ECGA vs. BOA in Discovering Stock Market Trading Experts

Piotr Lipinski
Institute of Computer Science
University of Wroclaw
ul. Joliot-Curie 15
50-383 Wroclaw, Poland
lipinski@ii.uni.wroc.pl

ABSTRACT

This paper presents two evolutionary algorithms, ECGA and BOA, applied to constructing stock market trading expertise, which is built on the basis of a set of specific trading rules analysing financial time series of recent price quotations. A few modifications of ECGA are proposed in order to reduce the computing time and make the algorithm applicable for real-time trading. In experiments carried out on real data from the Paris Stock Exchange, the algorithms were compared in terms of the efficiency in solving the optimization problem, in terms of the financial relevance of the investment strategies discovered as well as in terms of the computing time.

Categories and Subject Descriptors

I.2 [**Artificial Intelligence**]: Applications and Expert Systems; I.2 [**Artificial Intelligence**]: Learning; J.1 [**Computer Applications**]: Administrative Data Processing—*Financial*

General Terms

Experimentation

Keywords

estimation of distribution algorithms, extended compact genetic algorithm, bayesian optimization algorithm, decision support systems, stock market expertise, financial time series

1. INTRODUCTION

Financial data analysis remains a challenge for human analysts, stock market investors and computer expert systems. Although recent computational technologies, such as artificial intelligence, neural networks or evolutionary algorithms, enable an intensive development of financial analysis methods facilitating and speeding up computations, the size of data and the computing time remain the major constraints for computational methods.

However, there are some research on applying computational intelligence to financial modelling [2]. Genetic programming was applied to building decision trees for supporting financial decision making [16]. Some evolutionary algorithm were constructed for portfolio optimization [12]. Grammatical evolution was applied to discovering trading rules for stock market speculations [4].

In this paper, we discuss an evolutionary approach to constructing stock market trading expertise built on the basis of a set of specific trading rules analysing financial time series of recent price quotations [8]. In order to solve the optimization problem, corresponding to discovering the trading expertise, genetic algorithms may be applied. It may be solved using a number of fast algorithms, such as CGA [7], PBIL [1] or SGA [5]. However, much better results may be often obtained with more advanced, but also more time-consuming, algorithms, such as BOA [14] or ECGA [6]. Unfortunatelly, the time constraint is critical for stock trading decision support systems, especially for real-time trading, so in practice a less efficient but faster algorithm must be chosen.

In this paper, we analyze two algorithms, ECGA and BOA, applied to discovering trading expertise. We compare and discuss their performance and usability in modelling financial expertise.

This paper is structured in the following manner: Section 2 introduces the stock market trading expertise. In Section 3, construction of such an expertise is transformed into an optimization problem. Section 4 and 5 describe two algorithms proposed to solve the problem. In Section 6, some experiments and their results are presented. Finally, Section 7 concludes the paper.

2. STOCK MARKET TRADING EXPERTS

2.1 Stock Market Trading Rules

Financial analysts and stock market investors observe the stock market to sell stocks if they tend to lose value, to buy stocks if they tend to gain value, and to do nothing in the remaining cases. Investors often assume that future stock market prices can be, more or less accurately, predicted on the basis of publicly available information, such as past prices, order books, economic reports, tax rates and so on. Therefore, they analyze available data to take trading decisions in line with this analysis.

Obviously, various methods of financial data analysis are used widespread. For instance, Technical Analysis [13] mainly focuses on past prices, considering stock market data as the main source of information. Other ones, for instance Fundamental or Quantitative Analysis [3], take into consideration various information on specific firms, such as assets, liabilities, expenses or revenues. They try to estimate the future performance of the firm which is then considered as the main factor driving the stock price.

Technical Analysis introduces many functions to characterize stock market data. It attempts to detect trends and discover signals of the occurrence of particular future events like falls and rises in stock prices. Using these functions, investors make trading decisions: buying, selling or doing nothing.

In order to formalize financial analysis methods, the concept of *a stock market trading rule* is introduced. Generally speaking, a stock market trading rule is a function which evaluates a trading signal, based on available information. This information is referred to as *a factual financial knowledge* and denoted as $\mathcal{K}$. In Technical Analysis, the factual financial knowledge $\mathcal{K}$ represents historical prices and order books; in other cases, $\mathcal{K}$ may include tax rates, exchange rates, economic reports or recent press dispatches. A stock market trading rule is formalized in Definition 1.

Definition 1: A stock market trading rule is a function

$$f : \mathcal{K} \mapsto y \in \mathbf{R} \tag{1}$$

which maps a factual financial knowledge $\mathcal{K}$ to a real number y. The result may be interpreted later as a trading signal: values lower than a certain threshold α_1 correspond to a sell signal, values greater than a certain threshold α_2 correspond to a buy signal, and remaining values correspond to no signal.

