15. Discovering Hidden Patterns with Genetic Programming

Shu-Heng Chen and Tzu-Wen Kuo

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

In this chapter, we shall review some early applications of genetic programming to financial data mining and knowledge discovery, including some analyses of its statistical behavior. These early applications are known as *symbolic regression* in GP. In this type of application, genetic programming is formally demonstrated as an engine searching for the hidden relationships among observations. Here, we find evidence of the closest step ever made toward the original motivation of John Holland's invention of genetic algorithms: *Instead of trying to write your programs to perform a task you don't quite know how to do, evolve them.*

15.1 Discovering the Hidden Law of Motion

In the first type of application, genetic programming is applied to discover the underlying law of motion, or the data-generation process. The law of motion can be a deterministic chaotic process,

$$x_t = f(x_{t-1}, ...) \tag{15.1}$$

a stochastic process,

$$x_t = f(x_{t-1}, ...) + \epsilon_t \tag{15.2}$$

$$x_t = f(x_{t-1}, y_{t-1}, z_{t-1}, x_{t-2}, y_{t-2}, z_{t-2}, ...) + \epsilon_t \tag{15.3}$$

or a regression,

$$y_i = f(x_{1,i}, x_{2,i}, ...) + \epsilon_i \tag{15.4}$$

In all these applications, GP is used to discover the underlying data-generation process of a series of observations. While this type of application is well known to econometricians, the perspective from GP is novel. As [15.14] has stated,

> An important problem in economics is finding the mathematical relationship between the empirically observed variables measuring a system. In many conventional modeling techniques, one necessarily begins by selecting the size and shape of the model. After making

S.-H. Chen et al. (eds.), *Computational Intelligence in Economics and Finance*

> this choice, one usually then tries to find the values of certain coefficients required by the particular model so as to achieve the best fit between the observed data and the model. But, in many cases, *the most important issue is the size and shape of the model itself.* (p.57; italics added.)

Econometricians offer no general solution to the determination of *size* and *shape* (the functional form), but for Koza, finding the functional form of the model can be viewed as *searching a space of possible computer programs* for the particular computer program which produces the desired output for given inputs.

Koza employed GP to rediscover some basic physical laws from experimental data, for example, Kepler's third law and Ohm's law ([15.15]). He then also applied it to eliciting a very fundamental economic law, namely, the *quantity theory of money* or the *exchange equation* ([15.14]). Genetic programming was thus formally demonstrated as a *knowledge discovery* tool. This was probably the closest step ever made toward the original motivation of John Holland's invention: "Instead of trying to write your programs to perform a task you don't quite know how to do, *evolve them.*" Indeed, Koza did not evolve the parameters of an arbitrary chosen equation; instead, he evolved the whole equation from scratch, i.e. from the primitives of GP.

15.1.1 Deterministic Chaotic Processes

[15.7] applied GP to learn the following three chaotic equations:

$$x_{t+1} = 4x_t(1 - x_t), \quad x_t \in [0, 1] \ \forall t \tag{15.5}$$

$$x_{t+1} = 4x_t^3 - 3x_t, \quad x_t \in [-1, 1] \ \forall t \tag{15.6}$$

$$x_{t+1} = 8x_t^4 - 8x_t^2 + 1, \quad x_t \in [-1, 1] \ \forall t \tag{15.7}$$

Four experiments were implemented for each series. For each experiment, GP was run for 1,000 generations. For series (15.5) and (15.6), GP was able to discover the underlying law of motion in all four experiments. However, the number of generations required for this discovery was different. For series (15.5), it took 7, 12, 14, and 19 generations, whereas for series (15.6), it took 29, 37, 37, and 70 generations. As for series (15.7), GP failed to discover the law of motion in three out of the four simulations, and for the only success the law of motion was discovered at the 151th generation. These experiments show the effect of the length of the LISP S-expression (the algorithmic complexity of the program) on discovery.

These three laws of motion differ in 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:

$$(* (4 * (x_t (- 1 x_t)))) \tag{15.8}$$

$$(- (* 4 (* x_t (* x_t x_t))) (* 3 x_t)) \tag{15.9}$$

$$(+ (- (* 8 (* x_t (* x_t (* x_t x_t))))) (* 8 (* x_t x_t))) 1) \tag{15.10}$$

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. (15.8) to (15.10), the lengths of the LISP S-expressions are 7, 11, and 17, respectively. Therefore, in terms of algorithmic complexity, Eq. (15.5) is the simplest, while Eq. (15.7) is the most complex. [15.7] then examined how this difference might affect the performance of GP.

