Pretests for Genetic-Programming Evolved Trading Programs: "zero-intelligence" Strategies and Lottery Trading

Shu-Heng Chen[1] and Nicolas Navet[1,2]

[1] AI-ECON Research Center, Department of Economics,
National Chengchi University, Taipei, Taiwan 11623
chchen@nccu.edu.tw
[2] LORIA-INRIA, Campus-Scientifique, BP239, F-54506 Vandoeuvre, France
nnavet@loria.fr

Abstract. Over the last decade, numerous papers have investigated the use of GP for creating financial trading strategies. Typically in the literature results are inconclusive but the investigators always suggest the possibility of further improvements, leaving the conclusion regarding the effectiveness of GP undecided. In this paper, we discuss a series of pretests, based on several variants of random search, aiming at giving more clear-cut answers on whether a GP scheme, or any other machine-learning technique, can be effective with the training data at hand. The analysis is illustrated with GP-evolved strategies for three stock exchanges exhibiting different trends.

1 Motivation and Introduction

The computational intelligence techniques such as genetic programming[1], with their continuous advancement, persistently bring us something positive to expect, and incessantly push the application domain to more challenging issues. However, sometimes, the costs and benefits of using this advanced CI techniques are uncertain. Usually the benefits are not assured, while the costs is immediate. On the one hand, the CI techniques are frequently used as intensive search algorithms, which inevitable are computationally demanding, and take up a great amount of computational resources. On the other hand, whether there is a needle in the sea remains to be dubious.[2] Certainly, if such a needle does not exist at all, the all efforts are made with no avert. Given this asymmetry between costs and benefits, it would be economical, at the first stage, to test the existence of such a needle before a full-fledged version of search is applied. We call this procedure a *pretest*.

[1] Although, in this paper, we only focus on genetic programming, but the general ideas and some specific implementations may also be applicable to other computational intelligence technique used to induce trading strategies.

[2] For example, in the financial application domain, it can be particularly due to the efficient market hypothesis or the no-arbitrage condition.

I. King et al. (Eds.): ICONIP 2006, Part III, LNCS 4234, pp. 450–460, 2006.

The pretest procedure proposed here is similar in a sense to the pretests used in econometrics where the estimator of an unknown parameter is chosen on the basis of the outcome of a pretest ([1]). Pretesting, also known as "data-snooping" in finance, serves classically for selecting the right model that will be used later on for forecasting purpose ([2,3]). More broadly, pretesting can be considered as a practice of a sequential decision-making process, which is used when the decision involves a great deal of uncertainty, and the costs of making a wrong decision is huge. [3] In this case, in the first stage, we would like to spend some limited resources in probing to gain some initial information, e.g. the distribution of a very uncertain environment, while in later stages, we make our decision based on the gauged distribution.

The reason behind prestesting is very intuitive, and [4] is the first who applies this idea to the financial application of genetic programming (GP). [4] proposed a measure known as the η statistic. The η statistic is a measure of predictability. Basically, using simple (vanilla) version of GP, one can first gauge the predictability by η. When η is low to zero, it indicates that there is nothing to forecast. So, the use of full-fledged GP is not advised. The virtue of this doing is to distinguish *two kinds of possibilities* when we see a failure of an initial attempts based on simple GP. First, the series itself has nothing to forecast; second, GP has not been used appropriately. Understanding this distinction can result in big differences in our second stage of the decision. For the former case, we may simply give up the further search to avoid a waste. For the latter, we should keep on exploring different deliberations of GP to search for potential gains before a final conclusion can be made. In either case, we have a clear-cut situation. However, when a pretest is absent, we become less conclusive: we are no longer sure whether it is due to the non-existence of the needle, or the improper use of GP.

Unfortunately, in most financial trading applications of GP, a pretest has been largely neglected.[4] We think that this negligence may cause many observed inconclusive results. Typically, what happens is that the results with GP are not very convincing, but the investigators always suggest directions for further improvements, leaving the actual conclusion regarding the effectiveness of GP undecided. Therefore, this study attempts to provide practical pretesting procedure aiming at reducing the number of cases where the conclusion is inconclusive.

Needless to say, there are various ways to implement different pretesting. For example, the η statistic mentioned above can be used as a pretest, as [4] did, but that is mainly applied to forecasting time series. A series being predictable does not necessarily imply that we can develop profitable trading strategies. For example, the fluctuation is not volatile enough to cover the round-trip transaction

[3] The problem of sequential decision making under incomplete knowledge has been studied by researchers in various fields, such as optimal control, psychology, economics, and game theory.