Formally, in order to get a trading signal, selling, buying or doing nothing, an auxiliary function $D(y; \alpha_1, \alpha_2)$ is introduced

$$D(y; \alpha_1, \alpha_2) = \begin{cases} -1 & \text{if } y \leq \alpha_1 \\ 0 & \text{if } \alpha_1 < y < \alpha_2 \\ 1 & \text{if } \alpha_2 \leq y \end{cases} \tag{2}$$

where $-1 \leq \alpha_1 < \alpha_2 \leq 1$ denote thresholds specified as parameters. Therefore, $D(y; \alpha_1, \alpha_2) = -1$ denotes a sell signal, $D(y; \alpha_1, \alpha_2) = 1$ denotes a buy signal, and $D(y; \alpha_1, \alpha_2) = 0$ denotes no signal.

As the parameters α_1, α_2 are usually constant, fixed for specific values depending on the considered experiments, they will be omitted, if it does not lead to misunderstanding. In this paper, $\alpha_1 = -0.5$ and $\alpha_2 = 0.5$ are used, although other values are possible. It is worth noticing that these values determine the number of signals generated by the trading rules. The closer are the values of α_1 and α_2, the more buy and sell signals are generated.

2.2 Example of a Trading Rule : the Stochastic Oscillator

Among the large number of trading rules (in experiments, 350 trading rules are considered), only one, namely the Stochastic Oscillator, will be discussed in details in this paper.

The Stochastic Oscillator indicator was introduced by George C. Lane [13]. It shows the location of the current close relative to the high and low range over a number of periods. Closing levels consistently near the top of the range indicate accumulation (buying pressure) and those near the bottom indicate distribution (selling pressure).

The indicator is defined as follows: Let

$$\kappa(t) = \frac{\mathrm{Close}(t) - \mathrm{M}_*(t; T_1)}{\mathrm{M}^*(t; T_1) - \mathrm{M}_*(t; T_1)}, \tag{3}$$

where $\mathrm{M}_*(t; T)$ and $\mathrm{M}^*(t; T)$ are the lowest low and the highest high respectively over the last T periods

$$\mathrm{M}_*(t; T) = \min\{\mathrm{Low}(\tilde{t}) : t - T < \tilde{t} \leq t\}, \tag{4}$$

$$\mathrm{M}^*(t; T) = \max\{\mathrm{High}(\tilde{t}) : t - T < \tilde{t} \leq t\}, \tag{5}$$

$\mathrm{Close}(t), \mathrm{Low}(t), \mathrm{High}(t)$, denote the close, high and low of the t-th time period.
Let $\delta(t)$ be the T_2-period moving average of κ

$$\delta(t) = \frac{1}{T_2} \sum_{\tilde{t}=t-T_2+1}^{t} \kappa(\tilde{t}). \tag{6}$$

T_1 and T_2 are two parameters of the Stochastic Oscillator indicator. Their values vary according to the type of signal desired. $T_1 = 14$ and $T_2 = 3$ are usual values, although other values are possible.

The Stochastic Oscillator may lead to diverse interpretations to generate buy and sell signals. In this paper, for the sake of simplicity, it is assumed that sell signals occur when δ is above a specified threshold θ_1 (e.g. $\theta_1 = 0.80$) and buy signals occur when δ is below a specified threshold θ_2 (e.g. $\theta_2 = 0.20$). However, the other approaches are also considered in some experiments, but they are not referred to as the original Stochastic Oscillator.

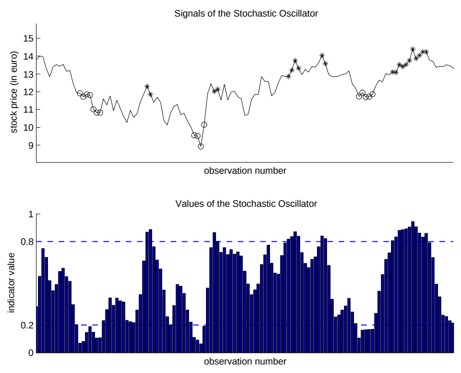

Figure 1: Stochastic Oscillator with parameters $T_1 = 14$ and $T_2 = 3$ for daily price quotations of AXA over the period from January 2, 2003 to June 30, 2003. Top: the signals generated. Circles denote buy signals and stars denote sell signals. Bottom: values of the indicator.

Figure 1 shows the Stochastic Oscillator with parameters $T_1 = 14$ and $T_2 = 3$ on a time series including daily prices of AXA over a period from January 2, 2003 to June 30, 2003. The bottom plot presents the indicator values. The top plot presents the corresponding prices and signals. One can see that results are quite efficient in this case.

2.3 Stock Market Trading Experts

Stock market trading rules produce trading advices for financial analysts and investors. Naturally, investors may smoothly accept these advices when all the trading rules point in the same direction, but they must wonder about a final trading decision when some trading rules are discordant. As, in practice, there are often opposing advices produced by trading rules, investors must spend some effort to transform these advices into a trading decision. One solution is to follow the majority of trading rules. An other solution is to define a set of favorite rules and consider only advices produced by this subset. A more complex solution is to use a weighted average of advices.