The *function set* originally employed by [15.7] is $\{+, -, \times, \%\}$, and the terminal set is $\{x_t, \mathcal{R}\}$, where $\mathcal{R}$ is the ephemeral random constant. If we add the function "cubic" to the function set, then the minimal description of Eq. 15.6 is simply

$$(- (* 4 (\mathit{cubic}\ x_t)) (* 3 x_t)), \tag{15.11}$$

and the program length of it is only 8. Alternatively, if we add x_t^3 to the terminal set, then the minimal description becomes

$$(- (* 4 x_t^3) (* 3 x_t)). \tag{15.12}$$

In this case, the program length of Eq. (15.6) is even shorter and is 7, which is the same as that of Eq. (15.5). It is, therefore, likely that it will discover the hidden law of series (15.6) as easily as it will discover that of series (15.5). So, depending on the user-supplied function set and terminal set, a mathematical function can have different program lengths.[1]

Formally speaking, let $\mathcal{T}$ be the terminal set and $\mathcal{F}$ be the function set, and let us denote their cardinality by $\mid \mathcal{T} \mid$ and $\mid \mathcal{F} \mid$.

$$Prob(f^* \in G_n) = P[K(f^* \mid \mathcal{F} \bigcup \mathcal{T})], \tag{15.13}$$

and

$$\frac{\Delta Prob(f^* \in G_n)}{\Delta K} \mid_{\mathcal{F} \bigcup \mathcal{T}} \leq 0, \tag{15.14}$$

where f^* is an targeted decision rule, say, the optimal decision rule, G_n is the nth generation of population, and $K(f^* \mid \mathcal{F} \bigcup \mathcal{T})$ refers to the algorithmic complexity of the decision rule of f^* given the functional set $\mathcal{F}$ and the terminal set $\mathcal{T}$.

[1] While the assertion and the example given above are based on deterministic functions, we are conducting a study to see whether this assertion, or some modifictaion of it, can be applied to functions with noise.

Since a larger function set and a terminal set will help abbreviate the representation of a decision rule, the algorithmic complexity of a function can only be non-positively related to $\mid \mathcal{F} \mid$ and $\mid \mathcal{T} \mid$, i.e.

$$\frac{\Delta K}{\Delta \mid \mathcal{F} \mid} \leq 0, \quad and \quad \frac{\Delta K}{\Delta \mid \mathcal{T} \mid} \leq 0. \tag{15.15}$$

Going back to Eq. (15.14), it seems that a larger terminal set and a larger function set can enhance search efficiency. In fact, there is empirical evidence that suggests that this is indeed the case ([15.10]). Still, the influence of search space and population size also has to be taken into account. Let $\mathcal{S}$ be the search space, which is a collection of all potential species. The size of it, $\mid \mathcal{S} \mid$, grows exponentially with $\mid \mathcal{F} \mid$ and $\mid \mathcal{T} \mid$. Therefore, if population size does not grow exponentially with $\mid \mathcal{F} \mid$ and $\mid \mathcal{T} \mid$,

$$\lim_{|\Delta \mathcal{F}|, |\Delta \mathcal{T}| \to \infty} s = 0, \tag{15.16}$$

where

$$s = \frac{\mid G \mid}{\mid S \mid}. \tag{15.17}$$

To take into account the effect of s, let us rewrite Eq. (15.13) into a conditional density,

$$Prob(f^* \in G_n \mid s) = P[K(f^* \mid \mathcal{F} \bigcup \mathcal{T})], \tag{15.18}$$

Eq. (15.18) says that the probability of finding f^* in a finite number of generations n is *conditional upon* the population size ratio s. Since in practice the population size ratio cannot grow exponentially with the size of the function set and the terminal set, reducing the algorithmic complexity of a decision rule by enlarging the terminal and function sets may help gain a little efficiency. Therefore, constrained by the population size ratio, the sizes of $\mathcal{F}$ and $\mathcal{T}$ have to kept within certain bounds.