[4] This may not be completely so. In fact, most earlier studies selected the buy-and-hold strategies or a risk-free investment (e.g., treasury bills) as the benchmark. However, the conclusion that "GP performs better than buy-and-hold in a bearish market and worst in a bullish market" is often found in the literature. This shows the limits of choosing buy-and-hold as a pretest. See, for example, [5].

costs. Consequently, literature of forecasting with GP and literature of trading with GP usually are separated. Therefore, in this paper, we attempt to develop pretest procedures more suitable for the trading purpose.

More precisely, we will propose several different styles of pretests, which when put together can help us decide whether there are hidden patterns to discover and whether GP is designed properly to do the job. The essential idea underlying all proposed pretests is to compare the performance of GP with random trading strategies or behavior. However, as we shall see in Section 2, just making trading strategies or trading behavior arbitrarily random is not sufficient to give a fair and informative comparison. To do so, some constraints are expected, and the intriguing point is how to impose these constraints properly.

The rest of the paper is organized as follows. Section 2 provides a detail formulation of four pretests. The first three concerns the trading strategies, whereas the last one concerns the trading behavior. Normally, trading behavior comes from trading strategies, and they cannot be separated; but, when randomness is introduced, difference between the two can arise. In particular, in the vein of algorithmic complexity, random trading strategies can imply trading behavior actually using knowledge, while random trading behavior presumably exclude such possibility. We, therefore, intentionally distinguish the two by called the former *zero-intelligence strategies*, and the latter *lottery trading*. Section 3 discuss how to use these proposed tests together to make a better judgement given the initial results we have. Section 4 illustrates the proposed pretests based on the real detail and the experimental designs detailed in the appendix. Section 5 makes the concluding remarks.

2 Pretests : Description and Rationale

In this section, we describe a series of 4 pretests and discuss their purpose and implementation. Out of the 4 pretests, we highlight that 2 are of particular interest and, as shown in section 3, enable us to gain complementary knowledge on the data under study and on the efficiency of the GP implementation. In the following, we consider GP with a validation stage before the actual testing on the out-of-sample data. Validation means that the best rules induced on the training interval are further selected on unseen data, the validation period, before being applied out-of-sample. The validation step is a device to fight overfitting[5] that has been widely used in earlier GP work (see for instance [7,8]). Note that our pretest proposals remain valid for GP without validation step except that pretest 2 replaces pretest 1, which requires validation.

2.1 GP Versus Equivalent Intensity Random Search

The basic idea is here to compare the outcome of GP with an *equivalent intensity random search*. We say that two search algorithms are equivalent in terms

[5] The actual effectiveness of validation is still an open question, see [5] and [6].

of search intensity if their execution leads to the evaluation of the same number of different trading strategies on the training data. For instance, let us consider GP with the parameters chosen for this study: a population of 500 individuals evolved over 100 generations. In first approximation, the equivalent random search (ERS) would consist in evaluating 50,000 randomly created solutions. In practice, search algorithms sometimes rediscover identical solutions over the course of their execution. This can be detected by keeping track of all created individuals since the beginning of the execution, and doing so useless fitness evaluations can be skipped, which actually saves computing time when the fitness function is rather time-consuming as it is in our context. Since, computationally speaking, what is preponderant is the fitness evaluation, and since the extent to which GP re-discovers the same individuals is very dependent upon the implementation, we impose that our definition of equivalent search intensity only accounts for unique individuals, *i.e.* individuals which require evaluation. We consider two solutions to be different if their expression is *syntactically different*[6], in our context, if the trees representing the programs are different.

The three following pretests compare GP with a random search with and without training and validation stage. In the latter search technique, the biologically inspired evolution process of GP is simply replaced by the creation of solutions at random. Since with random search the strategies do not benefit from the "intelligence" resulting from the evolution or learning process, we dub randomly created solutions *zero-intelligence trading strategies.*

For each pretest i, we formulate the null hypothesis $\mathcal{H}_{i,0}$ that GP does not outperform the technique it is compared with at pretest i, where the alternative hypothesis is denoted $\mathcal{H}_{i,1}$.The experiments will provide us with the answer on whether $\mathcal{H}_{i,0}$ should be rejected in favour of $\mathcal{H}_{i,1}$ or not.