In a decision support system inspired from experience of financial analysts and investors, the same problem appears. In this paper, in order to solve it, we try to select an efficient set of trading rules which defines the final trading decision by the trading advice proposed by the majority of trading rules in the chosen subset, ignoring the other trading rules. Such a set will be referred to as *a stock market trading expert.* It is formalized in Definition 2.

Definition 2: Let $\mathcal{R} = \{f_1, f_2, \dots, f_d\}$ be the entire set of available trading rules. A trading expert is a subset of the entire set of trading rules

$$E \subset \mathcal{R}. \tag{7}$$

A result $E(\mathcal{K})$ of a trading expert $E = \{f_{i_1}, f_{i_2}, \dots, f_{i_k}\}$, where $i_1 < i_2 < \dots < i_k$, for a given factual financial knowledge $\mathcal{K}$, is an arithmetic average of the results of the trading rules from the subset E

$$E(\mathcal{K}) = \frac{1}{k}(f_{i_1}(\mathcal{K}) + f_{i_2}(\mathcal{K}) + \dots + f_{i_k}(\mathcal{K})). \tag{8}$$

Obviously, in order to get a trading signal, selling, buying or doing nothing, the result of the trading expert must be transformed using the auxiliary function defined in (2).

3. PROBLEM DEFINITION

3.1 Valuation of Trading Experts

Stock market trading experts, defined in the previous section, are designed not to produce a single advice at a given date, but a sequence of advices at successive dates. In practice, a given advice may be accidental, depending on the context, while a sequence of such advices may suit accurately the stock market state.

Consider a specific time period. Let $t_0, t_1, t_2, \dots, t_{T-1}$ denote its successive dates. Since $\mathcal{K}$ varies through time, $\mathcal{K}_{t_i}$ will denote the factual financial knowledge available at time t_i. For instance, if $\mathcal{K}$ represents financial time series, including prices for a specific stock, $\mathcal{K}_{t_i}$ may be a data table, whose rows consist of open, high, low and close price, transaction volume and stock market index collected since a given date and up to date t_i.

Each trading expert E proposes a sequence of trading decisions at successive dates in the time period considered

$$D(E(\mathcal{K}_{t_0})), D(E(\mathcal{K}_{t_1})), \dots, D(E(\mathcal{K}_{t_{T-1}})). \tag{9}$$

For a sequence of trading decision to be useful, it is necessary to define the volume of stocks to be traded. In practice, this volume depends on individual trading abilities and preferences, but in decision support systems a common approach must be introduced.

However, when a trading expert advises to buy, some investors can invest all their capital, others will only invest a part of it, and others will invest nothing, due to a lack of cash. An advice to sell may be more difficult to follow if investors do not hold the stock, short selling conditions being sometimes restrictive.

In this paper, volumes of stock to buy or sell are obtained by simulating the behavior of an hypothetical investor. This investor is given an initial endowment (c_0, s_0) with c_0 the amount of cash and s_0 the initial quantity of stocks (in experiments, $c_0 = 10000$ and $s_0 = 100$). Since trading generates transaction costs, they are assumed proportional with rate $\tau\%$ (in simulations, $\tau = 0.2$).

At time t_0, the investor takes a decision $D(E(\mathcal{K}_{t_0}))$. If the decision is to sell, i.e. $D(E(\mathcal{K}_{t_0})) = -1$, he sells $q\%$ of stocks, i.e. the amount of stock Δs in the investor's order is equal to

$$\Delta s = s_0 \cdot q/100. \tag{10}$$

If the decision is to buy, i.e. $D(E(\mathcal{K}_{t_0})) = 1$, he invests $q\%$ of money in stocks, i.e. the amount of stock Δs in the investor's order is equal to

$$\Delta s = \frac{c_0 \cdot q/100}{(1 + \tau/100) \cdot \mathrm{Open}(t_1)}, \tag{11}$$

where Open(t) denotes the opening price at date t. The parameter q in experiments equals 50, which guarantees that the investor will not empty his account too fast. The transaction is executed at time t_1 and the investor's capital changes accordingly.

Therefore, at time t_1, the investor's capital consists of the amount of money c_1 and the amount of stocks s_1

$$c_1 = c_0 - D(E(\mathcal{K}_{t_0})) \cdot \Delta s \cdot \mathrm{Open}(t_1) - \tau/100 \cdot \Delta s \cdot \mathrm{Open}(t_1), \tag{12}$$

$$s_1 = s_0 + D(E(\mathcal{K}_{t_0})) \cdot \Delta s. \tag{13}$$

At time t_1, the investor, makes a decision $D(E(\mathcal{K}_{t_1}))$, which is executed at time t_2 and the investor's capital again changes, and so on. Finally, $c_0, c_1, \dots, c_T$ denote the successive cash volumes and $s_0, s_1, \dots, s_T$ the corresponding quantities of stocks over the time period considered.

