Overfitting or Poor Learning: A Critique of Current Financial Applications of GP

Shu-Heng Chen and Tzu-Wen Kuo

AI-ECON Research Center, Department of Economics
National Chengchi University, Taipei, Taiwan 11623
chchen@nccu.edu.tw, kuo@aiecon.org

Abstract. Motivated by a measure of predictability, this paper uses the extracted signal ratio as a measure of the degree of overfitting. With this measure, we examine the performance of one type of overfitting-avoidance design frequently used in financial applications of GP. Based on the simulation results run with the software Simple GP, we find that this design is not effective in avoiding overfitting. Furthermore, within the range of search intensity typically considered by these applications, we find that underfitting, instead of overfitting, is the more prevalent problem. This problem becomes more serious when the data is generated by a process that has a high degree of algorithmic complexity. This paper, therefore, casts doubt on the conclusions made by those early applications regarding the poor performance of GP, and recommends that changes be made to ensure progress.

1 Motivation and Introduction

Overfitting is one of the most intensively addressed issues in data mining ([2], [6], [8], [9], [14]). A great many techniques have been developed over the past decade, and some of these techniques have also been applied to the financial applications of genetic programming ([4]). For example, the use of *cross-validation* now seems to have become a standard procedure followed by many financial GP users. This has become particularly so following a series of prestigious journal publications that have recommended this way of preventing overfitting ([1], [10], [11], and [16]). By this procedure, one run of GP uses two time periods. The first period is called the *training period* and is used to train the genetic programs. The second period is referred to as the *selection period*, which is used to select the best performing programs and decide when to stop the training. For example, in [10], the termination criterion is achieved if no new best rule appears for 25 generations.

As opposed to the problem of overfitting, *poor learning* (*underfitting*) has received much less attention. The asymmetry seems to be natural. Given the rich expressive power of the heavy data-mining machinery, people tend to believe that they may easily abuse the power if caution is not well taken. Validation is a classical tool for resolving this issue and has also become one of the common overfitting-avoidance designs ([15]). This validation scheme has been used as if it will make learning stop at a right moment, no more (overfitting) and no less (underfitting). However, *is that really so*? It may not surprise us if this scheme fails to avoid overfitting, as already suggested by some empirical studies ([12]). Nonetheless, it may surprise us if we are told that this design can also result in underfitting.

In this paper, we shall show that underfitting can occur mainly due to our ignorance of the extremely large search space extended by the primitives of GP. Coming with this ignorance is the causal design of search intensity, e.g., a causal combination of the population size and the number of generations. Little attention has been paid to whether the resultant search intensity is good enough. Under such circumstances, worrying about abusing the expressive power of GP may be a little far-fetched, and adding a validation scheme may make things even worse.

To show this, we introduce a technical notion of overfitting based on the *signal ratio*. Then, by artificially generating time-series data with different signal ratios and different *algorithmic complexity* (*node complexity*), we test how likely it is that a standard GP may overfit the data. We first start the experiments (Experiment Series 1) without imposing any overfitting-avoidance design, and then examine how serious is the overfitting problem that may arise. In this series of experiments, we find that even with a moderate degree of search intensity, overfitting is in effect not a serious problem. On the contrary, for most of the cases, the real concern is underfitting. In another series of experiments (Experiment Series 3), we heighten the search intensity by doubling the population size. As one may expect, this will result in the problem of overfitting becoming more likely to appear. That is true. Nevertheless, our results show that what bothers us more is still the problem of underfitting. Given this situation, introducing an overfitting-avoidance design can do more harm than good. The last point is exactly what we show in the other two series of experiments (Experiment Series 2 and 4).

The rest of the paper is organized as follows. Section 2 proposes a technical and practical notion of overfitting. Section 3 gives the details of the experimental designs. Section 4 presents the simulation results with accompanying discussions, and is followed by the concluding remarks in Section 5.

2 Signal Ratio and Overfitting

Motivated by [7], we apply the *signal ratio* as a measure of overfitting. The idea is straightforward. Consider a series $\{y_t\}$. Its signal-noise orthogonal decomposition can be written as

$$y_t = x_t + \epsilon_t, \tag{1}$$

where the *signal* x_t follows a deterministic process, whereas the noise ϵ_t is an identically independently distributed process with a mean zero and a variance σ_ϵ^2. The variance decomposition of y_t can then be written as follows.

$$\sigma_y^2 = \sigma_x^2 + \sigma_\epsilon^2, \tag{2}$$

where σ_y^2 is the variance of y_t. However, rigorously speaking, σ_x^2 is not the variance of x_t, since by the signal we mean that x_t is not random but deterministic. Therefore, σ_x^2 should be interpreted as the *fluctuation* of an ensemble of the deterministic process of $\{x_t\}$ [1]. With the decomposition (2), the *signal ratio*, denoted by θ, is defined as

$$\theta = \frac{\sigma_x^2}{\sigma_y^2} = 1 - \frac{\sigma_\epsilon^2}{\sigma_y^2}. \tag{3}$$

