Available online at www.sciencedirect.com

SciVerse ScienceDirect

Procedia Computer Science 9 (2012) 1306 – 1313

Procedia Computer Science

International Conference on Computational Science, ICCS 2012

Parallel genetic algorithms for stock market trading rules

Janko Straßburg[a], Christian González-Martel[b], Vassil Alexandrov[c]

[a]University of Reading, UK and Barcelona Supercomputing Center, c/ Jordi Girona, 29, Edifici Nexus II, 08034 Barcelona, Spain
[b]Departamento de Métodos Cuantitativos en Economía y Gestión, Universidad de Las Palmas de Gran Canaria, Spain
[c]ICREA and Barcelona Supercomputing Center, c/ Jordi Girona, 29, Edifici Nexus II, 08034 Barcelona, Spain

Abstract

Finding the best trading rules is a well-known problem in the field of technical analysis of stock markets. One option is to employ genetic algorithms, as they offer valuable characteristics towards retrieving a "good enough" solution in a timely manner. However, depending on the problem size, their application might not be a viable option as the iterative search through a multitude of possible solutions does take considerable time. Even more so if a variety of stocks are to be analysed.

In this paper we concentrate on the enhancement of a previously published genetic algorithm for the optimisation of technical trading rules, using example data from the Madrid Stock Exchange General Index (IGBM).

Keywords: Genetic Algorithms, Stock Market, Parallel, Technical Trading Rules

1. Introduction

Financial markets all over the world are relying on computers to analyse market data, give recommendations and make transactions. Nowadays humans alone would not be able to handle the vast amount of data that is constantly being produced and transmitted, to make an informed decision within a reasonable time span. To overcome this problem, computer trading systems have been developed and are constantly being improved upon.

This paper deals with the enhancement of an algorithm for the optimisation of technical trading rules (TTRs), presented and demonstrated on data from the Madrid Stock Exchange General Index (IGBM) in [1]. Background information about the research that has been done so far in this area will be presented in the next section, the concept of technical analysis will be dealt with in Section 3. Genetic algorithms are considered in Section 4. Finally, the experiments undertaken and their outcomes are presented in Sections 5 & 6, with Section 7 concluding and giving an outlook on future work.

2. Background

In financial markets it is important and valuable to be capable of foreseeing the trend of prices or even to forecast them. Financial market traders use technical and fundamental analysis to assist them in their decisions. Both are

Email addresses: j.strassburg@reading.ac.uk (Janko Straßburg), cgonzalez@dmc.ulpgc.es (Christian González-Martel), vassil.alexandrov@bsc.es (Vassil Alexandrov)

doi:10.1016/j.procs.2012.04.143

used widely for decision making, but while fundamental analysis is mainly based on economic factors, the technical analysis is based on psychological factors. Behind the use of algorithms and charts resides the psychology of the market.

Technical analysis uses only historical data, usually consisting of only past prices, but sometimes also includes volume to determine future movements in financial asset prices. This type of analysis therefore stands in contrast to fundamental analysis, which relies on financial statements and the economic surroundings of the stock option in question to assess its performance. Both theories are disputed by the efficient-market hypothesis which essentially maintains that stock market prices are unpredictable.

However, in the academic setting, technical analysis contradicts the EHM. In its weak form this hypothesis claims that prices on traded assets already reflect all past publicly available information. The works of Fama [2] and Alexander [3] were unable to find evidence of profitability and thus concluded that technical analysis is not useful. Therefore, the most appropriate investment strategy according to them is the long-term oriented buy and hold strategy that consists of simply buying and then holding the asset.