3.2 Optimization Problem

Such a simulation characterizes the behavior of a trading expert over a specific time period. These sequences of cash volumes and quantities of stocks might be used for valuation the trading expert, for instance by evaluating the obtained profit over the specific time period. The problem is that in practice the goal is to build a trading expert efficient over a future period with unknown stock prices, so the simulation cannot be performed over that period. However, the future performance of an expert may be assessed on the basis of its behavior over a past period using so-called performance measures introduced by financial analysts and market traders considering not only the future return rate, but also the risk related to achieving it. There are a number of different performance measure, such as the Sharpe ratio, the Sortino ratio or the Sterling ratio [10]. In this paper, we focus on the Sharpe ratio.

Let

$$\begin{aligned} C_i &= c_i + s_i \cdot \mathrm{Open}(t_i), \quad i = 0, 1, \dots, T && (14) \\ R_i &= \frac{C_i - C_{i-1}}{C_{i-1}}, \quad i = 1, 2, \dots, T && (15) \end{aligned}$$

denote the investor's date-t_i wealth and the return on the portfolio for the period $[t_{i-1}; t_i]$. The Sharpe ratio for a trading expert behavior over the period $[t_0, t_0+T)$ is defined then as

$$\varrho(E) = \frac{\mathbf{E}[R] - r_0}{\mathbf{Std}[R]}, \tag{16}$$

where $\mathbf{E}[R]$ denotes the expected return rate of the investment strategy proposed by the trading expert E over the period $[t_0, t_0+T)$, the $\mathbf{Std}[R]$ denotes the standard deviation of the return rate and r_0 denote the reference return rate of risk-free asset.

Using the performance measure defined, discovering efficient stock market trading experts may be transformed to an optimization problem with the objective function $\varrho(E)$ over the search space of all trading experts E. Formally, let $\mathcal{R} = \{f_1, f_2, \ldots, f_d\}$ be the set of all available trading rules and $\mathcal{E}(\mathcal{R})$ be the set of all available trading experts built on these rules. The objective is to find a trading expert $E \in \mathcal{E}(\mathcal{R})$ such as

$$\varrho(\tilde{E}) \leq \varrho(E) \tag{17}$$

for all $\tilde{E} \in \mathcal{E}(\mathcal{R})$, for a given training period $[t_0, t_0+T)$ and for given parameters of the simulation described in the previous section.

3.3 Representation of Trading Experts

Each trading expert $E \in \mathcal{E}$ given by (7) may be represented in a natural way by a binary vector $\mathbf{e} = (e_1, e_2, \ldots, e_d)$, such that

$$e_i = \begin{array}{ll} 0, & \text{if } f_i \notin E \\ 1, & \text{if } f_i \in E \end{array}, \tag{18}$$

i.e. e_i corresponds to the i-th trading rule f_i from $\mathcal{R}$; $e_i = 0$ denotes the absence and $e_i = 1$ denotes the presence of the trading rule f_i in the set E.

Then, a one-to-one map is defined between trading experts and binary vectors of length d. Consequently, in the optimization problem defined in the previous section, the search space is simply $\{0,1\}^d$ and the objective function is the performance $\varrho(\mathbf{e})$ of the trading expert E corresponding to the binary vector $\mathbf{e}$.

4. DISCOVERING TRADING EXPERTS USING EXTENDED COMPACT GENETIC ALGORITHM (ECGA)

In this section, we discuss solving the optimization problem using the Extended Compact Genetic Algorithm (ECGA) [6], which evolves a population of trading experts E encoded in binary chromosomes $\mathbf{e} \in \{0,1\}^d$, as defined in (18). The problem is considered in the context of a given set $\mathcal{R}$ of trading rules, a given performance measure ϱ being the Sharpe ratio, a given stock and a given training period.

In the ECGA, the main idea is to use linkage learning to build a chromosome partition, i.e. to divide the chromosome into a number of groups consisting of related genes, called *building blocks*. Such a structure over the chromosome is being discovered dynamically by the ECGA in the evolution process. It corresponds to discovering dependencies among stock market trading rules in trading experts, more exactly, to grouping related trading rules in blocks. Previous research proves that trading rules are strongly correlated and such relations are usually frequent [9].

Due to size constraints, only the overview of the ECGA is discussed here. A complete specification of the ECGA may be found in [6].

4.1 Algorithm

Figure 2 shows the framework of the ECGA designed to optimize an objective function ϱ with a population $\mathcal{P}$ composed of N trading experts.

```
EXTENDED-COMPACT-GENETIC-ALGORITHM($\varrho$, $N$)
1  $\mathcal{P} \leftarrow$ RANDOM-POPULATION($N$);
2  POPULATION-EVALUATION($\mathcal{P}, \varrho$);
3  while not TERMINATION-CONDITION($\mathcal{P}$)
4  do
5      WEAK-INDIVIDUAL-ELIMINATION($\mathcal{P}$);
6      $\mathcal{S} \leftarrow$ CHROMOSOME-PARTITION-GENERATING($\mathcal{P}$);
7      POPULATION-REGENERATING($\mathcal{P}, \mathcal{S}, N$);
8      POPULATION-EVALUATION($\mathcal{P}, \varrho$);
```

Figure 2: The Extended Compact Genetic Algorithm designed to optimize an objective function ϱ with a population $\mathcal{P}$ composed of N trading experts.