Pretest 1: GP Versus Equal Search Intensity Random Search with Training and Validation Stage. The implementation of the random search strategy is straightforward: parameters of GP are set in such a way that only the initial generation, where individuals are created at random, is used. The size of the initial population is adjusted so that the resulting search intensity is identical to the one of the regular GP.

[6] Two individuals can be syntactically different while being equivalent in the sense that they lead to equivalent trading decisions, the equivalence could thus be also defined in terms of semantics. With symbolic simplification using rewriting rules and interval arithmetic on the function arguments, one could detect that some syntactically different individuals are in fact semantically identical. However, there is no way to make sure that all duplicates will be detected and the implementation of this procedure would be so complex and time consuming at run time that, in our opinion, a definition based on semantics would be of little practical interest. Alternatively, the equivalence in search intensity could be defined in terms of equivalent computing time, however there is such a difference of complexity between a full-fledged GP implementation and random search that it is hard to imagine how we can ensure that the two implementations have been optimized in a similar manner, while a better implementation of GP for instance may lead to an opposite conclusion.

- **Hypothesis $\mathcal{H}_{1,0}$ cannot be rejected:** the first explanation that can be envisaged is that, GP or not, there is nothing essential to be learned from the past. It that case GP would strongly "overfits" the training data, possibly explaining that its out-of-sample performance is worse than with a random search. This can be due by the market being efficient or because the training interval is very dissimilar to the out-of-sample[7]. Another explanation is that the GP machinery is not working properly, for instance due to a wrong choice of the function/terminal sets, because the parameters are inappropriate (e.g. too low search intensity), or the genetic operators unable to create better-than-random individuals.
- **Hypothesis $\mathcal{H}_{1,0}$ is rejected in favour of $\mathcal{H}_{1,1}$:** there may be something to learn from the past and GP, with the chosen parameters, may be effective in that task.

Rejecting $\mathcal{H}_{1,0}$ is of course a first indication of the efficiency of GP but we cannot rule out the case where there would nothing useful to learn on the data at hand and GP would beat random search by mere luck. We will see in Section 3, that further investigation may provide additional evidence to answer that question.

Pretest 2: GP Versus Equal Search Intensity Random Search with Training But Without Validation Stage. Here, the best random solutions on the training interval are applied directly to the out-of-sample period. With regards to pretest 1, pretest 2 could give us some insight about how effective is validation as a device to fight against overfitting. However, since overfitting is unlikely to occur with random solutions, the rationale of using pretest 2 is unclear and it will not be further considered in this study. A more direct and effective way to evaluate the effect of the validation stage is simply to compare regular GP with and without validation[8].

Pretest 3: GP Versus Equal Search Intensity Random Search Without Training and Without Validation Stage. In pretest 3, selection of the strategies on the training set is removed: a large number random strategies are created and applied directly out-of-sample. The performance is evaluated as the average performance (e.g. average total return) over the set of random strategies. Comparing the outcome of pretest 3 with regards to pretest 1 and regular GP tells us something about how effective is the selection process, the extent to which a top performing rule on the training and validation sets will keep on performing well out-of-sample. If strategies selected by GP or random search on the training and validation intervals have some predictive ability out-of-sample, it provides use with some evidence that there is something to learn from the past. It is worth to point out that the randomness of the strategies is here constrained by the GP language: rules can only be made with GP functions/terminals organized according to the typing scheme. For instance, it is possible that the GP

[7] In [5], numerous experiments have highlighted that when training and out-of-sample data sets are very "dissimilar", for instance if the market exhibits an opposite trend, then there is little chance that GP performs well out-of-sample.

[8] For instance, as it is done in [5].

language is not expressive enough to represent a rule consisting in buying and selling every other period[9]. In the rest of this study, we will consider pretest 4, presented in Section 2.2, that is similar in spirit to prestest 3, but is more random in the sense that it does not possess the bias in randomness induced by the GP language.

2.2 GP Versus Lottery Trading

We call *lottery trading* a strategy that would consist in making the investment decision at random on the basis of the outcome of a random variable. In its simplest form, the random variable would follow a Bernoulli distribution where the parameter p expresses the probability to take a long position and $1-p$ the probability to be out of the market.