With this definition of the signal ratio, the problem of *overfitting* can then be stated as follows. Let $\hat{y}_t$ be the signal extracted by a machine learning tool, say, GP. Then we say that the problem of overfitting is detected if

$$\hat{\theta} = \frac{\sigma_{\hat{y}}^2}{\sigma_y^2} > \theta. \tag{4}$$

In other words, overfitting is detected if the information (signal) extracted by GP is even greater than the maximum information which we actually have from y_t. When Equation (4) is satisfied, this means that GP has learned more than it should and, as a result, has started to mistake noise as a signal. Similarly, if the inequality is turned the other way around, i.e. $\hat{\theta} < \theta$, then we say that GP has not learned enough so that some information is left unexploited. Contrary to the case of overfitting, here we encounter the problem of *underfitting.*

3 Experimental Designs

3.1 Data

Based on the notion of overfitting discussed in Section 2, we propose the following data-generation mechanism. First, we start with the generation process for signals, $\{x_t\}$. [3] examined the predictability of the chaotic time series with GP. The three chaotic time series considered by them are

$$x_t = f(x_{t-1}) = 4x_{t-1}(1 - x_{t-1}), \quad x_0 \in [0, 1], \tag{5}$$

$$x_t = f(x_{t-1}) = 4x_{t-1}^3 - 3x_{t-1}, \quad x_0 \in [-1, 1], \tag{6}$$

and

$$x_t = f(x_{t-1}) = 8x_{t-1}^4 - 8x_{t-1}^2 + 1, \quad x_0 \in [-1, 1]. \tag{7}$$

We choose these three chaotic time series as the signal generation processes. Since neither can we control the signal ratio nor can we know the true signal ratio in real financial time series, the test proposed in this paper will be conducted by using only these deterministic chaotic series. Certainly, as the title of the paper suggests, to be a genuine critique of current financial applications, we shall extend our tests to real financial data in the future.

[1] We shall be more specific on this when we come to the design of the experiments.

These three laws of motion are different in terms of their *algorithmic size*, i.e. the *length* of their symbolic expression. To see this, we rewrite each of the equations above into the corresponding LISP S-expression:

$$(\; * \; (\; 4 \; * \; (\; x_t \; (\; - \; 1 \; x_t \;) \;) \;) \;) \tag{8}$$

$$(\; - \; (\; * \; 4 \; (\; * \; x_t \; (\; * \; x_t \; x_t \;) \;) \;) \; (\; * \; 3 \; x_t \;) \;) \tag{9}$$

$$\begin{aligned}&(\; + \; (\; - \; (\; * \; 8 \; (\; * \; x_t \; (\; * \; x_t \; (\; * \; x_t \; x_t \;) \;) \;) \;) \\ &(\; * \; 8 \; (\; * \; x_t \; x_t \;) \;) \;) \; 1 \;)\end{aligned} \tag{10}$$

The *length* of a LISP S–expression is determined by counting from the leftmost to the rightmost position the number of elements (atoms) in the string that makes up the S–expression. From Eqs. (8) to (10), the lengths of the LISP S-expression are 7, 11, and 17, respectively. Therefore, in terms of algorithmic complexity, Eq. (5) is the simplest, while Eq. (7) is the most complex expression.

In addition to the signal, an *i.i.d.* noise is added to obscure the observation series $\{y_t\}$.

$$y_t = x_t + \epsilon_t = f(x_{t-1}) + \epsilon_t. \tag{11}$$

GP is applied to fit the hidden relationship between y_t and x_{t-1}, and the predicted value of y_t is

$$\hat{y}_t = \hat{f}(x_{t-1}). \tag{12}$$

Based on Equation (3), different signal ratios can be derived by manipulating σ_ϵ^2. In this study, we consider five values of θ, ranging from a very high noise ratio $\theta = 0.05$, to 0.15, 0.25, and 0.35, and to a high signal ratio of 0.5. Therefore, a total of 15 different time series of $\{y_t\}$ are generated from the signal series depicted in Eqs. (5) to (7) with five different values of θ.