First, the algorithm creates an initial population $\mathcal{P} = \{\mathbf{e}_1, \mathbf{e}_2, \ldots, \mathbf{e}_N\}$ of random trading experts. After creation, the population is evaluated.

After the population is generated, weak individuals are eliminated and the chromosome partition $\mathcal{S}$ is generated on the basis of the population $\mathcal{P}$ using the algorithm shown in Figure 3 described further.

After the chromosome partition is generated, the population is generated anew on the basis of the chromosome partition $\mathcal{S}$, in such a way that offspring individuals inherit all the genes in one building block together [6], and the process is repeated, until a termination condition is satisfied.

4.2 Chromosome Partition Generating

Figure 3 shows the framework of the algorithm for generating the chromosome partition $\mathcal{S}$ on the basis of the population $\mathcal{P}$. It tries to discover an optimial chromosome partition according to *the combined complexity* measure evaluated on the basis of the current population [6].

In the classic ECGA, the chromosome partition generating is performed by a greedy algorithm. Such an algorithm is very time-consuming, which constitutes a serious disadvantage of the ECGA.

First, the algorithm creates an initial chromosome partition $\mathcal{S}$, where all the genes in the chromosome are considered independent, so the chromosome partition $\mathcal{S}$ consists of d building blocks of size 1.

In order to update the chromosome partition $\mathcal{S}$, the algorithm considers all the candidate chromosome partitions created from the current partition $\mathcal{S}$ by merging two building blocks. Assuming that the current partition consists of n building blocks, there are $\frac{n(n-1)}{2}$ candidate partitions to evaluate. The best of new candidate partitions, i.e. that one which minimizes the combined complexity among them, replaces the current partition $\mathcal{S}$, unless it is not better than $\mathcal{S}$ when the algorithm terminates. The whole process is repeated until none of new candidate partitions is better than $\mathcal{S}$, i.e. has lower combined complexity than $\mathcal{S}$.

It is worth noticing that, in each iteration, the size n of

```
CHROMOSOME-PARTITION-GENERATING($\mathcal{P}$)
 1  $\mathcal{S} \leftarrow \{\{1\}, \{2\}, \ldots, \{d\}\}$;
 2  repeat
 3          $\mathcal{S}_{min} = \emptyset$;
 4          $CC_{min} \leftarrow$ COMBINED-COMPLEXITY($\mathcal{S}, \mathcal{P}$);
 5          for each $B_i, B_j \in \mathcal{S}$
 6          do
 7              $\tilde{\mathcal{S}} \leftarrow \mathcal{S} \setminus \{B_i, B_j\} \cup \{B_i \cup B_j\}$;
 8              $\tilde{CC} \leftarrow$ COMBINED-COMPLEXITY($\tilde{\mathcal{S}}, \mathcal{P}$);
 9              if $\tilde{CC} < CC_{min}$
10                 then $\mathcal{S}_{min} \leftarrow \tilde{\mathcal{S}}$;
11                      $CC_{min} \leftarrow \tilde{CC}$;
12          if $\mathcal{S}_{min} \neq \emptyset$
13             then $\mathcal{S} \leftarrow \mathcal{S}_{min}$;
14    until $\mathcal{S}_{min} = \emptyset$;
15  return $\mathcal{S}$;
```

Figure 3: Generating the chromosome partition $\mathcal{S}$ on the basis of the population $\mathcal{P}$ in the ECGA.

the chromosome partition decreases by 1, which makes the partition restructuring rather slow.

4.3 Reducing Computing Time

Although applying ECGA to discovering trading experts leads to very good solutions, one of serious disadvantages is the computing time. The bottleneck is the greedy algorithm for updating the chromosome partition. Although a few efficient implementations were proposed [11], which reduce the computing time by managing and reusing previously calculated combined complexities for chromosome partitions, further modifications are necessary.

In order to illustrate how long may be the computing time, results of a few experiments are shown in Table 1. In each experiment, the algorithm was run to discover an optimal trading expert E based on a specific set $\mathcal{R}$ of 350 trading rules maximizing the Sharpe ratio ϱ over the training period from July 28, 2000 to January 16, 2001 (120 days) for financial time series including daily price quotations of the stock Renault. Similar results were obtained for other optimization problems.

Table 1: Computing times for ECGA run to discover an optimal trading expert based on a specific set of 350 trading rules with varying population size N

N	Avg(ϱ)	Max(ϱ)	Iterations	Time per Iter.
500	0.1178	0.1191	12	18 s
1000	0.1178	0.1201	36	50 s
2000	0.1269	0.1302	46	112 s
4000	0.1232	0.1317	35	229 s
8000	0.1335	0.1358	45	7 min
20000	0.1244	0.1342	30	18 min

In this paper, we propose some modifications in order to limit the greedy algorithm for updating the chromosome partition and make ECGA practical. Some of them are similar to improvements of the model building process in the BOA proposed in [15].