Yet in recent years studies have been presented that try to dispute these conclusions using different techniques, most of them based on machine learning. These methods are a branch of artificial intelligence research, concerned with designing and developing algorithms that able to use empiric data to evolve certain behaviours. It has been successfully demonstrated that artificial intelligence can be used in financial trading. Artificial neural networks are able to judge intuitively, detect data patterns that elude conventional analytical techniques, and learn from past mistakes [4, 5, 6]. The usage of genetic algorithms to find technical trading rules has been shown earlier and proven to be a worthy undertaking, yielding excess returns when compared to a simple buy-and-hold strategy [1, 7, 8]. Furthermore, there have been undertakings recently to enhance genetic algorithm based technical trading systems, for example by optimizing the parameters of technical rules, as shown in [9]. Also the design of the genetic algorithm itself and its influence on the results in the context of technical trading systems has been studied [10]. In comparison with some of the before mentioned investigations that have been looking at the American S&P 500 index, this work is researching European markets, specifically the Spanish one with data from the Madrid stock exchange.

3. Technical Analysis

Technical analysis leads to certain rules that are applied to assist the financial market traders in their decisions. It does not depend on characteristics of a company or its estimated value, but simply relies on statistical data that has been generated through market transactions. These rules are based on charts or technical indicators that are mathematical formulas, which use historical data on price and/or volume as input and return a signal. This can then be used by traders to take a position in the market.

A considerable amount of work has provided support for the view that simple technical trading rules (TTRs) are capable of producing valuable economic signals [see 11, 12, 13, 14, among others].

The most common and simplest rule is the moving average (MA) rule. A moving average (MA) is simply an average of current and past prices over a specified period of time. The aim of using a MA is to capture the underlying trend, erasing the noise in the price series.

We consider a generalised moving average (GMA) rule. It consists of the comparison between two moving averages, a short MA and a long MA. The second one captures the main trend, while the first one captures the short swings. If the short MA rises above the long MA, then the asset is bought and held until the short MA falls below the long MA, at which time the asset is sold. A filter is added to this rule with intend to suppress false signals. A signal is only considered to be valid, if the short MA stays above or below the long MA for a certain amount of time.

The rule to determine buy and hold signals can be expressed in the formula:

$$S\left(\Theta\right)_t = MA\left(\theta_1\right)_t - \left(1 + \left(1 - 2S_{t-1}\right)\theta_3\right) MA\left(\theta_2\right)_t \tag{1}$$

where $\Theta = [\theta_1, \theta_2, \theta_3]$ denotes parameters associated with the GMA rule and $MA\left(\theta\right)$ is the MA indicator, defined as

$$MA_t\left(\theta\right) = \frac{1}{\theta}\sum_{i=0}^{\theta-1} P_{t-i}, \qquad t = \theta, \theta+1, \ldots, N$$

In this equation, θ_1 is the length of a short MA and θ_2 is the length of the long MA, representing the number of days used to calculated the MAs, while θ_3 is the filter parameter, that measured in basic points. 100 basic points are equivalent to one per cent.

The trading rule considered in this study is based on a simple market timing strategy, consisting of investing total funds in either the stock market or a risk free security. This rule is used to classify each day into periods "in" (earning the market return) or "out" of the market (earning the risk-free rate of return). The trading strategy specifies the position to be taken the following day, given the current position and the "buy" or "sell" signals generated by rule. On the one hand, if the current state is "in" (i. e., the investor is holding shares in the market) and the share prices are expected to fall on the basis of a sell signal generated by the rule ($S\left(\Theta\right)_t \leq 0$), then shares are sold and the proceeds from the sale are invested in the risk free security [earning the risk-free rate of return $r_f(t)$]. On the other hand, if the current state is "out" and the rule indicates that share market prices will increase in the near future, the rule returns a "buy" signal $S\left(\Theta\right)_t > 0$) and, as a result, the risk free security is sold and shares are bought [earning the market rate of return $r_m(t)$]. Finally, in the remaining two cases [stock held and buy signal; stock sold and sell signal], the current state is preserved.

