Genetic Algorithms with collective sharing for Robust Optimization in Financial Applications

Olivier V. Pictet, Michel M. Dacorogna, Rakhal D. Davé [*], Bastien Chopard [†], Roberto Schirru [‡] and Marco Tomassini [§]

OVP.1995-02-06

January 22, 1996

[*] 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

Contents

Abstract

In this study, optimal indicators and strategies for foreign exchange trading models are investigated using 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, rather they indicate 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 algorithms 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 on 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 (Pictet et al., 1992). 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 (Müller et al., 1993; Guillaume et al., 1994), 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 for financial assets without overfitting is a very difficult task because the scientific understanding of financial markets is still very limited. 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 movements observed during a particular time period. Such a model usually exhibits a different behavior (or may fail to trade successfully at all) when tested out-of-sample. 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 (Guillaume et al., 1994).

To minimize overfitting during optimization, the optimization process must include the following important ingredients:

- a good measure of the trading model performance (Pictet et al., 1992),
- indicator evaluation for different time series,
- large data samples,
- a robust optimization technique,
- last but not least, strict testing procedures.

Olsen & Associates has been continuously collecting tick-by-tick data on the foreign exchange (FX) markets 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 (Pictet et al., 1992).

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 (Allen and Karjalainen, 1993). 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 gene, which is formally the data structure containing the values of the quantities characterizing the solutions.

In the framework of the present application, a gene will contain the indicator parameters, for example time horizons and a weighting function for the past, and 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 so we can study indicator behavior.

The basic rule of a simple trend-following trading model is

```
IF |I| > K THEN G := sign(I) ELSE G := 0
```

where `I` is an indicator whose sign and value give the direction and strength of the current trend. The constant `K` is a break level and `G`, called the 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 so that they perform well out-of-sample).

More complex models can be developed afterwards combining such simple trading models or using more complex rules. For instance to introduce a simple contrarian strategy the previous rule can be modified as

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

where `S` is an indicator that gives the strategy (trend following if `S < L` or contrarian if `S > L`), and the constant `L` is the overbought/oversold break level.

[1] For a good definition of the trading model terminology used here see (Pictet et al., 1992) or (Ward, 1992)

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 (Müller, 1989a; Müller, 1991a; Müller, 1991b; Pictet et al., 1992), 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 combine them sensibly 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 antisymmetric I_a indicators:

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

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 (Pictet et al., 1992) of the logarithm of price itself ($I_a(X) = X$ where $X = x$). A typical symmetric indicator would be a measure of the volatility. The simplest one is the absolute price change ($I_s(X) = |X|$ where $X = x(t_i) - x(t_i - \Delta t)$). 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 EMAs have many useful properties[3](Müller, 1991b), we use here a weighting function with two parameters defined as

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

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.

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

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

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

[2] We use here the same definition as in (Müller et al., 1990).

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

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:

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

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 in the problem may be extended. For instance, the volatility(Guillaume et al., 1994) 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 obtain 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 size of the function space and prevent the generation of very complex functions, the operations * and / are restricted to operate between a symmetric and an antisymmetric indicator and the operations + and - are restricted to operate between indicators of the same type.
- For all the operations the problem of normalization is present. In the case of multiplications or divisions scaling is not necessary because we have already constructed the indicators so that they are of order one. In the case of additions and subtractions the resulting indicator need to be renormalized.
- 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 large number of possible operations; square roots, power functions or any *log* or *exp* functions. This 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 at 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 assumed by the model. As risk is a major concern of investors it is necessary to add a risk component to the fitness function. We use a risk-sensitive performance measure of the trading model which was developed for the optimization of FX real-time trading models (Pictet et al., 1992). This performance measure, called X_{eff}, is defined as

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

where $\overline{R}$ is the annualized average total return, C ($C > 0$) 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 $X_{\mathit{eff}} \leq \overline{R}$. While the total return may mask considerable risk introduced by a high return volatility, 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.

A good approach to obtain more robust indicators 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.

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

This correction decreases the probability of obtaining indicators that have a good average overall but which vary strongly from one time series to another.

[4] For a more detailed definition of X_{eff} see (Pictet et al., 1992)

2.4 Trading model optimization

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 is 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 over time, the rest of the historical data, from 1 January 1987 to 30 June 1994, is 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 we increase the probability of getting more robust and up-to-date parameters. The size of each in-sample period must be large enough to obtain statistically valid performance measures. Here we use in-sample periods with a size of one and a half years. The performance measure of the model is then given by the X_{eff} performance measure obtained from all in-sample periods.

The optimization process involves thousands of simulation 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 (Dacorogna et al., 1993)). This choice of data sampling produces trading models with 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 keep the system in the high performance region in terms of X_{eff}.

