Using Genetic Algorithms for Robust Optimization in Financial Applications

OLIVIER V. PICTET, MICHEL M. DACOROGNA, *
BASTIEN CHOPARD, MOULOUD OUSSAIDENE †
ROBERTO SCHIRRU ‡ AND MARCO TOMASSINI §

OVP.1995-02-06

April 27, 1995

Olsen & Associates
Research Institute for Applied Economics
Seefeldstrasse 233
8008 Zürich, Switzerland

Contents

*Olsen & Associates, Research Institute for Applied Economics,Seefeldstr. 233, CH-8008 Zürich, Switzerland

†Centre Universitaire d'Informatique, Université de Genève, CH-1211 Geneva 4, Switzerland

‡IDSIA, Corso Elvezia 36, CH-6900 Lugano, Switzerland

§CSCS, Manno and EPFL-DI-LSL, CH-11015 Lausanne, Switzerland

Abstract

In this study, optimal indicators and strategies for foreign exchange trading models are investigated in the framework of genetic algorithms. We first explain how the relevant quantities of our application can be encoded in "genes" so as to fit the requirements of the genetic evolutionary optimization technique.

In financial problems, sharp peaks of high fitness are usually not representative of a general solution but, rather, indicative of some accidental fluctuations. Such fluctuations may arise out of inherent noise in the time series or due to threshold effects in the trading model performance. Peaks in such a discontinuous, noisy and multimodal fitness space generally correspond to trading models which will not perform well in out-of-sample tests.

In this paper we show that standard genetic algorithms will be quickly attracted to one of the accidental peaks of the fitness space whereas genetic algorithm for multimodal functions employing clustering and a specially designed fitness sharing scheme will find optimal parameters which correspond to broad regions where the fitness function is higher on average. The optimization and the quality tests have been performed over eight years of high frequency data of the main foreign exchange rates.

1 Introduction

Trading models are algorithms proposing trading recommendations for financial assets. In our approach we limit this definition to a set of rules based on past financial data [1]. The financial data, which are typically series of prices, enter the trading model in the form of indicators, corresponding to various kinds of averages. Although progress has been made in understanding financial markets [2, 3], there is no definitive prescription on how to build a successful trading model and how to define the indicators. Automatic search and optimization techniques can be considered when addressing this problem.

However, optimizing trading models of financial assets without overfitting is a very difficult task because the scientific knowledge on financial markets is still very sparse. Overfitting means building the indicators to fit a set of past data so well that they are no longer of general value: instead of modeling the principles underlying the price movements, they model the specific price moves observed during a particular time period. Such a model usually exhibits a different behavior or may fail to perform for the test data that was not used in the optimization process. This difficulty is related to the fact that many financial time series do not show stability of their statistical behavior over time especially when they are analyzed intra-daily [3].

To minimize overfitting during optimization, a few important elements are to be present:

- a good measure of the trading model performance [1],
- indicator evaluation for different time series,
- large data samples,
- a robust optimization technique,
- and, last but not least, strict testing procedures.

Olsen & Associates has been continuously collecting tick-by-tick data on the foreign exchange (FX) market since 1986, thus providing large data samples for developing trading models. It has also produced a good trading model technology that has found applications in successful real-time trading models for the major FX rates [1].

The new element we want to present in this paper is a way to automize the search for improved trading models. Genetic algorithms offer a promising approach for addressing such problems [4]. Genetic algorithms consider a population of possible solutions to a given problem and evolve it according to mechanisms borrowed from natural genetic evolution: reproduction and selection. The criterion for selecting an individual is based on its fitness to the environment, or more precisely to the quality of the solution it bears. A possible solution is coded as a chromosome (gene), which is formally the data structure containing the values of the quantities characterizing the solutions.

In the framework of the present optimization, a gene will contain the indicators parameters, like a time horizon and a weighting function of the past, but also the type of operations used to combine them. The fitness function will be based on the return obtained when following the recommendations of a given trading model.

2 The main ingredients of simple trading models

In this section, we review the different ingredients that constitute the basis of a trading model and reformulate them in terms of simple quantities that can be used in conjunction with a genetic algorithm. Real trading models can be quite complicated and may require many different rules that also depend on the models own trading history. Here we limit ourselves to simple models that depend essentially on a set of indicators that are pure functions of the price history or of the current return. The purpose of this simplification is to make model coding and representation issues easier and to study indicator behavior.