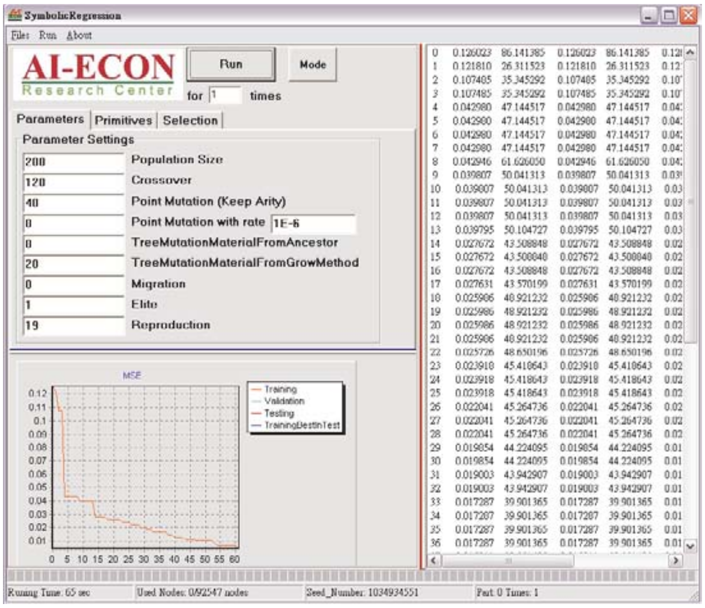
3.2 Genetic Programming

The genetic programming software used in this paper is **Simple GP**, developed by the AI-ECON Research Center[2]. All of the main GP control parameters can be set by directly inputting the values into the main menu, shown in Figure 1[3].

The validation scheme, used in [1], [10], [11], and [16], is incorporated into **Simple GP**. This can be seen from Figure 2. In the top half of the menu, there is a place for users to indicate whether they wish to impose the validation scheme as an early termination criterion. This can be done by simply answering how many pieces of the data are to be divided (the first box, Figure 2) and by showing how these pieces of data are to be distributed to be used for training, validating, and testing. In the example in Figure 2, [3]-(1,1,1) states that the data are to be divided into three equal parts, and that the first part will be used

[2] Some detailed descriptions of it can be found in [5].

[3] The purpose of indicating this software is mainly for those interested readers who would like to replicate or validate the results obtained in this paper.



Parameter Settings

200	Population Size
120	Crossover
40	Point Mutation (Keep Arity)
0	Point Mutation with rate 1E-6
0	TreeMutationMaterialFromAncestor
20	TreeMutationMaterialFromGrowMethod
0	Migration
1	Elite
19	Reproduction

```
0   0.126023  86.141385  0.126023  86.141385  0.12
1   0.121810  26.311523  0.121810  26.311523  0.12
2   0.107485  35.345292  0.107485  35.345292  0.10
3   0.107485  35.345292  0.107485  35.345292  0.10
4   0.042980  47.144517  0.042980  47.144517  0.04
5   0.042980  47.144517  0.042980  47.144517  0.04
6   0.042980  47.144517  0.042980  47.144517  0.04
7   0.042980  47.144517  0.042980  47.144517  0.04
8   0.042946  61.626050  0.042946  61.626050  0.04
9   0.039807  50.041313  0.039807  50.041313  0.03
10  0.039807  50.041313  0.039807  50.041313  0.03
11  0.039807  50.041313  0.039807  50.041313  0.03
12  0.039807  50.041313  0.039807  50.041313  0.03
13  0.039795  50.104727  0.039795  50.104727  0.03
14  0.027672  43.508848  0.027672  43.508848  0.02
15  0.027672  43.508848  0.027672  43.508848  0.02
16  0.027672  43.508848  0.027672  43.508848  0.02
17  0.027631  43.570199  0.027631  43.570199  0.02
18  0.025986  48.921232  0.025986  48.921232  0.02
19  0.025986  48.921232  0.025986  48.921232  0.02
20  0.025986  48.921232  0.025986  48.921232  0.02
21  0.025986  48.921232  0.025986  48.921232  0.02
22  0.025726  48.650196  0.025726  48.650196  0.02
23  0.023918  45.418643  0.023918  45.418643  0.02
24  0.023918  45.418643  0.023918  45.418643  0.02
25  0.023918  45.418643  0.023918  45.418643  0.02
26  0.022041  45.264736  0.022041  45.264736  0.02
27  0.022041  45.264736  0.022041  45.264736  0.02
28  0.022041  45.264736  0.022041  45.264736  0.02
29  0.019854  44.224095  0.019854  44.224095  0.01
30  0.019854  44.224095  0.019854  44.224095  0.01
31  0.019003  43.942907  0.019003  43.942907  0.01
32  0.019003  43.942907  0.019003  43.942907  0.01
33  0.017287  39.901365  0.017287  39.901365  0.01
34  0.017287  39.901365  0.017287  39.901365  0.01
35  0.017287  39.901365  0.017287  39.901365  0.01
36  0.017287  39.901365  0.017287  39.901365  0.01
```

