the end of 2013 has been simulated, with the assumption of fee equal to 0.25% of the position value.

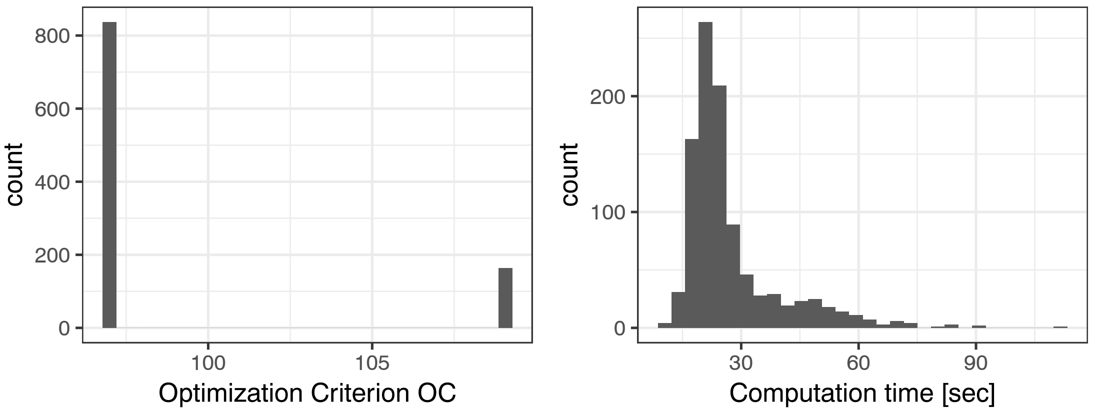

Fig. 11: The histograms of the reached optimization criterion and the execution time of DEM for HGFCBF – in-sample

OC - optimization criterion calculated as 100 * (%ARC * %ARC) / (%ASD * %MDD). The optimization criterion has been calculated from the sample of 1000 independent algorithm executions. The strategies have been working on the daily frequency, investing 20% of capital in position on each asset with rebalancing every 5 trading days. Trading from the beginning of 1998 to the end of 2013 has been simulated, with the assumption of fee equal to 0.25% of the position value.

5.4.2 The out-of-sample results

All the considered machine learning methods finally selected the same strategy, which was slightly worse than the optimal one in the in-sample period but significantly better in the out-of-sample period. The annualized returns were about *6.60%* with reasonable standard deviation and maximum drawdown. It is worth noticing that the strategy with the highest optimization criterion was omitted by all machine learning methods, probably because of the low stability of criterion around that point. The parameters' vector was probably surrounded by the low-value ones, and therefore, machine learning methods could not find it. In consequence, the method

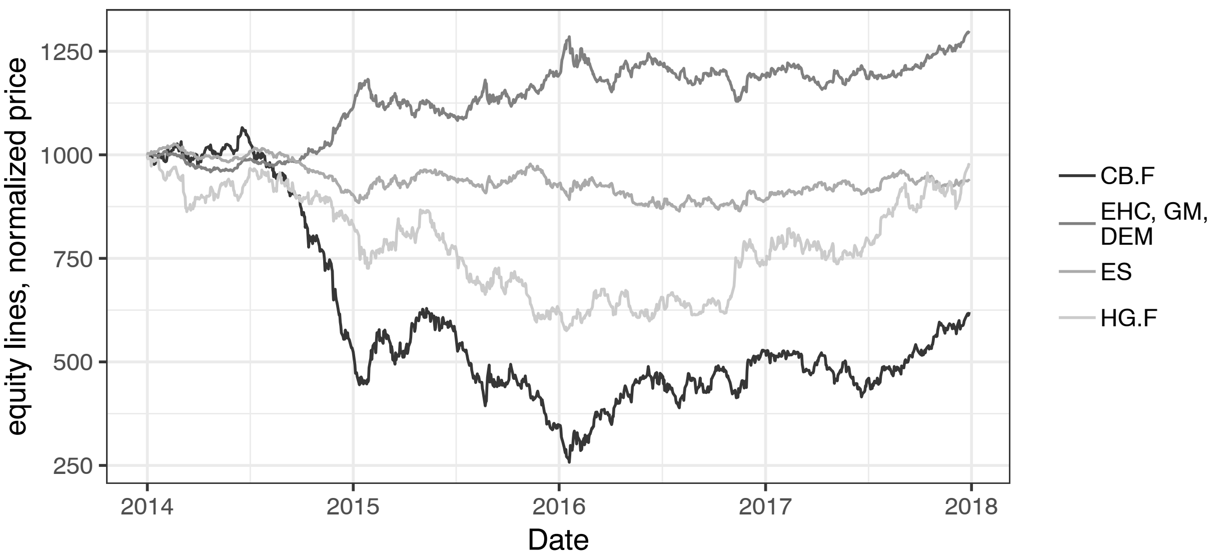

Fig. 12: The equity lines of the strategies selected by the different methods for HGFCBF – out-of-sample