The basic rule of a simple trading model with possible contrarian strategy is as follows,

```
IF |I| > K THEN G := sign(I) * sign(L - S) ELSE G := 0
```

where `I` is an indicator whose sign and value give the direction and strength of the current trend and `S` is an indicator that gives the strategy (trend following if `S < L` or contrarian if `S > L`). The constants `K` and `L` are break levels and `G`, called gearing, is the recommended position of the model such that long $= +1$, short $= -1$ and neutral $= 0$. [1].

In this study we investigate how to construct and select good indicators or combinations of indicators in order to make such simple trading models robust (i.e performing well on out-of-sample data). More complex models can be developed afterwards combining such simple trading models or using more complex rules in the evaluation of the strategy components.

2.1 Trading model indicators

Indicators are variables of the trading system algorithm whose values, together with the system rules, determine the trading decision process. In various papers [6, 7, 8, 1], we gave different descriptions of indicators that have been used in conjunction with trading models. Here, we focus on some abstract classification of these indicators in order to use them in a proper way in combination with the genetic algorithm. For the time being, the indicators we use are function of the time series itself. In a later stage, we can envisage using as indicators for a particular time series a function of other time series, like, for instance, interest rate functions for studying FX-rates.

First, we define two general classes of indicators. The symmetric I_s and the anti-symmetric I_a indicators:

$$I_s(X) \quad = \quad I_s(-X) \quad \text{and} \quad I_a(X) \quad = \quad -I_a(-X) \tag{2.1}$$

[1] For a good definition of the trading model terminology used here see [1] or [5]

where X is the basic variable used to define the indicator. This variable is generally a function the logarithm of price[2] x but can be also a function the current return of the model r_c. A typical antisymmetric indicator is a momentum [1] of the logarithm of price itself ($X = x$). A typical symmetric indicator would be the momentum of the absolute price change ($X = |x(t_i) - x(t_i - \Delta t)|$). The two classes will be used differently in the trading model. The antisymmetric indicators are the ones that provide the dealing signal while the symmetric indicators *modulate* it. For instance, they may forbid the model to trade or may modulate the threshold values. An indicator using the current return can be used for programming stop losses or stop profit, or to compute the risk of an open position.

Indicators are characterized by several parameters. We first describe the parameters that are common to both classes of indicators. Any indicator of the sort described here is composed of moving averages (MA) of different types so the first parameter is the *range* Δt_r of the moving average. The second is the weighting function of the past. As linear combinations of repeated applications of EMA's have been found to have remarkable properties[3][8], we use here a weighting function with two parameters defined as

$$MA_{X,j,n}(\Delta t_r, t) \equiv EMA_X^{(j,n)}(\Delta t_r, t) = \frac{1}{n+1-j} \sum_{i=j}^{n} EMA_X^{(i)}(\Delta t_r, t) \quad (2.2)$$

where we have two additional parameters: j and n, with $1 < j < n$.
The quantity $EMA_X^{(i)}$ is the ith application of the EMA operator, i.e.

$$EMA_X^{(i)} = EMA(EMA_X^{(i-1)}(\Delta t_r, t)) \quad (2.3)$$

where $EMA_X^{(1)}$ is computed with the formula

$$EMA_X(\Delta t_r, t) = \frac{1}{\Delta t_r} \int_{-\infty}^{t} X(t')\, e^{-\frac{t-t'}{\Delta t_r}}\, dt' \quad (2.4)$$

and $EMA_X^{(0)} \equiv X$. With this definition, $MA_{X,j,n}$ can model a wide variety of moving averages of the series $X(t)$.

In addition to a moving average, an indicator may be a momentum of various order. The simple momentum (of order 0) is defined as:

$$m_{X,j,n}(t) \equiv X(t) - MA_{X,j,n}(t) \ , \quad (2.5)$$

[2] We use here the same definition as in [9].

[3] Essentially the repeated application of EMA's cause the point of highest weight to be shifted progressively back in time and a linear combination of such EMA's can be used to generate weighting functions with plateaus and even approximate a rectangular moving average.

The concept of momentum can be extended to the *first* momentum which is a difference of two moving averages with different ranges, and the *second* momentum which is a linear combination of three moving averages with different ranges with the property that this combination is equal to zero for a straight line; it indicates the overall curvature of the series for a certain depth in the past. To avoid the introduction of too many additional parameters in the indicators, we restrict the possible variation of the *order* of the momentum parameter to only three possible values 0, 1 and 2.

Because of the construction of some input variable X, mainly used in the computation of symmetric indicators, the number of parameters of the problem may be extended. For instance, the volatility[3] has two parameters, the price change time interval and the sample period on which the volatility is computed.