In our context, this requires refinement since we are interested in profitability and profitability takes into account transaction costs. So, to allow a fair comparison with GP, we should make sure that the expected number of transactions for lottery trading is the same as for GP. We call the expected number of transactions per unit of time the *frequency of a trading strategy.* Another important characteristic of a trading strategy is what we term its *intensity, i.e.* the number of periods where a position[10] "in the market" is held, over the length of the trading interval. We should also enforce lottery trading to have the same expected intensity as GP to avoid misleading results, for instance, the case where, given its frequency, the intensity of lottery trading is not sufficient to cover the transaction costs with the volatility of the market under study.

One denotes by F_{GP} and I_{GP} respectively the average frequency and average intensity observed for the set of GP evolved rules applied on the testing interval over all GP runs, N_{GP} is the number of transactions leading to F_{GP}. For the experiments made in the following, a sequence of investment decisions S_{LT} resulting from lottery trading is generated at random according to the following procedure:

- the intensity for lottery trading, I_{LT} , is uniformly chosen in $[I_{GP} \cdot (1-\alpha), \min(1, I_{GP} \cdot (1+\alpha))]$ with $0 \leq \alpha \leq 1$. In a first step, S_{LT} is made of the '0' positions (*i.e.* out of the market) followed by the block of '1' positions (*i.e.* in the market) corresponding to I_{LT},
- the number of transactions N_{LT} is uniformally chosen in the set of integer values that are even[11] in interval $[N_{GP} \cdot (1-\alpha), N_{GP} \cdot (1+\alpha)]$. The block of

[9] Period refers to the granularity of time used for trading, for instance, one second or one day.

[10] Implicitly, we consider here the trading of a single instrument, e.g. an index, where 2 decisions are possible at each time period: be in or be out of the market without short selling, or with short selling as implemented in [5], hold a long position or a short position. The concept remains valid where one can be holding a long position, a short position or be out of the market. One can also define the intensity and the frequency of a strategy for each instrument traded.

[11] N_{LT} has to be even since a "buy" transaction is followed by a sell transaction and no positions are left open.

'1' is subdivided at random in $N_{LT}/2$ sub-sequences and each sub-sequence is inserted at random inside the block of '0'. This design avoids the problem of overlapping '1' sub-sequences that occurs with other schemes.

We formulate the pretest comparing GP and lottery trading and denote by $\mathcal{H}_{4,0}$ the null hypothesis that GP does not outperform lottery trading.

Pretest 4: GP Versus Lottery Trading. Obviously, if GP is not able to outperform lottery trading, it gives strong evidence that GP will not be good at evolving effective trading strategies with the data at hand. In section 3, we shall discuss this point in more details.

3 What Does Pretest Tell Us ?

The outcomes of the pretests provide us with answers to the two following questions: is there something essential to learn on the training data that can be of interest for the out-of-sample period ? Does the GP implementation shows some evidence of effectiveness in that task ? Clearly, before actually trading with GP evolved programs, these two questions must be answered with reasonable certainty; the rest of this section explains how pretests may help in that regard.

3.1 Question 1: Is There Something to Learn ?

Null hypothesis $\mathcal{H}_{4,0}$ corresponding to pretest 4 has been presented in Section 2.2. We introduce pretest 5 that will be used in conjunction with pretest 4.

Pretest 5: Equivalent Intensity Random Search with Training and Validation Versus Lottery Trading. Here, we compare lottery trading to a random search with training and validation, and a search intensity equivalent to the one used for GP in pretest 4. Null hypothesis $\mathcal{H}_{5,0}$ is that the equivalent intensity random does not outperform lottery trading on the out-of-sample data. Depending on the validity of $\mathcal{H}_{4,0}$ and $\mathcal{H}_{5,0}$, we can draw the conclusions that are summarized in Table 1.

In case 1, best solutions on the training intervals, obtained with 2 different search algorithms, do not perform better than lottery trading on the out-of-sample period. This suggest to us than there is nothing to learn. In case 2, GP

Table 1. Information drawn from the outcomes of pretest 4 and pretest 5 ($\neg\mathbf{R}$ means that the null hypothesis $\mathcal{H}_{i,0}$ cannot rejected while $\mathbf{R}$ means that the hypothesis is rejected in favour of the alternative hypothesis)

	$\mathcal{H}_{4,0}$	$\mathcal{H}_{5,0}$	**Interpretation**
case 1	$\neg\mathbf{R}$	$\neg\mathbf{R}$	there is evidence that there is nothing to learn
case 2	$\mathbf{R}$	$\neg\mathbf{R}$	there may be something to learn (weak certainty)
case 3	$\mathbf{R}$	$\mathbf{R}$	there is evidence that there is something to learn
case 4	$\neg\mathbf{R}$	$\mathbf{R}$	there may be something to learn (weak certainty)

outperforms lottery trading but random search does not; it is possible that there is something to learn but that the selected random rules do not have a sufficient predictive ability. Anyway, this lead us to a less certain conclusion than in case 3 where both search techniques outperform lottery trading. Finally case 4 is a special case where random search performs better than lottery trading but GP does not. The whole evolution process of GP has thus a detrimental effect and a possible explanation is that GP induced solutions overfit the training data.