SPX - S&P500 Index, DAX - Deutscher Aktienindex, ES, EHC, GD, DEM - equity line of the median strategy resulted from respectively Exhaustive Search, Extended Hill Climbing, Grid Method and Differential Evolution Method. Prices of the both assets have been normalized in order to have initial value equal to 1000. The equity line has been calculated for the strategy working on daily frequency, investing 20% of capital in position on each asset with rebalancing every 5 trading days. Trading from the beginning of 2014 to the end of 2017 has been simulated, with the assumption of fee equal to 0.25% of the position value.

selected the point from the more stable neighbourhood, what resulted in better performance in the out-of-sample period, what showed a potential of machine learning in reducing overfitting risk.

5.5 The comparison of tested methods with the Exhaustive Search

Throughout the paper, three machine learning optimization methods, adjusted to the problem specificity, were discussed. The performance of each method was tested by solving three problems of selection of trading strategy parameters on the period from the beginning of 1998 to the end of 2017. The machine learning algorithms solved the problem in significantly shorter time than the Exhaustive Search procedure with no significant difference in the results' quality.

As noted before, the machine learning methods gave results similar to the optimal ones obtained by the Exhaustive Search procedure. The critical difference was in the computation time. Checking all the possible parameters required plenty of time. It lasted a few hours, whereas the machine learning methods produced the comparable results in a fraction of a minute. The advantage in time efficiency would be critical for complex problems, for instance, considering a larger parameter space. The relative time difference was significant, for instance, *DEM* took over *1650* times less time than the full exhaustive procedure in the *SPXDAX* and *AAPLMSFT* cases. Assuming the same proportion, the *DEM* optimization, requiring less than one hour could replace the *ES* lasting two months. The results obtained in the paper suggested that machine learning methods introduced before could be an effective replacement for the Exhaustive Search, reducing the computation time without affecting the quality of results (Tab. 8 and Tab. 9).

The first box plot (Fig. 13) presents the optimization criterion across the samples. There are almost no significant differences between the results of the tested methods. The second box plot (Fig. 14) presents the computation time across the samples. The *GM* had substantially higher median time, compared to the other machine learning methods, but without uncertainty. The time required for *DEM* execution was relatively low and stable, especially compared with *EHC*, which had lower time stability, which was illustrated on Fig. 14 by the box size and plenty of outliers.

Tab. 8: Mean and median optimization criterion reached by the different methods, referred to the ES method in percent – in-sample

	ES	Grid	EHC median	DEM median	EHC mean	DEM mean
SPXDAX	100	99.19	99.19	99.19	97.62	99.42
AAPLMSFT	100	100.00	100.00	100.00	99.55	99.92
HGFCBF	100	88.94	88.94	88.94	89.65	90.74

SPXDAX - case of trading on S&P500 Index and Deutscher Aktienindex, AAPLMSFT - case of trading on Apple Inc. and Microsoft Corp. stocks, HGFCBF - case of trading on High-Grade Copper and Crude Oil futures. ES - the Exhaustive Search, EHC - the Extended Hill Climbing, DEM - the Differential Evolution. The simulations has been performed for the strategy working on daily frequency, investing 20% of capital in position on each asset with rebalancing every 5 trading days. Trading from the beginning of 1998 to the end of 2013 has been simulated, with the assumption of fee equal to 0.25% of the position value.

Tab. 9: Mean and median computation time of the methods, referred to the ES method in percent

	ES	Grid	EHC median	DEM median	EHC mean	DEM mean
SPXDAX	100	0.35	0.08	0.09	0.12	0.12
AAPLMSFT	100	0.46	0.06	0.07	0.08	0.07
HGFCBF	100	0.27	0.08	0.05	0.10	0.06

SPXDAX - case of trading on S&P500 Index and Deutscher Aktienindex, AAPLMSFT - case of trading on Apple Inc. and Microsoft Corp. stocks, HGFCBF - case of trading on High-Grade Copper and Crude Oil futures. ES - the Exhaustive Search, EHC - the Extended Hill Climbing, DEM - the Differential Evolution. The simulations has been performed for the strategy working on daily frequency, investing 20% of capital in position on each asset with rebalancing every 5 trading days. Trading from the beginning of 1998 to the end of 2013 has been simulated, with the assumption of fee equal to 0.25% of the position value.