In order to be able to combine different indicators and get similar results for different FX rate time series we need to normalize each indicator. To obtain this normalization we divide the value of the indicator by the square root of a long term moving average of the squared values of the indicator. Such a normalization is also very useful in order to make the indicators more adaptive to the market changes.

2.2 Operations on indicators

The operations on the indicators are essentially the four mathematical operations: +, -, *, /. In order to generate sensible trading models these operations must be performed following a set of rules:

- In section 2, we saw that the trading model takes a position according to an antisymmetric indicator I and a symmetric or antisymmetric strategy indicator S. To limit the space and prevent the generation of very complex functions, the operations * and / are restricted to operation between a symmetric and an antisymmetric indicator and the operations + and - are restricted to operation between indicators of the same type.
- For all the operations, the problem of normalization is present. In the case of multiplications or divisions, the scaling is not necessary because we already have constructed the indicators to be of order one. In the case of additions and subtractions, the resulting indicator need to be renormalized.
- The division should only be used with a modified value of the indicator in order to avoid division by zero. One would never divide by I but by $I +$

$sign(I) * \varepsilon$ where ε is a small positive constant. This constant can be chosen to be always the same if the indicators are well normalized, for instance 0.0001.

- We shall assume that the *number of operations* allowed is limited. We limit this number to three at first.

We have deliberately left out a set of possible operations like square roots, power functions or any *log* or *exp* functions. This choice is for the sake of simplicity and also because we think that the present set is already wide enough to produce interesting results. These additional operations can be included in a later stage.

2.3 Risk-sensitive performance measure

We wrote in the introduction that optimizing trading models is difficult because of the noise present in the data and the risk of overfitting. The challenge is to find indicators that are robust in the sense of being smooth and giving consistent results out-of-sample.

The first step is to define a value describing trading model performance in order to minimize the overfitting in the in-sample period and allowing us to compare different trading models against each other. The profit made by the model could be this fitness function but such a measure does not take into account the risk involved by the model. The risk being a major concern of the investors, it is necessary to add a risk component to the fitness function. We use here a risk-sensitive performance measure of the trading model which was developed for the optimization of FX real-time trading models [1]. This performance measure, called X_{eff}, is defined as

$$X_{eff} \;=\; \overline{R} - \frac{C}{2}\sigma^2 \tag{2.6}$$

where $\overline{R}$ is the annualized average total return, C is a risk aversion constant and σ^2 is the variance of the the total return curve against time, where a steady linear growth of the total return represents the zero variance case[4]. Because the variance $\sigma^2 \geq 0$, we have $X_{eff} \leq \overline{X}$. While the total return may mask a considerable risk introduced by high volatility of return, the effective return is risk-sensitive: the higher the volatility of return the lower the effective return. In other words, high effective returns indicate highly stable returns.

[4] For a more detailed definition of X_{eff} see [1]

In order to get more robust indicators, a good approach is to test each new indicator simultaneously on different exchange rate time series (since our experience has shown that trading models are robust if they work simultaneously on different rates without changing parameters). This increases the number of possible situations tested by the model in-sample. The performance measure of the model is then given by the average of the X_{eff} performance measure obtained for each time series corrected by the variance of the different X_{eff} values.

$$X'_{eff} = \overline{X}_{eff} - \frac{\sigma_{X_{eff}}}{3} \tag{2.7}$$

This correction decreases the probability to get indicators that have an overall good average but vary strongly from one time series to the other.

2.4 Trading model optimization

In order to optimize and test our trading models, we split the available historical data into three different periods. The first period is used to build-up the indicators, the second one is for the optimization of the trading model parameters and the third for selecting the best trading models. The build-up period generally contains more than ten years of daily data which is used to update long term indicators. The end of this period is 12 December 1986 for all the exchange rates. As many exchange rate time series do not show stability of their statistical behavior overtime, the rest of the historical data, from 1 January 1987 to 31 September 1994, is then divided into different alternating periods of in-sample and out-of-sample data. By this subdivision of the in-sample period we may test a larger variety of statistical behaviors and increase the probability of getting more robust and up-to-date parameters. The size of each sub in-sample period must be large enough to obtain statistically valid performance measures. Here we use in-sample bands with a size of one and a half year. The performance measure of the model is then given by the X_{eff} performance measure obtained over all in-sample periods.