Genetic algorithms (GA) (Goldberg, 1989; Davis, 1991) have been shown to be useful in the optimization of multimodal functions in highly complex landscapes, especially when the function does not have an analytic description and is noisy or discontinuous. The usefulness of genetic algorithms for such problems comes from their evolutionary, 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 many regions of the parameter space simultaneously. In the GA, better than average solutions are sampled more frequently and thus, through the genetic operations 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 may be 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 gene is represented as an array of real numbers. In such a representation, each element of the array can be used to store parameters of different types (boolean, integer and real values). Except the boolean types 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 in 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 (Davis, 1991). 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 genes to be modified are either simply exchanged or 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 gene (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 gene, the evaluation program is able to translate straightforwardly the content of the gene into the trading model to test. If a given gene corresponds to an invalid trading model the evaluation program returns the minimum possible fitness. Such a gene is eliminated in the construction of the next generation. The number of invalid individuals is usually very small.

To speed up 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(Goldberg and Richardson, 1987; Deb and Goldberg, 1989; Yin and Germay, 1993). 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(Goldberg and Richardson, 1987). 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 the 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 as this requires prior knowledge about the number of peaks in the solution space. In our economic applications as well as in many realistic problems, this information is not readily available.

A new method is proposed in (Yin and Germay, 1993) based on a different sharing scheme and using an adaptive cluster methodology. The authors show that this method is effective at revealing 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 MacQueen's adaptive KMEAN clustering algorithm, into 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 in 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 number 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 $D_{\min}$. Moreover, when an individual is further away than a maximum distance $D_{\max}$ 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 a standard genetic algorithm with sharing and clustering has been applied to standard multimodal and continuous fitness functions (Yin and Germay, 1993; Chopard et al., 1995) with good results. One example of a more complex application is the determination of the optimum parameters of the *business* time scale[5] (Dacorogna et al., 1993) 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 this problem it was necessary to normalize the parameter space for 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 $D_{\min} = 0.05$ and $D_{\max} = 0.15$. But here, because of the high dimensionality of the parameter space, the values of the clustering parameters $D_{\min}$ and $D_{\max}$ 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 and the sharing and clustering methods 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 robust 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 robust 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 detects 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 a 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.

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

where N_c is the number of genes in the cluster c and where the average fitness value $\overline{f_c}$ and

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

standard deviation of the individual fitness values $\sigma(f_c)$ are defined as usual by

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

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. As we also need to keep enough clusters of resonable size, we have to limit the size of the largest clusters. The term N_c/N_{av} of the equation 3.5 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. In this study the constant N_{av} is chosen as the population size divide by the (pre-configured) expected number of clusters to be kept.

The second term $(1 - r_d)/r_d$ of equation 3.5 is used to penalize clusters with a concentration of genes around their centroid that is too high. The value r_d is defined as

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

where d_{ic} is the distance of 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.

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 $D_{\min} = D_{\max}$. In order to have reasonable clustering parameters for a parameter space of high dimensionality, the values of the two clustering parameters $D_{\min}$ and $D_{\max}$ are multiplied by $\sqrt{n}$ where n is the number of parameters to be optimized.

With this new sharing scheme, the selection pressure is no longer specific to each individual, as in a standard GA, but is the same for all genes present in a given cluster. This gives us 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 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 future generations. Moreover, the information of all the previously computed solutions can be used at the end to assess the reasonableness of the optimum solution.

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

4 Performance analysis

To test the new genetic algorithm we first restricted the search to simple known indicators that were developed in a previous study (Pictet et al., 1992) and for which we know the optimum solutions. These solutions have been found through a combination of 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, 40 generations of 200 individuals were generated. Each gene is evaluated simultaneously on seven exchange rates: USD/DEM, USD/CHF, USD/JPY, GBP/USD, USD/FRF, USD/ITL and USD/NLG.

We started by using the simple genetic algorithm described in the introduction to chapter 3. In Table 1 two sets of results are given. The first one is obtained by using the total return as a fitness measure and the second one is obtained by using the effective return. We observe that the result obtained using the effective return is less overfitted, but in both cases the optimization converges to a sub-optimum. If we increase the number of runs or the number of tested individuals, we invariably find a more global optimum which can gives much worse out-of-sample results (see first result of Table 2).

parameters				fitness	in-sample		out-of-sample	
K	j	n	Δt_r	measure	$\overline{R}$	X_{eff}	$\overline{R}$	X_{eff}
0.04	1	1	40	Tot.Ret	7.28	1.56	2.72	-4.79
0.34	3	3	16	X'_{eff}	6.09	1.91	4.55	-0.90

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

In a second test, we used the genetic algorithm with a clustering and sharing scheme described in the section 3.1. The obtained results are given in Table 2. The algorithm is better able to explore the fitness landscape and to find other optima. Here again, if we increase the number of runs or the number of tested individuals, the algorithm will invariably find a more global optimum (first solution of Table 2).

parameters				fitness	in-sample		out-of-sample	
K	j	n	Δt_r	measure	$\overline{R}$	X_{eff}	$\overline{R}$	X_{eff}
0.02	1	1	40	Tot.Ret	7.71	2.15	2.52	-7.30
0.10	1	6	12	Tot.Ret	6.69	1.18	4.29	-2.45
0.04	3	3	20	Tot.Ret	6.57	1.05	3.95	-3.98
0.36	3	3	16	X'_{eff}	5.93	1.84	4.26	-1.25
0.34	3	9	8	X'_{eff}	5.78	1.50	3.80	-3.47
0.34	2	7	12	X'_{eff}	5.83	1.41	3.73	-4.53

Table 2: Best results found by the GA using clustering and original sharing scheme. The first 4 columns give the resulting parameters and the last 4 the average yearly return $\overline{R}$ and the X_{eff} quality obtained in the corresponding in and out-of-sample periods respectively.