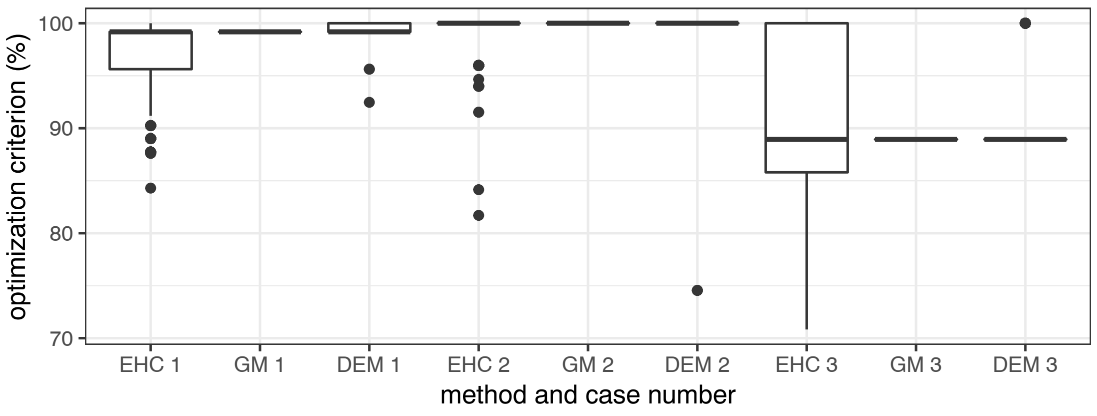

Fig. 13: The boxplot of the optimization criterion of strategies selected by the machine learning methods, as a percentage of the global maxima found by the Exhaustive Search

The samples were denoted by the algorithm acronym and the number of trading case, so *1, 2* and *3* stands for respectively *SPXDAX*, *AAPLMSFT* and *HGFCBF*. The box plots present the empirical distribution quartiles and highlight the outliers. Half of the observations are inside the corresponding box, when the line inside marks the median. The observation was considered as an outlier and marked by a circle if the distance from both first and third quartile (from the nearest side of the box) was higher than *1.5* interquartile range. The range of observations, without outliers was marked by the whiskers. That type of box plot was often called *the Turkey Box Plot*. It was worth to notice that the box plots of the *Grid Method* results were just a line because the results of that method were deterministic.

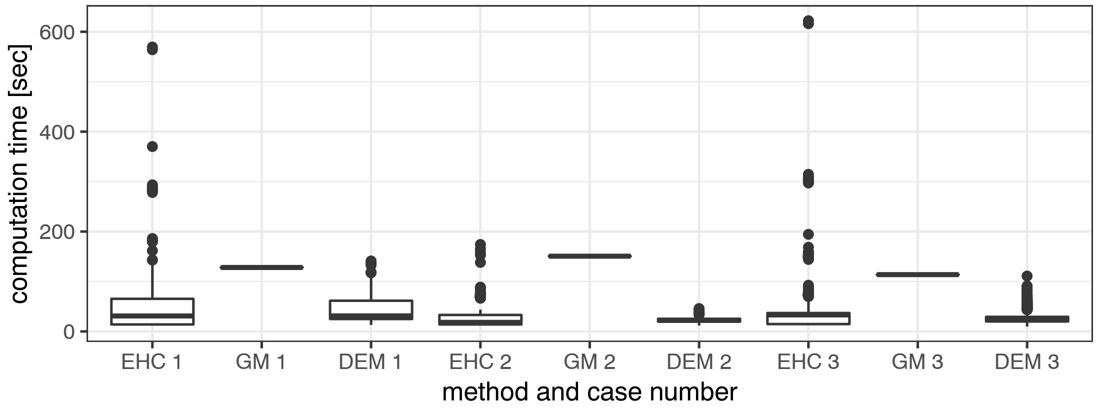

Fig. 14: The boxplot of machine learnings methods' computation time empirical distribution

The samples were denoted by the algorithm acronym and the number of trading case, so *1, 2* and *3* stands for respectively *SPXDAX*, *AAPLMSFT* and *HGFCBF*. The box plots presents the empirical distribution quartiles and highlight the outliers. A half of the observations are inside the corresponding box, when the line inside marks the median. The observation was considered as an outlier and marked by a circle if the distance from both first and third quartile (from the nearest side of the box) was higher than *1.5* interquartile range. The range of observations, without the outliers was marked by the whiskers. That type of box plot was often called *the Turkey Box Plot*. It was worth to notice that the box plots of *Grid Method* results were just a line because the results of that method were deterministic.