The optimization process involves thousands of simulations runs through the in-sample data. To obtain results in a reasonable amount of time we extract and use only hourly data (equally spaced in the business time scale [10]). This choice of data sampling allows trading models to produce results which are very similar to the ones obtained with the full high frequency time series. To select the best parameter set we need an algorithm which explores the parameter space in an efficient way and chooses solutions that correspond not only to the largest average effective return but also lie on broad peaks or high flat regions of the effective return function. Such a requirement will ensure that small variations in the model parameters would keep the system state within a high performance region in term of X_{eff}.

Genetic algorithms (GA) [11, 12] have been shown to be useful in the optimization of multimodal functions in highly complex landscapes, especially when the function does not have an analytical description and is noisy or discontinuous. The usefulness of the genetic algorithm for such problems comes from its biological-like adaptive capabilities. When given a measure of the fitness (performance) of a particular solution and a population of adequately coded feasible solutions, the GA is able to search several regions of the search space simultaneously. In the GA better than average solutions are sampled more frequently and thus, through the genetic operators of crossover and mutation new promising solutions are generated and the average fitness of the whole population improves over time. Although GAs are not guaranteed to find the global optimum, they tend to converge towards good regions of high fitness. This is all we need, since global optima are of little use in our application. In the next section we will first describe a "naive" genetic algorithm approach to our problem, pointing out its drawbacks and how they can be circumvented.

3 The genetic algorithm to find and optimize simple trading models

For our first attempt, we used a classical genetic algorithm with the following features:

- Each chromosome is represented as an array of real numbers. In such representation each element of the array can be used to store parameters of different types (boolean, integer and real values). Except the boolean numbers which can take only the values 0 or 1, all the other parameters can be selected from a given list of possible values or varied into a given range.
- The population of a few tens of individuals is initialized at random, although seeding the initial population with preselected individuals is also possible. The selection of individuals for reproduction is fitness-proportionate, sometimes called roulette wheel selection [12]. We use here a two point crossover and mutation. For the mutation, we pick randomly a value inside the allowed range (list). In the crossover, the corresponding elements of the two chromosomes to be modified are just exchanged or can also be computed by linear

interpolation. The probabilities of crossover and mutation are respectively 0.9 and 0.08.

- We use generational replacement with elitism and no duplicates. At each new generation the major part of the population of parents is replaced by their offsprings. Only a limited number of the best individuals (the elite) are kept unchanged. The elite rate is generally of the order of 5% of the population size. We also eliminate all duplicate individuals to maintain population diversity and to avoid useless and time consuming fitness evaluations.

The fitness evaluation is done by the trading model evaluation program. This is a lengthy process since each chromosome (trading model) runs through a long time series of prices. In fact, this phase accounts for most of the total computing time. Because the different parameters are directly stored in the chromosome, the evaluation program is able to translate straightforwardly the content of the chromosome into the trading model to test. If a given chromosome corresponds to an invalid trading model the evaluation program returns the minimum possible fitness. Such a chromosome is eliminated in the construction of the next generation. The number of invalid individuals is usually very small.

To speedup this process, all genes of a given generation are evaluated in parallel over several machines of a workstation network. This has been done so far with a special-purpose network job queuing system. In future versions, we plan to use the more portable PVM (Parallel Virtual Machine) system, which will allow us to take advantage of several different platforms in a transparent way. In this distributed computing approach special attention must be paid to fault-recovery and checkpointing issues.

When running the above genetic algorithm, the performance of the selected models on the data set used in the learning process (in-sample) was excellent. The behavior of these models on the test data set (out-of-sample) was, however, not satisfactory. This is a very common phenomenon known as overfitting that plagues most real-world data-driven processes. As explained in the introduction, there are many techniques to try to avoid overfitting. In this case, it seems that the GA described above is itself partly responsible for the poor generalization capabilities. In fact, the population invariably converges after a while on one high-fitness but often unstable peak. In the next section we present modifications to the standard genetic algorithms that allow simultaneous searching for different high-fitness solutions. It will show that a judicious selection among the "best" solutions helps reducing the overfitting problem.

3.1 Genetic algorithms with sharing scheme for multi-modal functions

In the context of genetic algorithms, optimizing multimodal functions has been investigated using methods inspired from the natural notions of niche and species[13, 14, 15]. The general goal is to be able to create and maintain several subpopulations, ideally one per major peak of the fitness function, instead of having the whole population converge to one global optimum.