[15.8] conducted a series of experiments to test what has been analyzed above. They started with the function set $\mathcal{F} = \{+, -, \times, /\}$, and the terminal set $\mathcal{T} = \{X_{t-1}, 0, 1, 2, ..., 9\}$. Thus, as predicted above, in a finite number of generations, the probability of discovering (15.6) is lower than that of discovering (15.5). To show how much lower it is, [15.8] conducted two experiments. In the first experiment $\{x_t\}$ was generated from Eq. (15.5), and in the second experiment it was generated from Eq. (15.6). For each experiment, they ran 100 trials of GP. This number of runs enables us to get an estimate of $Prob(g^* \in G_{200})$ (Eq. (15.13)). As a result, they found that the chance of discovering (15.5) within 200 generations was about 83%, whereas there was only a 5% chance of discovering (15.6) with the same search intensity.

However, if we add the variable "X^3_{t-1}" to the terminal set, then the algorithmic complexity of Eq. (15.6) becomes shorter – 7, to be exact, which

is the same as that of the Data Generating Mechanism (15.5). To see the effect of adding the terminal X_{t-1}^3, they conducted another two experiments. In both experiments, $\{x_t\}$ were generated from Eq. (15.5). The first experiment did not include X_{t-1}^3 in $\mathcal{T}$, but the second did. 100 trials were run for each experiment. For the experiment that did not include X_{t-1}^3, $Prob(g^* \in G_{200})$ was only 4%. However, after the inclusion of X_{t-1}^3, $Prob(g^* \in G_{200})$ increased up to 84%, which was about the same as the chance of finding (15.5) in the previous experiment.

A series of other chaotic laws of motion were also studied by George Szpiro and Mahamoud Kaboudan. [15.19] applied GP to learn and to forecast a set of chaotic time series including the *Rossler Attractor*,

$$x(t+\delta) = x(t) - [y(t) + z(t)]\delta, \tag{15.19}$$

$$y(t+\delta) = y(t) + [x(t) + ay(t)]\delta, \tag{15.20}$$

$$z(t+\delta) = z(t) + [b + x(t)z(t) - cz(t)]\delta, \tag{15.21}$$

and the *Mackey-Glass delay differential equation*,

$$\dot{x}(t) = bx(t) + \frac{ax(t-\tau)}{1 + x^c(t-\tau)}. \tag{15.22}$$

For the Rossler attractor, i.e. Eqs. (15.19)-(15.21), Szpiro simulated the time series with $a = 0.2, b = 0.2, c = 0.57$, initial values $x_0 = -1, y_0 = 0, z_0 = 0$, and $\delta = 0.02$. Genetic programming was applied to predict the time series $\{x_t\}$ and $\{z_t\}$. This produced an R^2 of 0.912 for the first series, whereas only 0.568 for the second. For Eq. (15.22), Szpiro considered the series with parameters $a = 0.2$, $b = 0.1$, $c = 10$, and $\tau = 30, 100$. [2] When $\tau = 30$, genetic programming can predict the series with an R^2 value of between 0.40 and 0.65, and when $\tau = 100$ it can produce R^2 values of between 0.45 to 0.80.

[15.21] and [15.22] also examined the GP forecasting performance over the *Henon map*,

$$x_t = 1 - 1.4x_{t-1}^2 + 0.3x_{t-2}, \tag{15.23}$$

and the *tent map*,

$$x_t = \begin{cases} cx_{t-1} & if \quad x_{t-1} < 0.5, \\ c - cx_{t-1} & if \quad x_{t-1} > 0.5. \end{cases} \tag{15.24}$$

For both, the performance of GP was surprisly good.

[2] The significance of the τ is its associated fractal dimension. For the case $\tau = 30$, it was shown that $x(t)$ is chaotic with fractal dimension 3.5, and for the case where $\tau = 100$, about 7.5. The data for the MacKey-Glass time series can be obtained by applying the Runge-Kutta method.

15.1.2 Nonlinear Stochastic Time Series

The deterministic chaotic time series can hardly satisfy the needs of economists and finance people, because it is difficult to believe that economic and financial time series are generated only in a deterministic manner. Therefore, in many applications, Eqs. (15.2) and (15.3) are assumed to be a typical environment to work with. For the former, in addition to the artificial data, GP was also applied to the real financial data, in particular the stock price and the foreign exchange rate. We shall see more of these applications in Section 15.2.