Running Time: 65 sec | Used Nodes: 0/92547 nodes | Seed_Number: 1034934551 | Part: 0 Times: 1

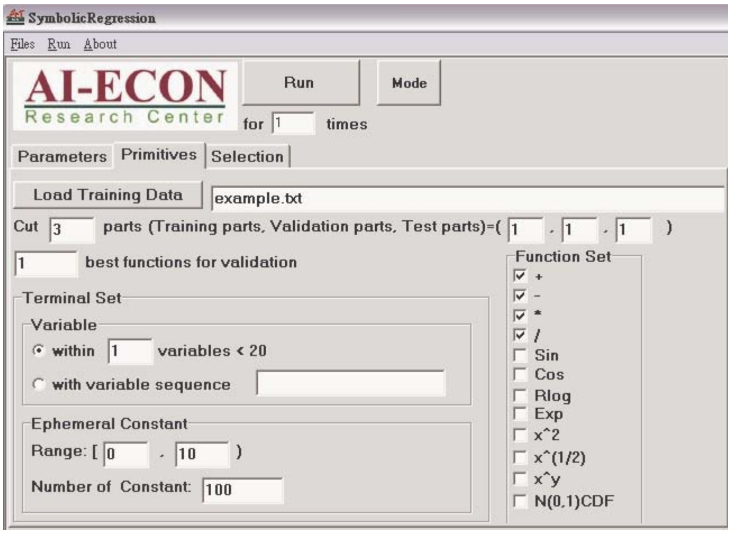
Fig. 1. Simple GP: Main Menu 1

for training, the second for validation, and the third for testing. By inserting a zero number into the validation place, one can in effect turn off the validation scheme. For example, [1]-(1,0,0) means that there is only one training set. This specification is exactly what we were faced with when doing the first and the third series of experiments, while for the second and fourth series of experiments the validation scheme is added.

The steps involved in **Simple GP** are detailed as follows:

1. Create an initial generation of *Pop* random forecasting models using the *ramp-half-and-half* method.
2. Measure the fitness of each model over the *training set* and rank according to fitness.
3. Select the top-ranked k models and calculate its fitness over the validation set. Save it as the initial best k models.
4. Implement all genetic operators in the standard way, and a new generation of models will be born.
5. Measure the fitness of each model in the new generation over the training set. The best k models are selected, and their fitness over the *validation set* is calculated and compared with that of the previously saved best k models. Then a $(k + k)$-selection scheme is applied to select the best k models, and they are saved as the *new* best k models.
6. Stop if none of these best k models are replaced for g generations, or after *Gen* generations. Otherwise, return to step 4.

Step 5 is the validation scheme used in [1], [10], [11], and [16].



SymbolicRegression

Files Run About

AI-ECON Research Center

Run Mode

for 1 times

Parameters Primitives Selection

Load Training Data example.txt

Cut 3 parts (Training parts, Validation parts, Test parts)=(1 , 1 , 1)

1 best functions for validation

Terminal Set

Variable

within 1 variables < 20

with variable sequence

Ephemeral Constant

Range: [0 , 10)

Number of Constant: 100

Function Set

✓ +

✓ -

✓ *

✓ /

Sin

Cos

Rlog

Exp

x^2

x^(1/2)

x^y

N(0,1)CDF

Fig. 2. Simple GP: Main Menu 2

Four series of experiments were conducted. Experiments 1 and 3 did not include the validation scheme. They differed in search intensity characterized by population size. The one set in Experiment Series 1 was 200, and the one set in Experiment Series 3 was 400. Validation was involved in Experiments 2 and 4. In each series of experiments, a hundred runs of GP were implemented for each chaotic time series with each of the five signal ratios. In other words, there were in total 1,500 (=15 $\times$ 100) independent runs conducted in each series of experiments. The values of the control parameters are summarized in Table 1.

4 Results

One interesting thing to observe is whether overfitting can easily appear when validation is not imposed (Experiment Series 1). To see this we average the $\hat{\theta}$ obtained from 100 runs, called $\bar{\theta}$. They are reported in the first block of Table 2. Since we have 100 runs (a large sample), $\bar{\theta}$ can be regarded as a reasonable estimate of the corresponding *population* "signal ratio", extracted by GP, referred to as θ^*. Furthermore, by applying the central limit theorem, one can decide whether there is either overfitting or underfitting by testing the following null hypothesis:

$$H_0 : \theta^* = \theta. \tag{13}$$

GP correctly extracts what it is supposed to extract if the null fails to be rejected; otherwise overfitting is found if $\theta^* > \theta$ significantly or underfitting is found if $\theta^* < \theta$ significantly. In Table 2, immediately below $\bar{\theta}$, is the test statistic, which is followed by the *p-value*.