One of the best methods was proposed by Goldberg and Richardson[13]. The idea is that the GA perception of the fitness function is changed in such a way that when individuals tend to concentrate around a high peak, the fitness there is reduced by a factor proportional to the number of individuals in the region. This has the effect of diminishing the attractiveness of the peak and allowing parts of the population to concentrate on other regions. This effective fitness of an individual i, called shared fitness s_f is given by:

$$s_f(i) = \frac{f(i)}{m(i)} \tag{3.1}$$

where $f(i)$ is the original fitness and $m(i)$ is called the niche count. For an individual i, the quantity $m(i)$ is calculated by summing the sharing function values sh contributed by all N individuals of the population:

$$m(i) = \sum_{j=1}^{N} \mathrm{sh}(d_{ij}) \tag{3.2}$$

where d_{ij} is the distance between two individuals i and j and

$$\mathrm{sh}(d_{ij}) = \begin{cases} 1 - \left(\frac{d_{ij}}{\sigma_s}\right)^\alpha & \text{if } d_{ij} < \sigma_s \\ 0 & \text{otherwise} \end{cases} \tag{3.3}$$

The quantities α and σ_s are constants.

A difficulty of this method is to choose an adequate value of σ_s, for this requires prior knowledge about the number of peaks in the solution space. In our economical application as well as in many realistic problems, this information is not readily available.

A new method is proposed in [15] based on a different sharing scheme and using an adaptive cluster methodology. The authors show that this method is effective to reveal unknown multimodal function structures and is able to maintain subpopulation diversity. This method establishes analogies between clusters and niches in the following way: the GA population is divided, by the adaptive MacQueen's KMEAN clustering algorithm, in K clusters of individuals that correspond to K niches. The shared fitness calculation is the same as in the classical sharing method, but the niche count $m(i)$ is no longer associated with σ_s. In this case, the number of individuals of the cluster to which the individual i belongs plays a central role in the niche count calculation. As the number of clusters is associated with the number of niches (peaks), the individuals are put into a single partition of K clusters, where K is not fixed a priori, but determined by the algorithm itself. Therefore no a priori knowledge about the numbers of peaks of the fitness function is required as in the classical sharing method. The niche count $m(i)$ is computed as

$$m(i) \ = \ N_c \ - \ N_c * \left(\frac{d_{ic}}{2 \ D_{max}}\right)^{\alpha} \qquad\qquad x_i \ \in \ C_c \tag{3.4}$$

where N_c is the number of individuals in the cluster c, α is a constant, d_{ic} is the distance between the individual i and the centroid of its niche. The algorithm requires a distance metric in order to compute the distance between two clusters and the distance between one individual and one cluster. Two clusters are merged if the distance between their centroids is smaller than a threshold parameter $D_{\rm min}$. Moreover, when an individual is further away than a maximum distance $D_{\rm max}$ from all existing cluster centroids, a new cluster is formed with this individual as a member. The efficiency of the algorithm is improved by sorting the population in descending order according to the individual's fitness before the application of the clustering.

Such standard genetic algorithm with sharing and clustering has been applied to standard multimodal and continuous fitness functions [15, 16] with good results. One example of a more complex application was the determination of the optimum parameters of the *business* time scale[5] [10] which is used for analyzing price history and computing indicators. In this example, the optimization is quite difficult because we have to optimize simultaneously 17 parameters and the function to optimize is non-linear in some of the parameters. To solve such problem it was necessary to add a normalization of the parameter space in the genetic algorithm, i.e. each parameter is only allowed to vary in the range [0,1]. In simple problems the two clustering parameters are generally set to $D_{\rm min} = 0.05$ and $D_{\rm max} = 0.15$. But here, because of the large dimensionality of the parameter space, the value of the clustering parameters $D_{\rm min}$ and $D_{\rm max}$ must be much larger. In this case, the two parameters are multiplied by $\sqrt{n}$ where n in the number of parameters to be optimized. The results obtained with this genetic algorithm are very good

[5] This is a time scale that contracts and expands time based on seasonal activity or volatility of the time series.

and the sharing and clustering approach clearly increased the speed of convergence compared to the simple genetic algorithm described in the previous section.

When applied to the indicator optimization problem the genetic algorithm with sharing and clustering runs into difficulties. If the fitness landscape contains too many sharp peaks of high fitness all the selected clusters concentrate around these peaks and the genetic algorithm is unable to find stable solutions. In the next section, we propose some modifications to the genetic algorithm to detect clusters in the parameter space which correspond to more general and stable solutions.

3.2 Modified sharing function for robust optimizations

We need to find a new genetic algorithm which avoids the concentration of many individuals around sharp peaks of high fitness but detect broad regions of the parameter space which contain a group of individuals with a high average fitness level and a small variance of the individual fitness values.