In a noisy environment, GP will not do anything miraculous. As we shall see in Section 15.2, when the signal-noise ratios become less and less, the GP performance will also get worse and worse. In the extreme case, when the data is purely obtained from pseudo-random numbers, GP will completely collapse. Therefore, like all other data mining tools, our expectations of GP should be no more than just discovering the hidden law, be it linear or nonlinear.

15.2 Statistical Behavior of Genetic Programming

15.2.1 Kaboudan's Predictability Test

[15.11] proposed a *shuffling test* that measures GP's predictive ability. The idea is quite simple. If the series $\{x_t\}$ contains a predictable pattern, then randomly shuffling the series $\{x_t\}$ will ruin predictable information in the original sequence. Therefore, the series after shuffling, say $\{z_t\}$, should be more difficult to predict than the original series $\{x_t\}$. However, if the series contains no predictable patterns, then $\{x_t\}$ and $\{z_t\}$ should be equally hard to predict. Based on this idea, a statistic is developed.

$$\eta = max\left\{0, \frac{1}{n}\sum_{i=1}^{n}\left(1 - \frac{SSE_x}{SSE_z}\right)_i\right\} \tag{15.25}$$

The η statistic is the average ratio of the (before and after shuffling) SSEs subtracted from one. If the series before shuffling contained unpredictable patterns, the ratio of the fitness measure of the *sum of squared errors* (or *SSE*) before to the SSE after running must approach *unity*.

$$\frac{SSE_x}{SSE_z} \rightarrow 1. \tag{15.26}$$

If that series contained predictable patterns, the ratio must be less than unity.

$$\frac{SSE_x}{SSE_z} < 1. \tag{15.27}$$

n refers to the sample size, which is the number of runs that GP is applied to the series $\{x_t\}$ and $\{z_t\}$. It is not difficult to see that[3]

$$0 \leq \eta \leq 1. \tag{15.28}$$

This predictability measure was applied to examine the predictability of some artificial data as well as financial data. For the artificial data, Kaboudan considered five types of laws of motion: linear, non-linear, linear-stochastic, non-linear-stochastic and pseudo-random. In addition to the Henon map, the Mackey-Glass function, and two pseudo-random numbers, there were the following six equations.

$$x_t = 1.8708x_{t-1} - x_{t-2} \tag{15.29}$$

$$x_t = 3.9sin(x_{t-1}) + 0.85cos(x_{t-2}) \tag{15.30}$$

$$x_t = (1.43 - 4.5e^{-x_{t-1}^2})x_{t-2} \tag{15.31}$$

$$x_t = 0.6x_{t-1} + 0.15x_{t-2} + \epsilon_t \tag{15.32}$$

$$x_t = 0.6x_{t-1} + 0.3\epsilon_t x_{t-1} + \epsilon_t \tag{15.33}$$

$$x_t = \epsilon_t\sqrt{h_t}, \quad h_t = 1 + 0.25x_{t-1}^2 + 0.7h_{t-1} \tag{15.34}$$

$\epsilon_t \sim N(0,1)$. It was found that the linear process (15.29) was very predictable. Non-linear processes (15.30, 15.23, 15.31) free from noise were also quite predictable. The Mackey-Glass equation (15.22) generated the least predictable series among the noise-free non-linear series. Predictability decreased significantly when a stochastic component was part of the process, and the linear-stochastic (15.32) processes were more predictable than the non-linear-stochastic processes (15.33, 15.34). The random series investigated were totally unpredictable.

For financial data, Kaboudan considered three different frequencies. Prices were collected every 30 minutes, every minute, and every price change. Eight Dow Jones stocks were included in the analysis: Boeing, GE, GM, IBM, Sears, AT&T, Walmart, and Exxon. Time and quotes (TAQ) data available on CD-ROM from the NYSE, Inc. covering October 1996 through March 1997 were used. Analysis of 30-minute and one-minute data showed that none of the prices were predictable. The best which we have is one-minute AT & T with a predictability measure of 50%. Analysis of price-change returns showed significant improvements in terms of predictability.