First, the chromosome partition updating should be separated from the main evolutionary algorithm. A simple solution is to update the chromosome partition only once for a few iterations of the main evolutionary algorithm. Other one is to update the chromosome partition only a specific number of times and then turn it off. In both cases, a significant computing time reduction were obtained. However, despite the improvement, the computing time remains too long.

Second, the chromosome partition updating should be limited in such a way that only a specific number of iterations within the chromosome partition updating is performed. Therefore, each time, only a limited number of building blocks may be merged, even if there are still building blocks, whose merging decreases the combined complexity. Computing time reduction obtained in such a way makes the algorithm practical.

In experiments, the computing time of the original ECGA with a population of 4000 individuals and 30 iterations was about 115 minutes, where one iteration took about 229 seconds in average. In each iteration, the main part was updating the chromosome partition, while the remaining part including the population generation took only a few seconds. After the modifications proposed, the overall computing time for the same parameters, i.e. a population of 4000 individuals and 30 iterations, decreased to 1 – 2 minutes.

5. DISCOVERING TRADING EXPERTS USING BAYESIAN OPTIMIZATION ALGORITHM (BOA)

In this section, we discuss solving the optimization problem using the Bayesian Optimization Algorithm (BOA) [14]. Although the general concept of the approach is similar to the previous one, the difference lies in modelling dependencies among trading rules.

In the BOA, the main idea is to use bayesian networks to represent dependencies among genes in the chromosome. Such a network is being discovered dynamically by the BOA in the evolution process. It corresponds to discovering dependencies among stock market trading rules in trading experts, more exactly, to tracking values of which trading rules influence values of others.

As in the case of the ECGA, modelling dependencies among genes is a time consuming task. Therefore, a few approaches to more economic model building were proposed [15].

Due to size constraints, only the overview of the BOA is discussed here. A complete specification of the BOA may be found in [14].

5.1 Algorithm

Figure 4 shows the framework of the BOA designed to optimize an objective function ϱ with a population $\mathcal{P}$ composed of N trading experts.

First, the algorithm creates an initial population $\mathcal{P} = \{\mathbf{e}_1, \mathbf{e}_2, \ldots, \mathbf{e}_N\}$ of random trading experts. After creation, the population is evaluated.

After the population is generated, weak individuals are eliminated and the bayesian network $\mathcal{B}$ is generated on the basis of the population $\mathcal{P}$ as described further.

After the bayesian network is generated, the population is generated anew according to the joint probability distribution defined by the bayesian network $\mathcal{B}$ and the process is repeated, until a termination condition is satisfied.

In experiments, the bayesian network is represented by a collection of conditional probability tables, which defines

```
BAYESIAN-OPTIMIZATION-ALGORITHM($\varrho$, $N$)
1  $\mathcal{P} \leftarrow$ RANDOM-POPULATION($N$);
2  POPULATION-EVALUATION($\mathcal{P}$, $\varrho$);
3  while not TERMINATION-CONDITION($\mathcal{P}$)
4  do
5      WEAK-INDIVIDUAL-ELIMINATION($\mathcal{P}$);
6      $\mathcal{B} \leftarrow$ BAYESIAN-NETWORK-GENERATING($\mathcal{P}$);
7      POPULATION-REGENERATING($\mathcal{P}$, $\mathcal{B}$, $N$);
8      POPULATION-EVALUATION($\mathcal{P}$, $\varrho$);
```

Figure 4: The Bayesian Optimization Algorithm designed to optimize an objective function ϱ with a population $\mathcal{P}$ composed of N trading experts.

the probability of each value of each gene under conditions referring to values of the other genes.

5.2 Bayesian Network Generating

In the algorithm, the bayesian network $\mathcal{B}$ is optimized according to *the K2 metric* evaluated on the basis of the current population [14]. It measures the fitting of the probability model represented by the bayesian network to the current population.

In the classic BOA, the bayesian network generating is performed by a greedy algorithm, which tries either to insert a new edge to the current network, or to delete an edge, or to change the direction of an edge in the current network, and repeats this until there is no improvement in the network.

6. VALIDATION OF THE METHODS

In this section, we discuss a number of experiments aiming at comparing the two algorithms in solving the optimization problem described in this paper.
In the discussion, two issues are addressed separately:
– the efficiency of the different algorithms in solving the optimization problems corresponding to discovering the stock market expertises;
– the financial relevance of the investment strategies based on the stock market expertises discovered by these algorithms.
Moreover, the two studies generate some by-products, that is to say:
– comparisons of the different algorithms in terms of optimal values reached by the objective function and in terms of computation time;
– comparisons of the realized financial return generated by the investment strategies discovered by these algorithms with the one generated by usual investment strategies.