To solve this problem we propose a new sharing function that penalizes clusters with large variance of the individual fitness values and also penalizes clusters with too many solutions concentrated inside too small a region. The distance metric considered here is the Euclidean distance computed in the real parameter space (phenotypic sharing). In the proposed sharing scheme all the individuals that belong to a given cluster c will share the same fitness value, i.e.

$$s_f(i) \; = \; \overline{f_c} \; - \; \left(\frac{N_c}{N_{av}} + \frac{1 - r_d}{r_d} \right) \; \sigma(f_c) \qquad\qquad \forall x_i \; \in \; C_c \tag{3.5}$$

where N_c is the number of genes in the cluster c and where the average fitness value $\overline{f_c}$ and the standard deviation of the individual fitness values $\sigma(f_c)$ are defined as usual by

$$\overline{f_c} \; = \; \frac{1}{N_c} \sum_{i=1}^{N_c} f(i) \qquad \text{and} \qquad \sigma(f_c) \; = \; \frac{1}{N_c - 1} \sqrt{\sum_{i=1}^{N_c} (f(i) \; - \; \overline{f_c})^2} \tag{3.6}$$

The term $\frac{N_c}{N_{av}}$ is used to control the number of genes inside each cluster. If N_c is smaller than the expected average number of genes inside each cluster N_{av} the correction is reduced otherwise it is increased. The second term $\frac{1-r_d}{r_d}$ is used to penalize clusters with too high concentration of genes around their centroid. The value r_d is defined as

$$r_d = \sqrt{\frac{1}{N_c} \sum_{i=1}^{N_c} \left(\frac{d_{ic}}{D_{max}} \right)} \tag{3.7}$$

where d_{ic} is the distance of the gene i to the centroid of the corresponding cluster c. Here the square root is used to avoid too large a correction for an average concentration of genes as is often the case.

As the method is based on the distribution of gene fitness inside each cluster, we keep only the clusters that contain at least a minimum number of members. We use here a minimum cluster size of two individuals. To keep the cluster's space as large as possible, we also have to minimize the overlap between different clusters. To reduce this overlap, the clustering parameter D_{min} must be quite large and here we use $D_{\text{min}} = D_{\text{max}}$. In order to have reasonable clustering parameter for large dimensionality of the parameter space, the value of the two clustering parameters D_{min} and D_{max} are multiplied by $\sqrt{n}$ where n in the number of parameters to be optimized.

With this new sharing scheme, the selection pressure is no more specific to each individual, as in a standard GA, but is the same for all genes present in a given cluster. This allows to get a selection mechanism which tries to find subpopulations of solutions with an average high quality instead of the best individual solution. Of course, the overall convergence speed is a little reduced.

The selection pressure towards good solutions is still present through the adaptive cluster methodology which tends to create clusters around a group of good individuals and through the reproduction technique which uses elitism and mating restriction inside each cluster. Moreover, to keep a larger variety in the population, all the individuals which do not really belong to any clusters (i.e. which are further away than the maximum distance D_{max} from all existing cluster centroids) will have an unmodified fitness value. During the reproduction phase these individuals will have no mating restriction and generally a slightly higher selection probability.

To speed up the full process, the result of each different gene is stored and not recomputed when this gene appears again in the next generations. Moreover, the information of all the previously computed solutions can be used at the end to assess the reasonableness of the optimum solution.

At the end, the algorithm selects for each cluster the best solution which is not further away than the distance $D_{\text{max}}/2$ from the cluster centroid. The final solution is the solution selected for the cluster that has the maximum average fitness corrected by the variance, i.e. for the maximum value of $\overline{f_c} - \sigma(f_c)$.

The success of this type of genetic algorithm is still quite sensitive to the quality of the fitness measure but also to the normalization of the parameter space, i.e.

to the quality of the metric used in the cluster construction. If the parameters do not have all the same sensitivity this should be also reflected in the clustering algorithm. That is why, we introduce the possibility of modifying the normalization of the parameter space but in many applications this is not enough and we need to use some parameter mapping functions which depend on the specific problem to solve.

4 Performance analysis

To test the new genetic algorithm we have first restricted the search to simple known indicators that were developed in a previous study [1] and for which we know what are the optimum solutions. These solutions have been found through an exhaustive in-sample search and human judgement.