Three different MA rules that can be derived individually by imposing some restrictions on this equation are nested inside the GMA rule (A MA with $\Theta = 1$ is the price series):

1) Simple MA:

$$\theta_1 = 1, \theta_2 > 1, \theta_3 = 0$$
$$S(\Theta)_t = P_t - MA(\Theta_2)_t$$

2) Filtered MA:

$$\theta_1 = 1, \theta_2 > 1, \theta_3 > 0$$
$$S(\Theta)_t = P_t - (1 + (1 - 2S_{t-1})\theta_3)MA(\theta_2)_t$$

3) Double MA:

$$\theta_1 > 1, \theta_2 > \theta_1, \theta_3 = 0$$
$$S(\Theta)_t = MA(\theta_1)_t - MA(\theta_2)_t$$

4. Genetic Algorithms

Genetic algorithms (GAs), are a class of algorithms working on problems that cannot be solved in a deterministic and analytical manner. The main idea is to continuously generate varying solutions to a problem while combining, mutating and evaluating them. Using this approach, it is possible to very quickly converge towards a desired behaviour and to solve the original problem. However, the algorithm will not necessarily return the overall optimal solution, although it generates a set of "good enough" solutions. In the financial world this is a benefit, as analysts require a quick, good solution instead of undergoing the lengthy process to find the overall optimal solution. Especially as a once found optimal solution is unlikely to stay the best one with future development in the specific market space.

GAs, developed by Holland [15], are a class of adaptive search and optimization techniques that have the advantage of being able to evaluate loss functions associated with the predictor parameters with no assumption regarding the continuity or differentiability of the loss function. Especially in the last decades they have attracted more and more interest in financial analytics.

A GA starts with a population, called chromosomes, of randomly generated solution candidates, which are evaluated in terms of an objective function. These candidates are usually represented by vectors consisting in binary digits. Promising candidates, as represented by relatively better performing solutions, are then combined through a process of binary recombination, referred to as crossover. Finally, random mutations are introduced to safeguard against the loss of genetic diversity, avoiding local optima. Successive generations are created in the same manner and evaluated using the objective function until a well-defined criterion is satisfied.

An obvious characteristic that influences the performance of a genetic algorithm is the initial availability of candidates and the resulting broadness of possible choices for recombination resulting from within. However, the number of generated chromosomes naturally impacts on the computational performance needed to process the generations. A low initial number of competing chromosomes only requires moderate computation but also limits the available "gene pool", thus not promising a very high chance of good or acceptable results. In the opposite case, when generating a very high initial population, the computational demand is obviously much higher. This is due to a higher number of chromosomes that need to be evaluated, combined and mutated. It does however provide an extensive enough base, using which it is far more likely to obtain good results.

In order to determine which solution candidates are allowed to participate in the crossover and undergo possible mutation, we apply the GENITOR selection method proposed by [16]. This approach involves ranking all individuals according to performance and then replacing the poorly performing individuals by copies of better performing ones. In addition, we apply the commonly used single point crossover, consisting in randomly pairing candidates that survived the selection process, and randomly selecting a break point at a particular position in the binary representation of each candidate. This break point is used to separate each vector into two subvectors. The two subvectors to the right of the break point are exchanged between the two vectors, yielding two new candidates. Finally, mutation occurs by randomly selecting a particular element in a certain vector. If the element is a one it is mutated to zero, and vice versa. This occurs with a very low probability in order not to destroy promising areas of search space.

After sufficient iterations, this approach of selection and recombination leads to candidates that are more and more suited to solve the problem at hand. As the algorithm itself does not contain a stop criterion, one has to be included so that the evolution halts once it is satisfied. This is usually a maximum number of iterations as time and compute restraints have to be met. As these are the limiting factors to improve the quality of the results and extend the possible search space, a parallel implementation of a GA will be presented, making better use of the available computational resources and therefore producing results quicker.