We have a few remarks about this predictability measure. First, it is interesting to note the relationship between Theil's U statistic and Kaboudan's η statistic. Theil's U statistic would compare the SSE from GP with that

[3] Actually, it is possible that η may be less than 0. This may happen if shuffling actually introduced some pattern into an originally random data sequence. It may also happen if GP is not successful in depicting any pattern in the original data but is somehow successful in finding a pattern in the shuffled data.

from a chosen benchmark, but based on the same time series. Kaboudan's η statistic applies the same forecasting model to different series. The η statistic can be defined with other classes of models, such as neural nets. There is no guarantee that the predictability measure would be the same when GP is replaced with other models. This is probably a limitation inherent in this measure. Furthermore, as shown in [15.8], different settings of the control parameters may lead to different results. This is particularly true for the function (15.24), for which, to satisfy the closure property, one needs to include "if-then-else" in the function set. Similarly, the variable ϵ_t is required for the function (15.33). Since they were not included in the primitives in Kaboudan's application, the η statistic may not even be unique with respect to GP.

15.2.2 The Fitted Residuals of GP

The conventional diagnostics of fitted residuals were also applied to examine the statistical behavior of GP ([15.13]). Let the fitted function $\hat{f}$ derived from genetic programming, and the fitted residuals be

$$\hat{\epsilon}_t = x_t - \hat{f}(x_{t-1}, x_{t-2}, ...). \tag{15.35}$$

Then the question to be addressed here is whether the series $\hat{\epsilon}_t$ is satisfied with some standard properties from the statistical viewpoint; in particular, is the series $\hat{\epsilon}_t$ independent? If GP has successfully discovered the hidden relation, then one should expect that $\hat{\epsilon}_t$ is independent, or *white noise*, if it discovered the hidden linear relation. Therefore, the statistical behavior of the fitted residuals provides a way of evaluating the performance of GP.

The models considered in ([15.13]) are basically the same five types of models used in [15.11]. Among the noise-free equations are (15.29), (15.5), (15.23), (15.22), and the following linear threshold regression model:

$$x_t = \begin{cases} 2.5x_{t-1} + z_t & if \quad \mid x_{t-1} \mid \leq 2, \\ 0.85x_{t-1} & if \quad \mid x_{t-1} \mid > 2, \end{cases} \tag{15.36}$$

where

$$z_t = \begin{cases} 1 & if \ \ t \ \ is \ \ odd, \\ -1 & if \ \ t \ \ is \ \ even. \end{cases} \tag{15.37}$$

The linear stochastic functions involved are the AR(1) and ARMA(1,1) processes:

$$x_t = 0.9x_{t-1} + \epsilon_t, \tag{15.38}$$

and

$$x_t = 0.9x_{t-1} - 0.6\epsilon_{t-1} + \epsilon_t. \tag{15.39}$$

The non-linear stochastic functions selected for the examination are (15.33), (15.34), and the *threshold auto-regressive* (TAR) model:

$$x_t = \begin{cases} 0.9x_{t-1} + \epsilon_t & if \quad \mid x_{t-1} \mid \leq 1, \\ -0.3x_{t-1} + \epsilon_t & if \quad \mid x_{t-1} \mid > 1. \end{cases} \tag{15.40}$$

Finally, there are two pseudo-random numbers: one follows the normal distribution, and one follows the uniform distribution.

The null hypothesis that $\{\hat{\epsilon}_t\}$ is white was not rejected in (15.38), (15.39), (15.40), and (15.34). Among these four, the *normalized mean squared error* (NMSE) for Eq. (15.34) is the highest (0.70). [4] However, this result is not surprising, since, as we mentioned earlier, Eq. (15.34) is not predictable in terms of the mean. As a matter of fact, given the nature of unpredictability, we consider 0.70 low enough and it may even be low enough to indicate the possibility of *overfitting*. A similar result is also observed in the two pseudo random numbers. In both cases of normal and uniform distributions, the NMSE is 0.67, even though there is entirely nothing to learn from these two series. These empirical results suggest that overfitting in some cases can be a problem for genetic programming.

The null was also not rejected in Eq. (15.22). The NMSE in this case is 0.48, which is rather high considering that the series to be fitted is noise-free. Given this rather high NMSE, it is unlikely that the fitted residuals would be white. However, one should note that Equation (15.22) is noise free. In that case, it is questionable whether whiteness tests are still valid. This problem is also reflected in the remaining four noise-free series, in which the white noise in all cases is rejected. Actually Kaboudan warned readers not to use this to judge the adequacy of GP, because the associated normalized mean squared errors are very low in these four cases, which suggests that $\hat{f}$ actually fits f very well.