For this first test, we optimize simple trading models using for I the antisymmetric indicator constructed as a simple momentum of x (the logarithm of price) and no strategy component S. In this case the genes contain a set of four numbers

- the break level K,
- the minimum order of EMA operator j,
- the maximum order of EMA operator n,
- the range of the moving average Δt_r.

For the optimization, 20 generations of 200 individuals were generated. Each gene is evaluated simultaneously on seven exchange rates, i.e. USD/DEM, USD/CHF, USD/JPY, GBP/USD, USD/FRF, USD/ITL and USD/NLG. Although the optimization is performed on a very different in-sample period than in the previous study, one of the optimum solution obtained with the GA is very similar to the original solution and all the important parameter values are the same, i.e. the range of the moving average $\Delta t_r = 16$ days and the minimum and the maximum order of EMA operator are $j = 2$ and $n = 4$. Moreover the genetic algorithm also found one new solution which correspond to a moving average range of 10 days and seems to be a little more stable. One other optimum with the same moving average range of 16 days is also found but for different values of the order of EMA operator. The three best solutions and their in- and out-of-sample quality are given in the table 1.

In a second test, we optimize a little more complex model with the same directional indicator I but now with a strategy component S based on a volatility indicator[17]. The genes are extended with the following five parameters corresponding to the volatility variable and to the S component:

- the range of the volatility sampling Δt_s,
- the overbought/oversold level L,
- the minimum order of EMA operator j',
- the maximum order of EMA operator n',
- the range of the moving average $\Delta t'_r$.

The optimum solution obtained by the genetic algorithm is shown in Table 2. The optimum solution is quite stable but also less successful than the simpler solution found in the first test. This is probably partly due to the inadequate choice of the strategy component but also due to the simplistic manner of including a contrarian strategy. This illustrate the difficulty of describing both adequately and simply the trading model algorithm.

We start to optimize more complex indicators for the direction component I based on the different possible combinations described in section 2.2. For such complex optimization, the population size and the number of generations must be significantly increased. From the current results it seems that the current X_{eff} performance measure is not sufficiently robust and we first need to develop a better one. In addition, the optimization is quite sensitive to the choice of the parameter space normalization.

However, in all these first tests, the genetic algorithm found the solution very efficiently. It found the optimum solutions after evaluating only a few percent of the full parameter space. This is extremely important because each evaluation of a complex model on a full time series can take up to few minutes of CPU time on our computers.

5 Concluding remarks

The use of genetic algorithm with robust sharing scheme seems appropriate when addressing large and complex optimization problems such as indicator search and trading model optimization. But the success of such methods also depend on a good fitness measure, an adequate representation of the model and a strict separation of the different data sets.

The problem of the normalization of the parameter space used for the clustering is important and must be specifically studied in the case of each different application.

Our first tests indicate a promising avenue but we need to improve on the fitness function to better take into account drawdown periods and on the representation of the trading model algorithm.

Acknowledgements

The authors acknowledge useful discussions with Rakhal D. Davé and Ulrich M. Müller. The authors would like to acknowledge the Swiss National Science Foundation for its financial support.

References

[1] Olivier V. Pictet, Michel M. Dacorogna, Ulrich A. Müller, Richard B. Olsen, and J. Robert Ward. Real-time trading models for foreign exchange rates. *Neural Network World*, 2(6):713–744, 1992.

[2] U. A. Müller, M. M. Dacorogna, R. D. Davé, O. V. Pictet, R. B. Olsen, and J. R. Ward. *FRACTALS AND INTRINSIC TIME – A CHALLENGE TO ECONOMETRICIANS*. Invited presentation at the XXXIXth International AEA Conference on Real Time Econometrics, 14-15 Oct 1993 in Luxembourg, and the 4th International PASE Workshop, 22-26 Nov 1993 in Ascona (Switzerland); also in "Erfolgreiche Zinsprognose", ed. by B. Lüthje, Verband öffentlicher Banken, Bonn 1994, ISBN 3-927466-20-4; UAM.1993-08-16, Olsen & Associates, Seefeldstrasse 233, 8008 Zürich, Switzerland, August 16 1993.

[3] Dominique M. Guillaume, Michel M. Dacorogna, Rakhal R. Davé, Ulrich A. Müller, Richard B. Olsen, and Olivier V. Pictet. *FROM THE BIRD'S EYE TO THE MICROSCOPE: A survey of new stylized facts of the intra-daily foreign exchange markets*. Internal document DMG.1994-04-06, Olsen & Associates, Seefeldstrasse 233, 8008 Zürich, Switzerland, April 06 1994.