Table 1. Control Parameters of GP

Population size (*Pop*)	200 (Experiment 1, 2), 400 (Experiment 3, 4)
Offspring trees created	
by crossover	60%
by point mutation	20%
by tree mutation (grow method)	10%
by elite	0.5% (Exp. 1, 2), 0.25% (Exp. 3, 4)
by reproduction	9.5% (Exp. 1, 2), 9.75% (Exp. 3, 4)
Function set	$+, -, \times, \div$
Constant	100 numbers in [0, 10)
Replacement scheme	Tournament selection (size=2)
Stop criterion	
number of generations (*Gen*)	300
or MSE less than	0.000001
Validation (Experiment 3 only)	
number of best models saved (k)	1
number of quiet generations (g)	150

From these results, we can see that the null is rejected in 13 out of the 15 designs. Of the 13 cases, only one design (the design with Eq. 5 and $\theta = 0.05$) is proved to be an example of overfitting, and all others belong to the case of underfitting. It is then obvious that *overfitting turns out not to be a serious issue even though the validation design is not imposed.* Here, what concerns us more is just the opposite, i.e. the issue of underfitting. This result is particularly striking since the search intensity ($Pop = 200, Gen = 300$) is by no means low as opposed to most financial applications of GP[4].

In this case, if one just follows convention ([1], [10], [11], and [16]) and adds a validation step (Experiment Series 2), what will happen? Basically, nothing changes, and we benefit little from this setting. First, the validation step is an overfitting-avoidance design, and is therefore not supposed to solve the problem of underfitting. The second block in Table 2 confirms this: those 13 designs that were shown to be underfitting in Experiment 1, remain the same in Experiment 2. Second, as for the design in which overfitting is detected, imposing validation does not help either: it remains overfitting, while the over-extracted signal ratio declines from 0.0849 to 0.0808.

The evidence presented so far raises two questions. First, when shall we expect the problem of overfitting to appear? Second, is validation an effective overfitting-avoidance design? The next two series of experiments are designed to address these two questions. In Experiment 3, search intensity was enhanced by increasing the population size from 200 to 400 (Table 1). Would this more intensive search lead to a higher risk of overfitting? The third block in Table 2 seems to suggest so. Out of the 15 designs, which have the same settings

[4] For example, the population size set in [10], [11] and [16] is just 100, and the number of generations is also just 100.

Table 2. Simulation Results

Exp \ θ			0.05	0.15	0.25	0.35	0.5
1	eq1	$\bar{\theta}$	0.0849	0.1413	0.2560	0.3091	0.4191
		z	11.9532	-1.5340	0.7095	-3.5711	-5.0880
		p	0.0000	0.0625	0.2390	0.0002	0.0000
	eq2	$\bar{\theta}$	0.0352	0.1386	0.2323	0.3029	0.3998
		z	-12.5330	-3.0916	-2.9883	-5.3213	-9.6061
		p	0.0000	0.0010	0.0014	0.0000	0.0000
	eq3	$\bar{\theta}$	0.0438	0.0868	0.1760	0.1850	0.2947
		z	-3.5610	-15.2165	-11.3537	-20.4669	-18.8052
		p	0.0002	0.0000	0.0000	0.0000	0.0000
2	eq1	$\bar{\theta}$	0.0808	0.1411	0.2450	0.2893	0.3821
		z	10.1044	-1.5189	-0.4152	-4.9791	-5.7578
		p	0.0000	0.0644	0.3390	0.0000	0.0000
	eq2	$\bar{\theta}$	0.0301	0.1348	0.2298	0.3119	0.4156
		z	-16.4007	-4.0032	-3.5421	-4.5439	-8.7157
		p	0.0000	0.0000	0.0002	0.0000	0.0000
	eq3	$\bar{\theta}$	0.0425	0.0853	0.1758	0.1988	0.2866
		z	-4.1114	-15.5164	-12.7027	-16.9850	-18.7969
		p	0.0000	0.0000	0.0002	0.0000	0.0000
3	eq1	$\bar{\theta}$	0.0944	0.1587	0.2829	0.3332	0.4801
		z	20.0965	2.3321	6.4594	-2.2650	-1.5415
		p	0.0000	0.0098	0.0000	0.0118	0.0616
	eq2	$\bar{\theta}$	0.0407	0.1476	0.2559	0.3458	0.4317
		z	-7.7766	-0.7945	1.2601	-0.6809	-7.5529
		p	0.0000	0.2134	0.1038	0.2480	0.0000
	eq3	$\bar{\theta}$	0.0520	0.1056	0.1843	0.2346	0.3398
		z	1.2838	-10.8656	-10.1204	-14.3101	-14.1068
		p	0.0996	0.0000	0.0002	0.0000	0.0000
4	eq1	$\bar{\theta}$	0.0902	0.1588	0.2758	0.3219	0.4842
		z	19.8840	2.2404	4.8728	-3.0996	-1.1045
		p	0.0000	0.0125	0.0000	0.0010	0.1347
	eq2	$\bar{\theta}$	0.0345	0.1546	0.2514	0.3388	0.4471
		z	-11.2780	1.9955	0.2991	-1.6719	-6.1439
		p	0.0000	0.0230	0.1912	0.0473	0.0000
	eq3	$\bar{\theta}$	0.0477	0.1053	0.1836	0.2250	0.3267
		z	-1.2219	-11.0175	-10.0111	-15.6075	-14.7593
		p	0.1109	0.0000	0.0000	0.0000	0.0000