15.3 Discovering the Technical Trading Rules

In 1992 and subsequent years, partially because of the seminal work of [15.4], which showed the significant prediction potential of the simple trading strategies, applications of evolutionary computation were extended further to the study of trading strategies.[5] [15.5] and [15.1] initialized this line of research,

[4] The normalized mean squared error is the mean squared error divided by the sample variance. The NMSE is zero when a perfect prediction is generated, and is one when the resulting model is doing no better than predicting the mean value of the data set on average.

[5] [15.4] studied two of the simplest and most popular trading rules, the moving average and the trading range break, and found significant positive returns by utilizing the trading rules.

and [15.6] gave finance people the first systematic introductory book on genetic algorithms. Papers on this subject, however, did not find their way into prestigious journals until the late 1990s ([15.17], [15.2]).

15.3.1 Foreign Exchange Markets

[15.17] and [15.18] used GP techniques to find technical trading rules in the *foreign exchange market*. The former basically studied the *dollar exchange rate markets*, including the dollar/deutschemark, dollar/yen, dollar/pound, and dollar/Swiss franc markets, whereas the latter studied the EMS (European Monetary System) exchange rate markets, including the deutschemark/French franc, deutschemark/lira, deutschemark/guilder and deutschemark/pound markets. Two general issues were raised in this line of research:

1. Is it possible to identify *profitable trading rules* ex ante for these currencies using genetic programing?
2. If yes, what might be the source of the excess returns? In particular, can the observed excess returns be explained as compensation for bearing *systematic risk*?

Profitability and the Efficient Market Hypothesis. The answer is *positive* for the first issue, and *negative* for the second. These together make genetic programming particularly attractive for the investors in the foreign exchange market. [15.17] obtained rules for currencies selected in a given sample period (1975-1980), and then examined their performance over the period 1981-1995. They found that the generated rules earned *significant* out-of-sample excess returns over the period 1981-1995. [15.18] obtained rules for currencies selected in a given sample period (1979-1983), then examined their performance over the period 1983-1996. This finding is the same as that of [15.17]. In fact, in [15.18], the evidence of positive excess returns was further strengthened by careful statistical analysis involving calculating *Bayesian posterior probabilities* and the *Newey-West correction for serial correlation.* To understand whether the returns to the trading rules could be interpreted as *compensation for bearing systematic risk*, *beta* was calculated. Different benchmarks were selected for calculating beta, such as the MSCI (Morgan Stanley Capital International) world equity market index, the S&P 500, the Commerzbank of German equity, and the Nikkei. Most of the estimated betas were negative, indicating that the excess returns observed were not compensation for bearing systematic risk.

What Did GP Rules Learn?. If the excess returns were not compensation for bearing systematic risk, then what might be the source of the excess returns? To trace the source, both [15.17] and [15.18] conducted a series of analyses for the rules discovered by GP. First, *what kind of rules were found?* Were they similar to the rules commonly used by technical analysts? This

question is generally not easy to answer, because the rules discovered by GP often have a rather complex nested structure. In [15.17], the mean number of nodes for the 100 best rules obtained from 100 independent runs was 45.58, and only two rules had fewer than 10 nodes. Those whose structure could be harnessed were found to be the familiar moving-average rules or double moving-average rules. To confirm whether it was these moving-average and filter rules that contributed to the excess returns observed, [15.18] compared the performance of some simple MA rules and filter rules with the GP rules. Among the ten simple rules they examined, none could match the performance of GP rules over all currencies, and, in fact, many performed rather poorly. Therefore, they concluded that GP rules perform more profitably than the simple rules in EMS currencies. *What GP discovered was certainly richer than what simple rules can tell us.*

Second, *were the rules simply exploiting known statistical properties of the data?* To answer this question, bootstrapping simulations, based on a random walk model, an ARMA model, and an ARMA-GARCH(1,1) model, were carried out. Bootstrapping permits one to determine whether the observed performance of a trading rule is likely to have been generated under a given model for the data-generating process. The results indicate that it is unlikely to replicate the observed performance from any of the three models used for bootstrapping. In other words, *what GP rules discovered were not just some well-known econometric patterns.*