3.2 Question 2: Is the GP Machinery Working Properly?

The second question we ought to ask is whether GP is effective. Of course, this cannot be answered with the data at hand if pretests 4 and 5 have shown that there is nothing to be learned (case 1 in Table 1). In addition, in case 4 of Table 1, we already know that GP is not efficient since, by transitivity, it is outperformed by the random search based algorithm. Thus, the only two cases where one really needs to proceed to further examination are case 2 and case 3. The validity of null hypothesis $\mathcal{H}_{1,0}$, which can be tested with pretest 1, gives a helpful insight into the answer: only if $\mathcal{H}_{1,0}$ should be rejected we can conclude that GP shows some real effectiveness. We would like to stress that rejecting $\mathcal{H}_{1,0}$ is far from implying profitability, but beating a mere random search algorithm on a difficult problem with an infinite search space, is the bare minimum one can expect from GP.

4 Experiments

The aim of the experiments is to evaluate the extent to which the pretests proposed are reliable. The methodology adopted here is to check if the outcomes of the pretests are consistent with results already published in the literature. We call GP1 the GP implementation developed for this study and GP2 the software[12] used in [5], which will constitute our benchmark. The GP control parameters, identical to the one used in [5], are summarized in Table 1 (Appendix A).

The stock indexes from 3 stock markets are used: TSE 300 (Canada), Nikkei Dow Jones (Japan) and Capitalization Weighted Stock Index (Taiwan). They have been chosen among the 8 markets studied in [5] because they exhibit the main evolution patterns that can be found in the set of 8 markets. The aim of GP is to induce the most profitable strategy, measured by the accumulated return, for trading the stock exchange index. The use of short selling is possible. We adopt what is done classically in literature in terms of data-preprocessing and use normalized data that is obtained by dividing each day's price by a

[12] Although both programs have been developed by members of the AI-ECON Research Center, they have not been written by the same persons and do not share a single line of code. Furthermore GP2, which is based on the *Open-Beagle* library (see `http://beagle.gel.ulaval.ca/`), implements strongly-typed GP.

250-day moving average[13]. In a way similar to what is done usually, we subdivide the whole dataset into three sections: *training*, *validation* and *out-of-sample* test period. For each considered stock index, 3 different out-of-sample test periods of 2 years (*i.e.* 1999-2000, 2001-2002, 2003-2004) follow a 3 year validation and a 3 year training period. In the following, the term market refers to a stock exchange during a specific out-of-sample period. For instance, Canada-1 (C1 for short) is the market corresponding the TSE 300 during the out-of-sample period 1999-2000. Hypothesis testing is performed with the *Student's t-test* at a 95% confidence level. The samples for statistics are constituted of the results of 50 GP runs, 50 runs of equivalent search intensity random search with training and validation (ERS) and 100 runs of lottery trading (LT).

In 4 out of the 9 markets (*i.e.* C3, J2, T1, T3), there is evidence that there is something to learn from the training data (case 1 in Table 1). This is consistent with [5] where GP2 performs outstandingly on these 4 markets (respective total return: 0.34, 0.17, 0.52, 0.27). On markets C1, J3 and T2, pretests 4 and 5 suggest to us that there is nothing to learn (case 3). Except for C1, GP2 also performs poorly (-0.18 for J3 and -0.05 for T2). Finally, in the 3 markets where GP1 is shown to beat ERS ($\mathcal{H}_{1,0}$ is rejected in favor of $\mathcal{H}_{1,1}$ for J1, J2 and T1), GP results are very good : both GP1 and GP2 produce positive returns and outperform the buy-and-hold strategy.