All the experiments were performed on real-life data from the Paris Stock Exchange. Each experiment concerns a given stock, one of about 40 stocks constituting the CAC40 index, and a given training and test period. Thus, the problem of finding an optimal investment strategy for the chosen test period might be transformed to an optimization problem, the solution of which being a trading expert optimized on the chosen training period.

In order to assess and compare the effectiveness of the different optimization algorithms, each problem was solved several times with various algorithms. If e_0 denotes the best expert among all the experiments for the same problem (i.e. the same stock and the same training and test period), it will be considered as a quasi-maximum of the objective function ϱ for that problem. The ratio $\alpha = \varrho(e)/\varrho(e_0)$ then denotes the relative performance of an expert e with respect to the reference expert e_0. Efficient algorithms give experts with a ratio α close to 1.

For assessing the financial relevance of an investment strategy, deduced from an expert e, on the test period, its profitability is compared with the one generated by a Buy-and-Hold strategy, denoted as B&H. It consists in investing all the capital in stocks at the start of the period and keeping it until the end of the period under study. Although simple, the B&H strategy is an usual benchmark on financial markets.

In each experiment, the inputs were:
– a set of trading rules (the same set of 350 trading rules was used in all the experiments);
– a stock from the Paris Stock Exchange (randomly chosen one of about 40 stocks constituting the CAC40 index) and the financial time series of its price quotations;
– a performance measure and a randomly chosen training and test period (which define the objective function).

In most experiments, the test period contains 20 dates and the training period contains 60 preceding dates. Evaluation was carried out on about 40 data sets. Each data set consists of financial time series from the Paris Stock Exchange, including daily price quotations of a given stock over a period starting on January 4, 1999 and lasting on March 3, 2004.

In order to illustrate the methodology, we present a few experiments concerning the stock Renault. In all these experiments, the test period starts on May 7, 2002 and lasts on June 3, 2002 (20 days). The training period consists of 60 days preceding the test period, so it starts on February 7, 2002 and lasts on May 6, 2002. The objective function is constructed on the basis of the Sharpe ratio.

Table 2 compares the results obtained with the two algorithms (appearing in the first column). The next four columns concern the training period. The performance $\varrho(e)$ is reported in the second column; in the two following appear the returns on the optimized strategy and on the benchmark (B&H). The return on the CAC40 index is given for information on the next column. The remaining columns provide the returns on the two strategies and the return on the CAC40 on the test period.

In this example, ECGA was the most efficient – it generated the best expert with the highest performance. It overperformed the BOA and the B&H strategy on the test period, even if B&H generated a higher return on the training period.

In order to test the capabilities of the different algorithms, 100 optimization problems were prepared with randomly selected stocks and randomly selected training and test periods. Each algorithm was executed on each problem.

Table 3 reports the average values of α for each algorithm, the average number of iterations to obtain convergence and the mean computing time. The first row indicates the performance when random experts are generated with the best one being selected. ECGA proves to be the best performing algorithm, with the highest α, but also the most time-consuming.

Table 2: Characteristics of solutions to the optimization problem of discovering an optimal investment strategy with different algorithms for the stock Renault

	Training Period 02-07-2002 – 05-06-2002 (60 days)			
Algorithm	Performance	Return	B&H	CAC40
ECGA	0.0382	0.0193%	0.2979%	0.0167%
BOA	0.0347	0.0171%	0.2979%	0.0167%

	Test Period 05-07-2002 – 06-03-2002 (20 days)		
Algorithm	Return	B&H	CAC40
ECGA	0.0098%	0.0019%	-0.0161%
BOA	0.0083%	0.0019%	-0.0161%

Table 3: Comparisons of different algorithms in terms of highest values reached by the objective function and in terms of the computation time necessary to discover them

Algorithm	α	Iterations	Time
Random Generator	0.46 ± 0.08	100000	52 s
ECGA	0.94 ± 0.06	34	98 s
BOA	0.89 ± 0.04	51	47 s

As it may happen that the performance of an algorithm is not independent of market conditions, we divided the entire set of possible test periods into four categories, depending on the return of the stock during this period. Figure 5 shows the histogram of the return of the B&H strategy over periods of 20 days for the stock Renault (other stocks give similar results).

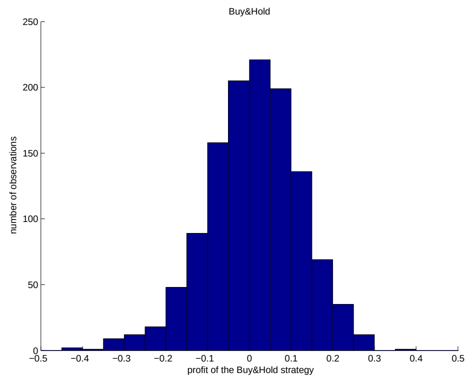

Figure 5: The histogram of the return of the Buy-And-Hold strategy over periods of 20 days for the stock Renault

In order to study in detail the financial relevance of the investment strategies based on the stock market expertises discovered by different algorithms, we test them separately on four types of stock market conditions:
1. extremely positive B&H, i.e. $0.05 \leq$ B&H,
2. positive B&H, i.e. $0.00 \leq$ B&H < 0.05,
3. negative B&H, i.e. $-0.05 \leq$ B&H < 0.00,
4. extremely negative B&H, i.e. B&H < -0.05.