Third, *were the rules discovered by GP exploiting quite general features of financial markets?* This question is mainly motivated by the claim made by technical analysts that the rules they use are not specific to any particular market. To investigate this claim, they took the successful dollar/deutschemark rules and ran them on the data for the other exchange rates. There was a marked improvement in performance over the previous rules in all cases except the dollar/yen. The authors attempted to use this result to support the claim, but the result seems too good to make such a conclusion. In fact, the result brings us a puzzle: *why did the rules trained on dataset A outperform the rules trained on dataset B when both were applied to dataset B?* We would like to draw readers' attention to [15.12] on a similar issue. It was shown in [15.12] that the evolved forecasting rules differ from day to day for the same stock and among stocks. Hence, the supporting evidence for the existence of the general rules is weak.

Fourth, from the viewpoint of knowledge discovery, *what did these GP rules actually discover*? What can we learn from these GP rules? This question was better addressed in [15.18]. [15.18] examined the structure of some of the simplest rules obtained for each currency. A feature that emerges consistently across all currencies is that the *interest differential* is the most important informational *input* to the trading rule. They confirmed the importance of the interest differential information by performing the following experiment. They ran the trading rules for all currencies over the validation period

with the interest differential set to zero in the signal calculations, but included in the return calculations. The mean annual return fell to -0.24% and annual trading frequency dropped to 0.24.

Diversity of GP Rules. For each exchange rate, the authors also examined the *homogeneity* (or the hetrogeneity) of the rules, i.e. whether the rules signal the same position, long or short, at the same time. The results are quite different for different currencies. A high proportion of the 100 best dollar/deutschemark rules, obtained from 100 independent runs, identified similar patterns in the data: for most of the time, 90% or more of the rules gave the same signal. However, in the case of the dollar/yen, for a substantial amount of time, the proportion hovered around 50%, indicating the maximum differences for these rules.

Given the great diversity of the rules, it would be interesting to know whether voting would help. That is, would a majority vote over all 100 rules determine the position of the trading rule? So, a long position was taken if 50 or more of the rules signalled long, and a short position otherwise. This is also called the *median portfolio rule*. The return of a median portfolio rule was calculated by the authors, and the result was again inconsistent. For some exchange rates, this median portfolio rule produced a substantial increase in excess return. However, for certain others, it did not.

Overfitting Avoidance. [15.17] and [15.18] adopted a technique, known as *the validation procedure*, to avoid overfitting. 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 the *selection period*, which is used to select the best performing programs and decide when to stop the training. For example, in [15.17], the termination criterion is achieved if no new best rule appears for 25 generations. However, it was found that this procedure did not eliminate the problem of overfitting, because the rule that did best in the selection period turned out to be one of the poorest in terms of performance out-of-sample. In addition to the *validation procedure*, they also experimented with different transaction costs, and found that a suitable chosen transaction cost in the training and selection periods introduced a useful friction that provides some protection against overfitting the data.

15.3.2 Stock Markets

[15.2] applied GP to the trade S&P 500 index from 3 January 1928 to 29 December 1995. This paper shares many similarities with [15.17] and [15.18]. First, they also transformed the data by dividing each day's price by a 250-day moving average. The normalized prices had an order of magnitude of around one. Second, the *validation procedure* was also adopted, but in a different style. [15.2] used ten successive training periods and reported the summarized results for each case. The five-year training and two-year selection periods started in 1929, 1934,..., 1974, with the out-of-sample test periods starting in

1936, 1941,..., 1981. With this design, the first trial used years 1929-1933 as the training period, 1934-1935 as the selection period, and 1936-1995 as the test period. For each of the ten training periods, they carried out ten trials. This procedure is known as the *rolling forward scheme*, a technical device used to tackle the issue of data-snooping.

Profitability and Transaction Costs. The major result found by [15.2] was significantly different from that found by [15.17] and [15.18]. [15.2] found that the markets for these stocks are *efficient* in the sense that it is not possible to find consistent excess returns after transaction costs over a simple buy-and-hold strategy in the holdout sample periods. In most of the periods, there are only a few rules with positive excess returns. Obviously, this result sharply contrasts with that of [15.17] and [15.18], and leaves the capabilities of GP to find successful technical trading rules *inconclusive*. It would be interesting to know what may cause this difference. Here, transaction costs seem to play an important role.