[4] Franklin Allen and Risto Karjalainen. Using genetic algorithms to find technical trading rules. *Working Paper. The Rodney L. White Center for Financial Research, The Wharton School, University of Pensylvania*, 20-93:1–17, 1993.

[5] J. Robert Ward. *Trading Model Users' Guide.* Internal document JRW.1992-03-27, Olsen & Associates, Seefeldstrasse 233, 8008 Zürich, Switzerland, 1992.

[6] Ulrich A. Müller. *INDICATORS FOR TRADING SYSTEMS.* Internal document UAM.1989-08-10, Olsen & Associates, Seefeldstrasse 233, 8008 Zürich, Switzerland, August 10 1989.

[7] Ulrich A. Müller. *DIRECTION CHANGE INDICATORS Design Document and Discussion.* Internal document UAM.1991-02-01, Olsen & Associates, Seefeldstrasse 233, 8008 Zürich, Switzerland, February 01 1991.

[8] Ulrich A. Müller. *SPECIALLY WEIGHTED MOVING AVERAGES WITH REPEATED APPLICATION OF THE EMA OPERATOR.* Internal document UAM.1991-10-14, Olsen & Associates, Seefeldstrasse 233, 8008 Zürich, Switzerland, October 14 1991.

[9] U. A. Müller, M. M. Dacorogna, R. B. Olsen, O. V. Pictet, M. Schwarz, and C. Morgenegg. Statistical study of foreign exchange rates, empirical evidence of a price change scaling law, and intraday analysis. *Journal of Banking and Finance*, 14:1189–1208, 12 1990.

[10] Michel M. Dacorogna, Ulrich A. Müller, Robert J. Nagler, Richard B. Olsen, and Olivier V. Pictet. A geographical model for the daily and weekly seasonal volatility in the FX market. *Journal of International Money and Finance*, 12(4):413–438, 1993.

[11] David E. Goldberg. *Genetic Algorithms in Search, Optimization & Machine Learning.* Addison-Wesley, Reading, Massachusetts, 1989.

[12] Lawrence Davis. *Handbook of Genetic Algorithms.* Van Nostrand, New York, 1991.

[13] D. E. Goldberg and J. Richardson. Genetic algorithms with sharing for multimodal function optimization. *Proceeding of the Second International Conference on Genetic Algorithms*, pages 41–49, 1987.

[14] K. Deb and D. E. Goldberg. An investigation of niche and species formation in genetic function optimization. *Proceeding of the Third International Conference on Genetic Algorithms*, pages 42–50, 1989.

[15] Xiaodong Yin and Noël Germay. A fast genetic algorithm with sharing scheme using cluster analysis methods in multimodal function optimization. *Proc. Inter. Conf. Artificial Neural Nets and Genetic Algorithms, Innsbruck, Austria*, pages 450–457, 1993.

[16] B. Chopard, M. Oussaidène, O. Pictet, R. Schirru, and M. Tomassini. Evolutionary algorithms for multimodal optimization in financial applications. *In Proceedings of the SPP-IF Seminar, Zürich*, pages 139–142, 1995.

[17] Ulrich A. Müller. *VOLATILITY — A Classification of Its Definitions.* Internal document UAM.1989-08-24, Olsen & Associates, Seefeldstrasse 233, 8008 Zürich, Switzerland, August 24 1989.

parameters				in-sample		out-of-sample	
K	j	n	Δt_r	$\overline{R}$	X'_{eff}	$\overline{R}$	X'_{eff}
0.32	2	7	10	5.91	1.60	6.78	0.11
0.33	3	3	16	6.24	2.07	5.28	-1.40
0.28	2	4	16	6.25	1.77	5.28	-2.24

Table 1: Best results found by the GA in the first test. The first 4 columns give the resulting parameters and the 4 last ones the average yearly return $\overline{R}$ and the X'_{eff} quality obtained in the corresponding in- and out-of-sample periods respectively.

I parameters				S parameter					in-sample		out-of-sample	
K	j	n	Δt_r	Δt_s	L	j'	n'	$\Delta t'_r$	$\overline{R}$	X'_{eff}	$\overline{R}$	X'_{eff}
0.35	2	8	10	10	31.0	1	7	14	4.35	1.10	4.12	-1.71
0.32	2	4	16	1	2.0	1	3	1	4.29	0.12	3.85	-3.12

Table 2: Best results found by the GA in the second test. The first 9 columns give the resulting parameters and the two last ones the average yearly return $\overline{R}$ and the X'_{eff} quality obtained in the corresponding in- and out-of-sample periods respectively.