Table 4 reports the excess return of the optimized strategy over the return of the B&H strategy, separately for each of these four types of stock market conditions. The mean excess return is positive for every type and every algorithm. ECGA always outperforms BOA. Obviously, efficiency of the optimized strategy varies with stock market conditions, but in most cases, the approach proposed in this paper largely overperforms the B&H strategy.

Table 4: Returns obtained by applying the optimal investment strategies discovered by different algorithms over the test period divided into four parts with respect to the Buy-And-Hold

	Algorithm	Return over B&H
1	ECGA	$0.09\% \pm 0.03$
	BOA	$0.07\% \pm 0.04$
2	ECGA	$0.23\% \pm 0.04$
	BOA	$0.17\% \pm 0.04$
3	ECGA	$0.15\% \pm 0.09$
	BOA	$0.14\% \pm 0.08$
4	ECGA	$0.20\% \pm 0.07$
	BOA	$0.16\% \pm 0.09$

7. CONCLUSIONS

In order to compare the two algorithms, a large number of experiments were performed on real-life data from the Paris Stock Exchange. Evaluation was performed on a set of 350 trading rules applied to financial time series including price quotations of a given stock. The results show that the two algorithms perform well, compared to a static strategy like B&H. ECGA leads to the best results but this algorithm is time consuming and, consequently, not well-suited for real-time applications. BOA, however it leads to slightly worse results, is faster and more suitable for real-time trading, where the computation time is very important. Therefore, BOA offers an interesting alternative in terms of the trade-off between performance and computation time.

8. REFERENCES

[1] Baluja, S., *Population-Based Incremental Learning: A Method for Integrating Genetic Search Based Function Optimization and Competitive Learning*, Research Report CMU-CS-94-163, Carnegie Mellon University, 1994.

[2] Brabazon, A., O'Neill, M. *Biologically Inspired Algorithms for Financial Modelling*, Springer, 2006.

[3] Cottle, S., Murray, R., F., Block, F., E., *Security Analysis*, McGraw-Hill, 5th Edition, 1988.

[4] Dempsey, I., O'Neill, M., Brabazon, A., *Adaptive Trading with Grammatical Evolution*, Proceedings of the 2006 Congress on Evolutionary Computation, CEC 2006, IEEE, 2006, pp.2587-2592.

[5] Goldberg, D.,E., *Genetic Algorithms in Search, Optimization and Machine Learning*, Addison Wesley, 1989.

[6] Harik, G., *Linkage Learning via Probabilistic Modeling in the ECGA*, IlliGAL Research Report 99010, University of Illinois at Urbana-Champaign, 1999.

[7] Harik, G., R., Lobo, F., G., Goldberg, D., E., *The Compact Genetic Algorithm*, IlliGAL Research Report 97006, University of Illinois at Urbana-Champaign, 1997.

[8] Korczak, J., Lipinski, P., *Evolutionary Building of Stock Trading Experts in a Real-Time System*, Proceedings of the 2004 Congress on Evolutionary Computation, CEC 2004, IEEE, 2004, pp.940-947.

[9] Lipinski, P., *Clustering of Large Number of Stock Market Trading Rules*, Proceedings of the 16th Symposium on Computational Statistics, CompStat 2004, Springer, 2004, pp.1397-1404.

[10] Lipinski, P., Korczak, J., *Performance Measures in an Evolutionary Stock Trading Expert System*, Proceedings of the International Conference on Computational Science, ICCS 2004, LNCS 3039, Springer, 2004, pp.835-842.

[11] Lobo, F., G., Harik, G., R., *Extended Compact Genetic Algorithm in C++*, IlliGAL Research Report 99016, University of Illinois at Urbana-Champaign, 1999.

[12] Loraschi, A., Tettamanzi, A., *An Evolutionary Algorithm for Portfolio Selection within a Downside Risk Framework*, Forecasting Financial Markets, ed. Christian L. Dunis, Wiley, 1996, pp. 275-286.

[13] Murphy, J., *Technical Analysis of the Financial Markets*, NUIF, 1998.

[14] Pelikan, M., Goldberg, D., Cantu-Paz, E., *BOA: The Bayesian Optimization Algorithm*, IlliGAL Research Report 99003, University of Illinois at Urbana-Champaign, 1999.

[15] Pelikan, M., Sastry, K., Goldberg, D., *Sporadic model building for efficiency enhancement of hierarchical BOA*, Proceedings of the Genetic and Evolutionary Computation Conference, ACM, 2006, pp.405-412.

[16] Tsang, E., Li, J., Markose, S., Er, H., Salhi, A., Iori, G., *EDDIE In Financial Decision Making*, Journal of Management and Economics, Vol.4, No.4, 2000.