Although more comprehensive tests are to be performed, the experiments conducted here show some preliminary evidence that the proposed pretests possess some predictive ability. Indeed, when the outcome is "nothing to learn", GP performs very poorly (except in one case out of three). When pretests suggest that there is something to learn, at least one implementation did good and when GP1 is more efficient than random search (*i.e.* ERS), GP2 from [5] is efficient too. In the light of the pretests, we should also conclude if our GP implementation (*i.e.* GP1) is more efficient than ERS, it is only slightly more efficient since one would expect more cases where GP beats LT and not LT. This suggest that GP1 is only able to take advantage of "simple" regularities in the data.

5 Conclusions

The main purpose of this paper is to enrich the earlier research on Genetic Programming (GP) induced market-timing decisions by proposing pretests aiming to shed light on the GP results. Actually in the literature, the results of applying GP for market-timing decisions are typically not very convincing but the investigators always suggest the possibility of further improvements. If the investigators can first convince that there is something to learn and that GP is suitable for that task, then their conclusion would be less vague and uncertain. We propose here a series of pretests, where GP is tested against a random behavior (*lottery trading*) and against strategies created at random (*zero-intelligence*

[13] See [5] for a discussion about how non-normalized data affects the performance of GP.

strategies), that aim to answer these two crucial questions. Of course there is the risk of getting a wrong pretest result and the possible reasons why GP may have failed should be thoroughly investigated before drawing conclusion. But, at the end, analyzing the results in the light of the pretests should help to draw more fine-grained conclusion.

Acknowledgment. This research was conducted when the second author (Nicolas Navet) was visiting researcher at the AI-ECON Research Center, National Chengchi University (NCCU), Taipei, Taiwan. The financial support from the AI-ECON Research Center as well as NCCU is greatly acknowledged. The authors would like also to acknowledge the grant from National Science Concil #95-2415-H-004-002-MY3.

References

1. J.A. Giles and D.E.A. Giles: Pre-test Estimation and Testing in Econometrics: Recent Developments. Journal of Economic Surveys **7**(2) (1993) 145–97
2. D. Danilov and J.R. Magnus: Forecast Accuracy After Pretesting with an Application to the Stock Market. Journal of Forecasting **23** (2004) 251–274
3. R. Sullivan and A. Timmermann and H. White: Data-Snooping, Technical Trading Rule Performance, and the Bootstrap. Journal of Finance **54** (1999) 1647–1692
4. M.A. Kaboudan: A Measure of Time Serie's Predictability Using Genetic Programming Applied to Stock Returns. Journal of Forecasting **18** (1999) 345–357
5. S.-H. Chen and T.-W. Kuo and K.-M. Hoi: Genetic Programming and Financial Trading: How Much about "What we Know". In: Handbook of Financial Engineering. Kluwer Academic Publishers (2006) Forthcoming.
6. S.-H. Chen and T.-Z. Kuo: Overfitting or Poor Learning: A Critique of Current Financial Applications of GP. In Springer-Verlag, ed.: Proceedings of the Sixth European Conference on Genetic Programming (EuroGP-2003). Number 2610 in LNCS (2003) 34–46
7. F. Allen and R. Karjalainen: Using Genetic Algorithms to Find Technical trading rules. Journal of Financial Economics **51** (1999) 245–271
8. C. Neely and P. Weller and R. Dittmar: Is Technical Analysis in the Foreign Exchange Market Profitable? A Genetic Programming Approach. Journal of Financial and Quantitative Analysis **32**(4) (1997) 405–427

A Genetic Programming Settings

Program GP2 implements strongly typed GP with the set of functions and terminals described in Table 1. The parameters here are basically identical to the ones in [5] (program GP1) except when fine-tuning GP2 have highlighted that better results may be obtained with different parameters. Precisely, we make use of more elitism, the size of the tournament selection is set to 5 and numerical mutation is implemented.

Population size	500
Number of generations	100
Maxim tree depth	10
Function set	+,-,*,/,norm,average,max,min,lag,and, or, not, >,<,if-then-else,true,false
Terminal set	price, real and integer ephemeral constants
Value range for real constants	[-1,1]
Value range for integer constants	[0,1000]
Offsprings created by:	
crossover	50%
standard mutation	20%
swap mutation	15%
reproduction	10%
ephemeral constant mutation	5%
Initialization	ramp-half-and-half
Evolution scheme	generation-by-generation replacement strategy
Elitism	25 best individuals kept for next generation
Selection scheme	tournament selection of size 5
Fitness function	accumulated return
Transaction costs	0.5%
Validation	
number of best trees saved	1 individual per run is saved for validation

Fig. 1. GP control parameters