$\bar{\theta}$ denotes the sample average of $\hat{\theta}$ (extracted signal ratio) over 100 runs. z refers to the test statistic derived from the central limit theorem. p is the p-value of the associated test statistic that is more extreme in the direction of the alternative hypothesis ($\theta^* \neq \theta$) when H_0 is true.

as those in Experiment 1 except for population size, the number of overfitting cases increases from one to three. The evidence based on the designs that use Equation (5) and where $\theta = 0.05$, 0.15 and 0.25 is now in favor of overfitting. Equally interesting is that the null of just fitting (Equation 13) has now failed to be rejected in five of the designs, a jump from the original two.

Nevertheless, what seems to be most disappointing is that the majority of them (7 out of 15) still exhibit the feature of underfitting. If we look at the distribution of these seven designs, most of them, four out of the seven, come from the one using Equation (7), which is the one with the highest degree of algorithmic complexity (node complexity) ([3]). This finding is very plausible. As analyzed by [5], laws of motion with a higher degree of algorithmic complexity generally require more intensive searches than those with less complexity. As a result, such a design is less likely to abuse the power of curve fitting when the data is generated by a process with a high degree of algorithmic complexity.

Since the risk of overfitting increases with search intensity, it is crucial to know how effectively an overfitting design can ameliorate the situation. Therefore, in Experiment 4, we add back the validation design as we did in Experiment 2, and test its performance. The results are shown in the last block in Table 2. Our attention is particularly drawn to the design where the problem of overfitting is found, i.e. the three designs mentioned before. Unfortunately, from Table 2, we can see that the problem of overfitting remains in all of these three designs. In two of the three cases, we do see the decrease in the extracted signal ratio, one in its decreasing from 0.0944 to 0.0902, and the other in its falling from 0.2829 to 0.2758. However, these changes are not enough given that the true signal ratios θ are just only 0.05 and 0.25, respectively. Furthermore, it is interesting to notice that in one case the problem even gets worse, i.e. the design using Equation (6) where $\theta = 0.15$. In that case, originally, GP extracted a signal ratio with $\bar{\theta} = 0.1476$, which is significantly below the threshold $\theta = 0.15$. However, when the validation design is imposed, the extracted signal ratio goes significantly beyond 0.15 to 0.1546; and hence exhibits the feature of overfitting.

5 Two Further Tests

In addition to the test of the null (13), we also carried out two examinations pertaining to the effect of *search density* and *validation* upon the degree of fitting (the signal ratio extracted by GP). We first conducted a statistical test for the hypothesis that *intensifying search shall increase the degree of fitting.* Let θ_i^* be the population signal ratio extracted by GP under Experiment Series i ($i = 1, 2, 3, 4$). The effect of search density can be analyzed by testing the null

$$\theta_1^* = \theta_3^*, \tag{14}$$

and

$$\theta_2^* = \theta_4^*, \tag{15}$$

separately. (14) is the null of equal fitting when validation is not imposed, whereas (15) is the null of equal fitting when validation is imposed. If these two null hypotheses fail to be rejected, then intensifying search intensity has little effect on the degree of fitting. As before, the large sample theorem is applied to these two tests. The test results of the null (14) and (15) are shown in the first and second block of Table 3 respectively. Form the corresponding Z statistics, the number of rejections is very high for both tests, if one takes the usual critical region $\mid Z \mid > Z_{0.025} = 1.96$.