The genetic algorithm procedure can be summarised by the following steps:

1. Create an initial population of candidates randomly
2. Evaluate the performance of each candidate in terms of an objective function
3. Select the candidates for recombination
4. Perform crossover and mutation
5. Evaluate the performance of the new candidates
6. Return to step 3, unless a termination criterion is satisfied

5. Experimental Setup

The majority of studies that have worked with TTRs have ignored the issue of parameter optimisation, leaving them open to the criticism of data-snooping and the possibility of a survivorship bias [see 17, 18, respectively]. To avoid this criticism, a more objective and valid approach consists in choosing TTRs based on an optimisation procedure utilising in-sample data and testing the performance of these rules out-of-sample. In this sense, a genetic algorithm is an appropriate method to discover TTRs, as shown in [19].

To be able to generate meaningful technical trading rule parameters, the underlying genetic algorithm has to be set up in a defined way. The first step is the generation of an initial population in which every individual is a vector

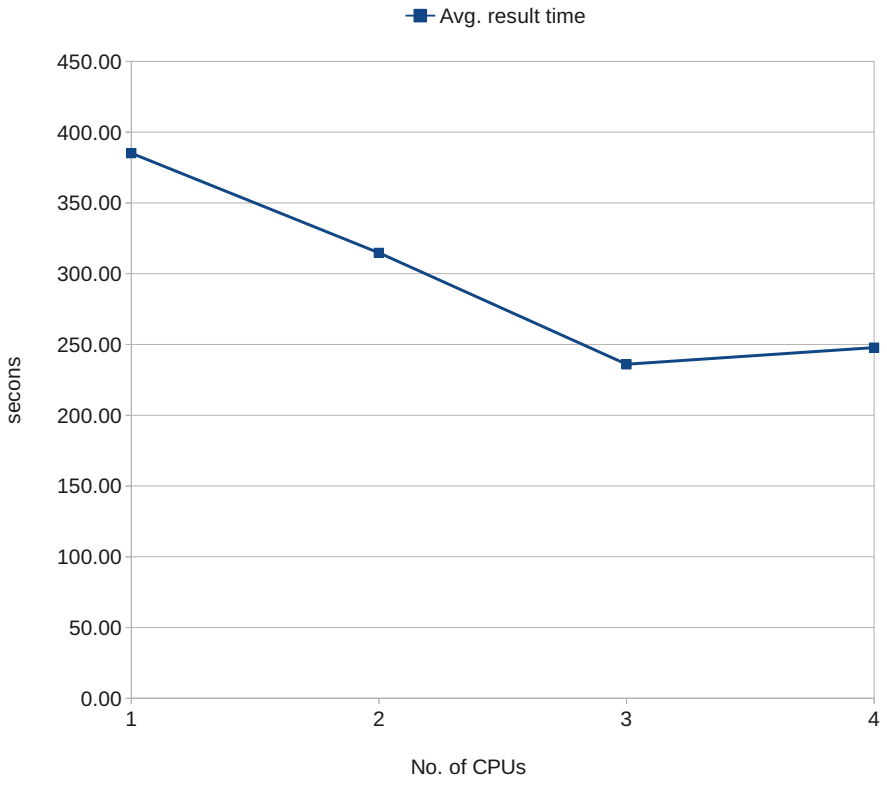

Figure 1: Average result time scaling

that contains three parameters for a TRR. These are the short MA, long MA and the filter parameter that prevents premature buy and sell signals.

The settings that have been used in this particular algorithm are a crossover probability of 0.6%, a mutation probability of 0.005% and a maximum number of iterations set to 200. An initial population of 120 chromosomes was generated and then underwent the cycle of evaluation, selection, crossover and mutation until the following generations in unison converged on an optimal chromosome or the maximum number of iterations was exhausted. The parameter selection was guided by previous studies and experimentation with different values [20, 1].

The data that was used as a basis is from the Madrid Stock Exchange General Index (IGBM) and consists of the daily closing prices of its general index as well as the daily 3 month rate in the interbank deposits markets. In total the data is comprised of 4376 observations and covers a period of 25 years from 1972 onwards. This dataset has been split into an in-sample optimisation period from 2nd of January 1972 to the 16th December 1988, and an out-of-sample test period ranging from the 16th of December 1988 to the 15th November 1997. Therefore, 2188 observations were allocated for each block.