6 Conclusions

Three machine learning methods (EHC, GM and DEM) were implemented and tested on simple moving averages' crossover strategy optimization problem. Machine learning methods were a heuristic searches, based on simple algorithms, commonly used for similar problems. The methods were adjusted to the considered problem specificity, such as discreteness of parameters or low regularity of the solution space.

Machine learning methods were compared based on the value of optimization criterion, including annualized rate of return from strategy and two risk measures – the annualized standard deviation and the maximum drawdown. All the statistics were calculated for the simulated trading on the period from the beginning of 1998 to the end of 2013. The optimization criterion calculated for the strategies and the computation time, required to proceed the whole search process, were compared with the Exhaustive Search. The considered strategies were traded on the specified pairs of assets and were tested separately on *SPX* and *DAX* indexes futures, *AAPL* and *MSFT* stocks, and finally, on the pair composed of two commodity futures – *HG.F* and *CB.F*.

The strategies were compared, in terms of the optimization criterion, based on the annualized returns and including the risk metrics, such as the annualized standard deviation of returns and the maximum drawdown of the equity line. Applying such an approach in the optimization process led to the selection of more stable strategies. Using maximum drawdown component eliminated the strategies generating all profits in one short period of time. That approach significantly reduced the risk of overfitting, caused by the adjustment strategy to a few past extreme market situations.

The first method, called the *Extended Hill Climbing* was composed of the independent local search walks, starting in the randomly drawn points with specified stopping rule, based on the level of optimization criterion improvement in the previous steps. That method generated stable results, which means that the strategies returned by the different program executions should be similar to each other. The method produced results comparable to the optimal one in relatively short time, but the stability of the execution time was low. On average, the method was quick and efficient, but the time of the whole process was varied.

The second implemented ML method was purely deterministic algorithm, called the *Grid Method*. The main idea of the search was to use denser and denser subgrids, centred at the points with high optimization criterion value. The method returned the strategy parameters, with optimization criterion similar to the optimal one with time a few times longer than two other ML methods, but still a few times shorter than the full exhaustive procedure. The big advantage of that method is the stable computation time and results, which came from its deterministic nature. This property of the optimization procedure could be appreciated especially for usage in more complex, automatic systems.

The last method, called *the Differential Evolution*, is in fact one of the most popular heuristic algorithm to solve irregular continuously parameterized problems, adjusted to the specificity of integer parameter spaces. The adjustment was based on the transformation of the discrete solution space into the continuous one, in a way preserving the problem specificity.

The performance of strategies in the in-sample period was better than in the out-of-sample. Despite the main goal was to introduce and compare optimization methods, it is worth to point out the difference between in-sample and out-of-sample strategies' accuracy. The strategies optimized by different methods in the in-sample periods bear losses in the out-of-sample period for two out of three cases (*SPXDAX* and *AAPLMSFT*). The unsatisfactory results during the second period led us to the conclusion, that the selected strategies were not supposed to generate profit in the future. The considered models had relatively few parameters, but it was enough to produce an overfitted strategy, too well-adjusted to the training data, and in consequence, ineffective on the test set.

Slightly different situation was in the case of out-of-sample results for commodity futures trading *(HG.F CB.F)*. In that case, all the tested machine learning methods omitted the strategy with the higher optimization criterion in the in-sample period, probably because of the weak performance of neighbouring strategies. In consequence, the strategy selected by all the methods (in fact strategy with median optimization criterion across the sample) performed well in the out-of-sample period, generating profits, while the one with the highest optimization criterion was bearing losses. It seems to confirm the basic intuition – the model avoided overfitting to the training dataset, which caused worse performance there, but also gave a chance to get better results in the future. Selected simple moving averages crossover strategies were generally not profitable on the price time series from outside the training set, but there was a significant premise; the machine learning methods developed in this paper, could be used to optimize trading systems, based on another logic and significantly improve its computation time. The optimization time is crucial, because the shorter the time, the faster the results of the optimization are available for a supervisor or the wider space of parameters and more sophisticated systems can be fitted in an efficient way.

To sum up, the presented results seems to be consistent with the main hypothesis. The machine learning methods required much less time than the Exhaustive Search and produced similar results in the considered cases. In consequence, the main hypothesis was not rejected. The machine learning methods reached only slightly worse in-sample optimization criterion but in a significantly lower execution time. The additional research question, that the machine learning methods leads to lower overfitting risk, could not be answered based on the results presented in this paper. In two scenarios, the machine learning methods selected very similar strategies to the optimal one. Nevertheless, the methods selected worse strategies in the in-sample period in the last case; the final strategy generated profit in the out-of-sample period, while the one obtained by the ES resulted in the loss of the invested capital. The property of the overfitting reduction was observed only in one case, so it cannot lead to certain conclusions.