In the first block of Table 3 , θ_3^* is bigger than θ_1^* in all of the fifteen cases, and the null (14) is rejected at a total of twelve times. In the second block of the Table, θ_4^* is also bigger than θ_2^* in all of the fifteen cases, and the null (15) has failed to be rejected only in one case. *As a result, it is quite evident that search intensity has a positive effect on the degree of fitting, and hence will contribute to a risk of overfitting.*

We then turn to see the effect of validation on the degree of overfitting. The two null hypotheses to test are

$$\theta_1^* = \theta_2^*, \tag{16}$$

and

$$\theta_3^* = \theta_4^*. \tag{17}$$

The test results are reported in the third and the fourth block of Table 3. For both tests, the null of equal fitting has failed to be rejected in only one out of the fifteen cases, if the same critical region is applied. Therefore, even though numerically θ_1 (θ_3) has a larger degree of fitting than θ_2 (θ_4), the difference is statistically negligible. *This finding may surprise many of us, who believe that validation shall help to avoid overfitting.*

One possible explanation for this seemingly surprising finding is as follows. When the algorithmic complexity associated with the problem is high, the desired termination point may be farther than what our designated search density may reach. In this case, the termination condition set by the validation design will rarely be met. Hence, using or not using the validation design would not make much difference. This explanation may apply to the cases of Equations (6) and (7) where the problem of underfitting is severe. However, it remains to be a puzzle why the design also failed to work for the case of Equation (5)[5].

6 Conclusion: Implications for Financial Applications

For most financial engineers, financial data are usually assumed to be *highly complex* (nonlinear) but also *informative.* The first assumption implies that the algorithmic complexity of the data generation process can be quite high, whereas the second assumption indicates a moderate level for the signal ratio. If these two properties are correct, then our simulation results seem to suggest that the problem of and the caution associated with overfitting may be exaggerated given the usual search intensity frequently set in many financial applications of GP. In this case, adding a validation design may actually make the hidden information in

[5] One conjecture is that the validation parameter g (see Table 1) may not be set appropriately. One of our next studies is to examine the role of this parameter.

Table 3. Significance of Search Intensity and Validation

Null \ θ			0.05	0.15	0.25	0.35	0.5
$\theta_1^* = \theta_3^*$	eq1	θ_1^*	0.0849	0.1413	0.2560	0.3091	0.4191
		θ_3^*	0.0944	0.1587	0.2829	0.3332	0.4801
		z	2.5983	2.5624	2.7387	1.7624	2.9742
	eq2	θ_1^*	0.0352	0.1386	0.2323	0.3029	0.3998
		θ_3^*	0.0407	0.1476	0.2559	0.3458	0.4317
		z	3.2509	1.8926	3.1265	3.9871	2.3133
	eq3	θ_1^*	0.0438	0.0868	0.1760	0.1850	0.2947
		θ_3^*	0.0520	0.1056	0.1843	0.2346	0.3398
		z	3.5101	3.2295	0.8971	4.3496	2.8683
$\theta_2^* = \theta_4^*$	eq1	θ_2^*	0.0808	0.1411	0.2450	0.2893	0.3821
		θ_4^*	0.0902	0.1588	0.2758	0.3219	0.4842
		z	2.5599	2.5072	2.3340	2.1508	4.0907
	eq2	θ_2^*	0.0301	0.1348	0.2298	0.3119	0.4156
		θ_4^*	0.0345	0.1546	0.2514	0.3388	0.4471
		z	2.4027	4.4574	2.9530	2.5051	2.4323
	eq3	θ_2^*	0.0425	0.0853	0.1758	0.1988	0.2866
		θ_4^*	0.0477	0.1053	0.1836	0.2250	0.3267
		z	2.0218	3.4492	0.8918	2.1906	2.4568
$\theta_1^* = \theta_2^*$	eq1	θ_1^*	0.0849	0.1413	0.2560	0.3091	0.4191
		θ_2^*	0.0808	0.1411	0.2450	0.2893	0.3821
		z	0.9647	0.0208	0.7456	1.1840	1.4263
	eq2	θ_1^*	0.0352	0.1386	0.2323	0.3029	0.3998
		θ_2^*	0.0301	0.1348	0.2298	0.3119	0.4156
		z	3.0306	0.7202	0.3008	-0.7408	-1.1133
	eq3	θ_1^*	0.0438	0.0868	0.1760	0.1850	0.2947
		θ_2^*	0.0425	0.0853	0.1758	0.1988	0.2866
		z	0.5065	0.2661	0.0311	-1.1474	0.5112
$\theta_3^* = \theta_4^*$	eq1	θ_3^*	0.0944	0.1587	0.2829	0.3332	0.4801
		θ_4^*	0.0902	0.1588	0.2758	0.3219	0.4842
		z	1.4089	-0.0094	0.9590	0.9575	-0.2164
	eq2	θ_3^*	0.0407	0.1476	0.2559	0.3458	0.4317
		θ_4^*	0.0345	0.1546	0.2514	0.3388	0.4471
		z	3.3936	-1.8452	0.6938	0.7738	-1.2350
	eq3	θ_3^*	0.0520	0.1056	0.1843	0.2346	0.3398
		θ_4^*	0.0477	0.1053	0.1836	0.2250	0.3267
		z	1.7624	0.0530	0.0693	0.8434	0.8029

The first two blocks are the test results of the null hypotheses (14) and (15), whereas the last two blocks are the test results of the null hypotheses (16) and (17). The Z statistic is the corresponding test statistic.

financial data even less exploited. This underfitting may partially be responsible for the inferior performance observed in [1], [16] and many other studies. This may also help explain why, in some applications, we found that the post-sample performance was even better than the in-sample performance ([13]).