Within the algorithm, trading rules signal whether to be "in" or "out" of the market at any given trading day during the investigated time scale. When a stock had been acquired (or held), the market return $rm_t = ln\left(\frac{P_t}{P_{t-1}}\right)$ is earned. Contrariwise, if the stock was sold (or refrained from buying), it earned the risk free rate rf_t. The function to evaluate the trading rules and calculate the returns is given in equation 2, with c denoting a transactional cost of 0.10%, T the number of transactions and S being the current time point in the series.

$$r_{tr} = \sum_{t=1}^{N} S_{t-1} rm_t + \sum_{t=1}^{N} (1 - S_{t-1}) rf_t - T * c \tag{2}$$

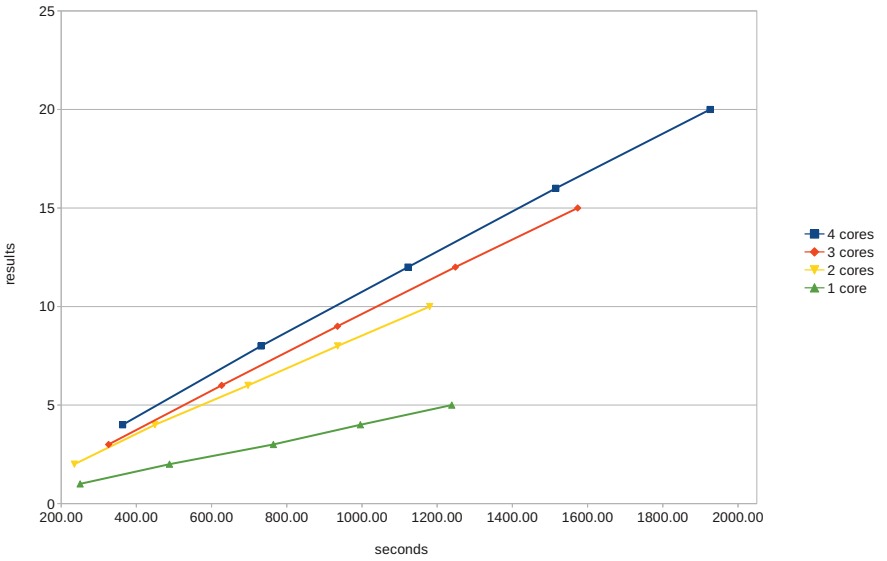

Figure 2: Average result time scaling; Fixed scaling

5.1. Selection

The selection process is vital for the outcome of the algorithm as it determines which chromosomes are carried over into the next generation. Hence it is important not to discard promising candidates and remove under-performing ones from the population.

To ensure the best chromosomes are selected, the annualized return rates of all chromosomes from a population are noted and the mean return rate calculated. Individual chromosomes are then assessed and their return rate compared to the annual mean. The chromosome is replicated into the new population, depending on how much higher its annualized return rate is compared to the mean. This ensures that the best chromosomes are given increased chances to be selected for a crossover.

6. Empirical Results

The original program to search for technical trading rules using genetic algorithms has been written in Matlab and has been modified to run in a simple parallel fashion. For testing, a computer with a quad core Intel i7 processor and 8 GB RAM has been used, running Matlab R2011a with added Parallel Computing Toolbox. The algorithm that was presented to find optimal technical trading rules in [1] has been developed further and is used in this paper to demonstrate possibilities of runtime enhancement. Several computations have been executed with varying degrees of parallelism and the results are presented in Figures 1, 2 and 3 respectively.

The average result time for one set of trading rules, utilizing from one up to four cores of the machine, is depicted in Figure 1. To account for possible glitches in the time measurement, all experiments have been run five times and then averaged. As expected, it can be seen that the response time is getting shorter with the usage of more CPU cores. In the case of all four cores being used however, the time needed to calculate the result is increasing again. This can be accounted for due to the increased communication needed between the CPU and the main memory, therefore exposing this as a bottleneck in the test machine.