In the third test, we used the genetic algorithm with the collective sharing described in section 3.2. The results are given in table 3. The solutions obtained using either total return or effective return fitness measures provide out-of-sample results which present less overfitting on average. Moreover, the effective return fitness measure allows us to find solutions which provide better out-of-sample results. Although these optimizations are performed on a very different in-sample period than in the previous study, one of the optimum solutions 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 orders of the EMA operator are $j = 2$ and $n = 4$.

parameters				fitness	in-sample		out-of-sample	
K	j	n	Δt_r	measure	$\overline{R}$	X_{eff}	$\overline{R}$	X_{eff}
0.06	2	3	16	Tot.Ret	6.59	1.03	4.76	-3.13
0.10	1	6	12	Tot.Ret	6.48	0.86	3.42	-3.26
0.04	3	3	20	Tot.Ret	5.89	0.77	3.86	-3.30
0.32	3	3	16	X'_{eff}	6.29	2.05	4.83	-0.59
0.28	2	4	16	X'_{eff}	6.25	1.77	4.54	-2.50
0.28	3	7	10	X'_{eff}	6.08	1.55	4.52	-2.36

Table 3: Best results found by the GA using clustering and robust sharing. The first 4 columns give the resulting parameters and the last 4 the average yearly return $\overline{R}$ and the X_{eff} quality obtained in the corresponding in and out-of-sample periods respectively.

Preliminary tests with more complex indicators for the direction component I (based on the different possible combinations described in section 2.2) have shown that the population size and the number of generations must be significantly increased. However, in this simple test the genetic algorithm found the solution very efficiently.

5 Concluding remarks

We have shown that genetic algorithms with collective sharing successfully find robust 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. However, the success of this type of genetic algorithm is still quite sensitive to the following two considerations.

Fitness measure: We have already seen from the results (Tables 1,2,3) how the choice of X_{eff} is much superior to the raw return R, as a performance measure. Our research continues in the direction of finding better performance measures (particularly with respect to drawdown periods) in order to make the method more robust. The above genetic algorithm methods cannot be used as a proxy for better performance measures.

Paramater space normalization: This is the distance metric used in the cluster construction. If the parameter space is not equally sensitive to all the parameters this should be also reflected in the clustering algorithm. For this reason we have introduced the possibility of modifying the distance metric of the parameter space. (In some applications even this may not be enough and parameter mapping functions which depend on the specific problem may become necessary).

Acknowledgements

The authors acknowledge the useful discussions with Ulrich M. Müller. The Swiss National Science Foundation is gratefully acknowledged for its financial support.

References

Allen F. and Karjalainen R., 1993, *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.

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

Dacorogna M. M., Müller U. A., Nagler R. J., Olsen R. B., and Pictet O. V., 1993, *A geographical model for the daily and weekly seasonal volatility in the FX market*, Journal of International Money and Finance, **12**(4), 413–438.

Davis L. ., 1991, *Handbook of Genetic Algorithms*, Van Nostrand, New York.

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

Goldberg D. E., 1989, *Genetic Algorithms in Search, Optimization & Machine Learning*, Addison-Wesley, Reading, Massachusetts.

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

Guillaume D. M., Dacorogna M. M., Davé R. D., Müller U. A., Olsen R. B., and Pictet O. V., 1994, *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.

Müller U. A., 1989a, *Indicators for trading systems*, Internal document UAM.1989-08-10, Olsen & Associates, Seefeldstrasse 233, 8008 Zürich, Switzerland.

Müller U. A., 1989b, *Volatility — A classification of its definitions*, Internal document UAM.1989-08-24, Olsen & Associates, Seefeldstrasse 233, 8008 Zürich, Switzerland.

Müller U. A., 1991a, *Direction change indicators design document and discussion*, Internal document UAM.1991-02-01, Olsen & Associates, Seefeldstrasse 233, 8008 Zürich, Switzerland.

Müller U. A., 1991b, *Specially weighted moving averages with repeated application of the EMA operator*, Internal document UAM.1991-10-14, Olsen & Associates, Seefeldstrasse 233, 8008 Zürich, Switzerland.

Müller U. A., Dacorogna M. M., Davé R. D., Pictet O. V., Olsen R. B., and Ward J. R., 1993, *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.

Müller U. A., Dacorogna M. M., Olsen R. B., Pictet O. V., Schwarz M., and Morgenegg C., 1990, *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.

Pictet O. V., Dacorogna M. M., Müller U. A., Olsen R. B., and Ward J. R., 1992, *Real-time trading models for foreign exchange rates*, Neural Network World, **2**(6), 713–744.

Ward J. R., 1992, *Trading model users' guide*, Internal document JRW.1992-03-27, Olsen & Associates, Seefeldstrasse 233, 8008 Zürich, Switzerland.

Yin X. and Germay N., 1993, *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, 450–457.