On the other hand, the ability to use cross-validation to resolve the problem of overfitting may be overestimated. As we have seen from our simulations, the validation design cannot effectively prevent any of the four overfitting cases detected in the designs without the use of the validation sample. Under these circumstances, using the validation design may mislead one to believe that the overfitting problem has been well taken care of, while in fact it has not ([12]).

In conclusion, the essence and the implication of this paper is a call for a prudent use of the overfitting-avoidance design in financial data mining. Instead of taking it for granted, this paper provides a thorough analysis of the validation design frequently used in financial applications of GP. Our results lead us to be very suspicious of its contribution. Our analysis also points out another weakness of existing financial applications of GP, i.e. the causal design of search intensity. So far, few studies have carefully documented the effect of different population sizes and different numbers of generations on information exploitation. Using the production theory from economics, [5] showed that what matters is the *combined effect* of *Pop* and *Gen*. Studies which change only *Pop* or *Gen* at one time may underestimate the significance of search intensity. We believe that a proper addressing of search intensity with a more effective overfitting-avoidance design may give rise to another series of interesting financial applications of GP, which is the direction for the next study.

Acknowledgements

The authors are grateful to two anonymous referees for their comments and suggestions.

References

1. Allen, F., Karjalainen, R.: Using Genetic Algorithms to Find Technical Trading Rules. Journal of Financial Economics **51**(2) (1999) 245–271
2. Bender, E.: Mathematical Methods in Artificial Intelligence. IEEE, Los Alamitos, Calif. (1996)
3. Chen, S.-H., Yeh, C.-H.: Toward a Computable Approach to the Efficient Market Hypothesis: An Application of Genetic Programming. Journal of Economic Dynamics and Control **21**(6) (1997) 1043–1063
4. Chen, S.-H. (ed.): Genetic Algorithms and Genetic Programming in Computational Finance. Kluwer Academic Publishers (2002)
5. Chen, S.-H., Kuo, T.-W.: Genetic Programming: A Tutorial with the Software Simple GP. In: Chen, S.-H. (ed.), Genetic Algorithms and Genetic Programming in Computational Finance, Kluwer Academic Publishers (2002) 55–77
6. Geman, S., Bienenstock, E., Doursat, R.: Neural Networks and the Bias/Variance Dilemma. Neural Computation **4**(1) (1992) 1–58

7. Kaboudan, M. A.: A Measure of Time Series's Predictability Using Genetic Programming Applied to Stock Returns. Journal of Forecasting **18** (1999) 345–357
8. Kecman, V.: Learning and Soft Computing: Support Vector Machines, Neural Networks, and Fuzzy Logic Models. MIT Press (2001)
9. Mehrotra, K., Mohan, C., Ranka, S.: Elements of Artificial Neural Networks. MIT Press, Cambridge, Mass. (1997)
10. Neely, C., Weller, P., Ditmar, R.: Is Technical Analysis in the Foreign Exchange Market Profitable? A Genetic Programming Approach. Journal of Financial and Quantitative Analysis **32**(4) (1997) 405–427
11. Neely, C. J., Weller, P. A.: Technical Trading Rules in the European Monetary System. Journal of International Money and Finance **18**(3) (1999) 429–458
12. Neely, C. J., Weller, P. A.: Using GP to Predict Exchange Rate Volatility. In: Chen, S.-H. (ed.): Genetic Algorithms and Genetic Programming in Computational Finance. Kluwer Academic Publishers, Boston Dordrecht London (2002) 263–279
13. Noe, T. H., Wang, J.: The Self-Evolving Logic of Financial Claim Prices. In: Chen, S.-H. (ed.): Genetic Algorithms and Genetic Programming in Computational Finance. Kluwer Academic Publishers, Boston Dordrecht London (2002) 249–262
14. Smith, M.: Neural Networks for Statistical Modeling. Van Nostrand Reinhold, New York (1993)
15. Stone, M.: Cross-Validatory Choice and Assessment of Statistical Predictors. Journal of the Royal Statistical Society **B36** (1974) 111–147
16. Wang, J.: Trading and Hedging in S&P 500 Spot and Futures Markets Using Genetic Programming. Journal of Futures Markets **20**(10) (2000) 911–942