One of the main incentives of parallelisation is to compute more results in the same time frame, or spent less time than before to compute an individual result. Figures 2 and 3 show the scaling characteristics for the modified algorithm in terms of results over time. It can clearly be seen from the fixed scaling image that the usage of more compute cores leads to an increased number of retrieved trading rules during the same time frame. A trend that is conceivable from the average result scaling time can be seen clearly in the logarithmic scaling in Figure 3. Due to the constantly increasing internal communication between the compute cores and the main memory, the slope of the scaling is decreasing with

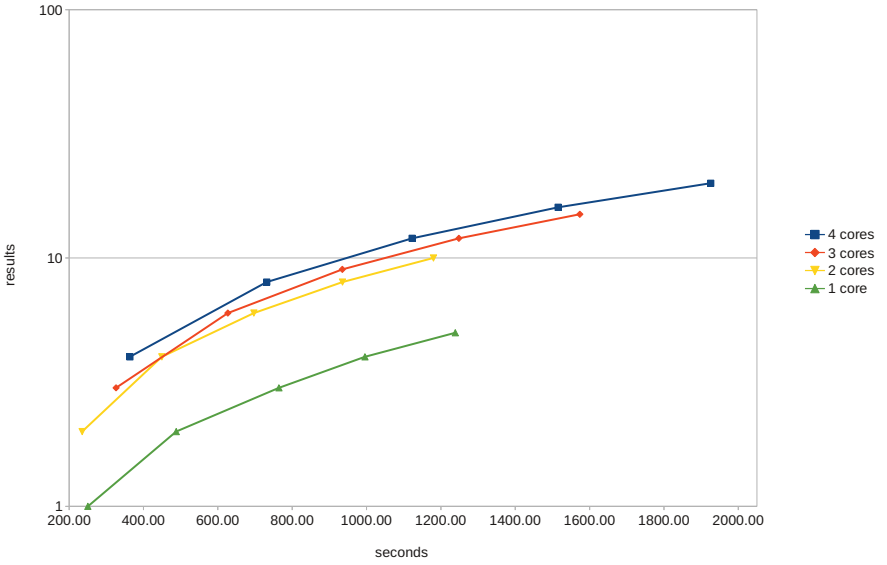

Figure 3: Average result time scaling; Logarithmic scaling

the addition of more and more employed cores. This can be attributed to the current implementation of the algorithm and needs to be addressed in a future reimplementation. Nevertheless, an improvement in speed over the original version of the code is noticeable.

7. Conclusions and Future Work

It has been shown that the considered genetic algorithm for the generation of technical trading rules can be improved upon by introducing parallelisation. The method shown in this paper sped up the process of computation, enabling the generation of more results in the same time and making it more feasible to analyse larger problems.

Several additional starting points for further improvements have been identified during the work on the program code. Those require an extensive rewrite or reimplementation of some of the core functions of the algorithm and will be presented in a future paper. One very promising candidate is the implementation of a lookup table to consecutively store and retrieve computed characteristics of chromosomes that are shared between individuals within a population. This will speed up the whole process immensely and therefore provide results even more timely.

Another promising approach is the possible exploitation of compute capabilities provided by graphics processing units (GPUs) to accelerate the algorithm. Matlab supports NVIDIA CUDA[1] enabled graphics cards that nowadays can widely be found in consumer hardware. Therefore, an increased execution speed is possible without needing access to a costly dedicated computational cluster.

Acknowledgements

Christian González Martel kindly acknowledges Project ECO2011-23189 for financing part of his investigations.

References

[1] F. Fernández-Rodríguez, C. González-Martel, S. Sosvilla-Rivero, Optimization of technical rules by genetic algorithms: evidence from the madrid stock market, Applied Financial Economics 15 (11) (2005) 773–775.