Bibliography

[1] Ardia D., Boudt K., Carl P., Mullen K. M., Peterson B.G. *Differential Evolution with DEoptim: An Application to Non-Convex Portfolio Optimization.* The R Journal, 2010.

[2] Ardia D., Boudt K., Carl P., Mullem K. M., Peterson B.G. *Large-scale portfolio optimization with DEoptim*. CRAN R, 2011a.

[3] Breiman L. *Statistical Modeling: The Two Cultures*. Statistical Science 2001, Vol. 16, No. 3, Pages 199–231, 2001.

[4] Brest J. *et al. Self-Adapting Control Parameters in Differential Evolution: A Comparative Study on Numerical Benchmark Problems*. IEEE Transactions on Evolutionary Computation, Volume 10, Issue 6, 2006.

[5] Choundhry R., Kumkum G. *A Hybrid Machine Learning System for Stock Market Forecasting*. International Journal of Computer and Information Engineering Vol:2, No:3, 2008.

[6] Conceicao E. *Differential Evolution Optimization in Pure R*. CRAN R Project, 2016.

[7] Dunis C.L, Nathani A. *Quantitative trading of gold and silver using nonlinear models* Neural Network World: International Journal on Neural and Mass – Parallel Computing and Information Systems, 2007.

[8] Gunasekarage A., Power D.M. *The profitability of moving average trading rules in South Asian stock markets*. Emerging Markets Review, Volume 2, Issue 1, Pages 17–33, 2001.

[9] Hastie T., Tibshirani R., Friedman J. H. *The Elements of Statistical Learning*. Springer, 2001.

[10] Hastie T., Tibshirani R., James G., Witten D. *An Introduction to Statistical Learning: With Applications in R*. Springer, 2013.

[11] Shen S., Jiang H., Zhang T. *Stock Market Forecasting Using Machine Learning Algorithms*. Department of Electrical Engineering, Stanford University, Stanford, CA, 1–5, 2012.

[12] Juels A., Wattenbergy M., *Stochastic Hillclimbing as a Baseline Method for Evaluating Genetic Algorithms*. Advances in Neural Information Processing Systems 8, 1995.

[13] Dahlquist J.R., Kirkpatrick C.D. *Technical Analysis: The Complete Resource for Financial Market Technicians*. FT Press, 2011.

[14] Patel J., Shah S., Thakkar P., Kotecha K. *Predicting stock and stock price index movement using Trend Deterministic Data Preparation and machine learning techniques*. Expert Systems with Applications Volume 42, Issue 1, Pages 259–268, 2015.

[15] Mullen K. *et al*. *Package 'DEoptim' – Global Optimization by Differential Evolution*. CRAN R Project, 2016.

[16] Pardo R. *The Evaluation and Optimization of Trading Strategies*. Wiley Trading, 2011.

[17] Radford M.N. *Probabilistic Inference Using Markov Chain Monte Carlo Methods*. Technical Report CRG-TR-93-1, Department of Computer Science University of Toronto, 1993. s Ritter G. *Machine Learning for trading*. New York, 2017.

[18] Russell S.J., Nowig P. *Artificial Intelligence – A Modern Approach, Second Edition*. Pearson Education, Inc. 2003.

[19] Samuel A. *Some Studies in Machine Learning Using the Game of Checkers"*. IBM Journal of Research and Development 3(3): pages 210–229, 1959.

[20] Skiena S. S. *The Algorithm Design Manual, Second Edition*. Springer-Verlag London Limited, 2008.

[21] Smola A., Vishwanathan S.V.N. *Introduction to Machine Learning*. Cambridge University Press, 2008.

[22] Stanković J., Marković I., Stojanović M. *Investment Strategy Optimization Using Technical Analysis and Predictive Modeling in Emerging Markets*. Procedia Economics and Finance Volume 19, 2015. Pages 51–62.

[23] Storn, R.M, Price, K.V *Differential Evolution – A Simple and Efficient Heuristic for Global Optimization over Continuous Spaces*. Journal of Global Optimization, 1997. Pages 341–359.

[24] Storn, R.M., Price, K.V. Lampinen J.A. *Differential Evolution – A Practical Approach to Global Optimization*. Berlin Heidelberg: Springer-Verlag, 2006.

[25] Valiant L. *A theory of the learnable*. CACM, 1984.