[2] E. Fama, M. Blume, Filter rules and stock market trading, Journal of Business 39 (1966) 226–241.

[1] CUDA™ is a parallel computing platform and programming model invented by NVIDIA.

[3] S. Alexander, Price movements in speculative markets: trends or random walks?, Industrial Management Review 2 (1961) 7–26.
[4] R. R. Trippi, E. Turban, Neural Networks in Finance and Investing: Using Artificial Intelligence to Improve Real World Performance, McGraw-Hill, Inc., New York, NY, USA, 1992.
[5] M. Matilla-Garca, C. Argello, A hybrid approach based on neural networks and genetic algorithms to the study of profitability in the spanish stock market, Applied Economics Letters 12 (5) (2005) 303–308. arXiv:http://www.tandfonline.com/doi/pdf/10.1080/1350485042000329103, doi:10.1080/1350485042000329103.
URL http://www.tandfonline.com/doi/abs/10.1080/1350485042000329103
[6] F. Fernández-Rodríguez, C. González-Martel, S. Sosvilla-Rivero, On the profitability of technical trading rules based on artificial neural networks:: Evidence from the madrid stock market, Economics Letters 69 (1) (2000) 89–94.
URL http://ideas.repec.org/a/eee/ecolet/v69y2000i1p89-94.html
[7] F. Allen, R. Karjalainen, Using genetic algorithms to find technical trading rules, RODNEY L WHITE CENTER FOR FINANCIAL RESEARCH-WORKING PAPERS-.
[8] M. Kaucic, Investment using evolutionary learning methods and technical rules, European Journal of Operational Research 207 (3) (2010) 1717–1727.
[9] S. Papadamou, G. Stephanides, Improving technical trading systems by using a new matlab-based genetic algorithm procedure, Mathematical and computer modelling 46 (1-2) (2007) 189–197.
[10] L. Nunez-Letamendia, Fitting the control parameters of a genetic algorithm: An application to technical trading systems design, European Journal of Operational Research 179 (3) (2007) 847–868.
[11] W. A. Brock, J. Lakonishhock, B. LeBaron, Simple technical trading rules and the stochastic of stock returns, Journal of Finance 47 (1992) 1731–1764.
[12] K. Bessembinder H. y Chan, The profitability of technical trading rules in the Asian stock markets, Pacific-Basic Finance Journal 3 (1995) 257–284.
[13] T. C. Mills, Technical analysis and the london stock exchange: Testing trading rules using the ft30, International Journal of Finance & Economics 2 (4) (1997) 319–31.
URL http://ideas.repec.org/a/ijf/ijfiec/v2y1997i4p319-31.html
[14] F. Fernández Rodríguez, S. Sosvilla Rivero, J. Andrada Félix, Análisis técnico en la bolsa de madrid, Moneda y Crédito 213 (99-05) (1999) 11–37.
[15] J. H. Holland, Adaptation in Natural and Artificial Systems - An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence, University of Michigan Press: Ann Arbor, MI., 1975.
[16] D. Whitley, The GENITOR algorithm and selection pressure: Why rank-based allocation of reproductive trial is best, in: D. J. Schaffer (Ed.), Proceedings of Third International Conference on Genetic Algoritms, Morgan Kaufman, California, 1989, pp. 116–121.
[17] A. Lo, MacKinley, Data snooping biases in test of financial asset pricing models, The Review of Financial Studies 3 (1990) 431–467.
[18] S. J. Brown, W. N. Goetzmann, S. A. Ross, Survival, Journal of Finance 50 (3) (1995) 853–73.
URL http://ideas.repec.org/a/bla/jfinan/v50y1995i3p853-73.html
[19] F. Allen, R. Karjalainen, Using genetic algorithms to find technical trading rules, Journal Finance Economics 51 (2) (1999) 245–271.
[20] R. Bauer, Genetic algorithms and investment strategies, Vol. 19, John Wiley & Sons Inc, 1994.