The transaction costs incorporated by [15.2] were much higher than the costs used by [15.17] and [15.18]. The former used one-way transaction costs of 0.1, 0.25, 0.5%, whereas the latter used one-way transaction costs of 0.05%. As we already saw from [15.17], transaction costs have a crucial implication for GP performance. In [15.2], some new evidence was presented. First, it was shown that the differences between "in" and "out" returns were statistically significant for the low and median transaction cost cases, but not for the high transaction cost cases. Second, trading frequency was adversely affected by the transaction costs. With low transaction costs, the trading frequency was high, with an average of 18 trades per year. However, with high transaction costs, the trading frequency dropped to an average of 1.4 trades per year.

One temptation here is to ditto [15.17] or [15.18] with higher transaction costs. However, it is not a really wise direction to move in, since we all know that keeping on increasing transaction costs will eventually discourage any possible trade. One possible solution is to report the *break-even transaction cost* proposed by [15.3], which is the level of transaction costs leading to zero trading profits.[6] By following this approach, one can test whether the break-even transaction cost in the stock market is higher than the one in the foreign exchange market. Furthermore, one can rank different markets according to their break-even transaction cost. This could be a direction to move in the future.

Diversity of Rules. Like [15.17] and [15.18], [15.2] provided a thorough analysis of the trading rules discovered by GP. Questions regarding the nature of the rules were addressed. They started with an examination of the *diversity* of the rules. In a similar way to what was found in [15.17] and [15.18], there existed a great degree of diversity in the rules discovered by

[6] [15.20] applied the break-even transaction cost to evaluate the performance of GA trading rules.

GP over different trials. The size of the rules varied from nine to 94 nodes, with a depth of between five and ten levels. While many of the rules appeared to be quite complicated at first sight, there was often a lot of redundant material. For example, some of the subtrees may never be visited when the rules are evaluated. Hence, seemingly complex tree structures may be pruned into much simpler ones. Based on that observation, they argued that *the complexity of the trading rules in general cannot be measured by the number of nodes.*[7]

Financial Knowledge Discovery with GP. At this point, they proposed an alternative measure of the complexity, i.e. to compare the behavior of the rules to a class of simple technical trading rules. Here, we are asking the same question as [15.17] did before: *did the GP trading rules tell us anything more than just the simple technical trading rules*? The answer to this question is "*it depends.*" Actually, it interestingly depends on the *transaction costs* and the *training periods.* With 0.25% transaction costs, most rules are effectively similar to a 250-day moving-average rule. With 0.1% transaction costs, half of the rules are similar to a 250-day moving-average rule, while the remaining rules resemble either 10-to-40-day moving-average rules or a trading range break rule. With 0.5% transaction costs, rules from the early training periods are similar to a 250-day moving-average rule, while the late rule from the late training periods match none of the simple rules considered.

This glossary of results leads us to the best place to see how genetic programming can automate the discovery of finance theory without too much external human tutoring or supervising. The glossary above certainly tells us much more than what simple trading rules taught us. First, general cookbooks of technical analysis would not tell us anything about the connection between transaction costs and effective trading rules. Second, while most of us believe that markets can never settle down, only genetic programming can show us how one effective trading rule was replaced by another at a different moment. These dynamics were also not taught in conventional technical analysis. Third, as we already see in many cases, there is simply no match between simple trading rules and GP rules. In these scenarios, one may learn something new from the GP-evolved rules. Fourth, even if the match exists, non-trivial differences should not be overlooked.

Actually, [15.2] analyzed a few examples of their GP rules. In one case, the rule was found to be something similar to a moving-average rule, but the window of the moving-average was not fixed. During bear markets, the rule looked for very short-term trends, with the moving-average window varying between three and four days. Then, in bull markets, the length of the time window increased to 16 days at a normalized price level of 1.1, and rose rapidly

[7] We believe that this is an important critique, because measuring the complexity of financial markets via measuring the complexity of evolved trading trees remains an interesting subject, and so far, the node complexity is the only measure popularly used.

thereafter to soon use all of the available price history. The interpretation of this rule given by [15.2] is that: "...this rule corresponds to a hypothesis that stock prices are in general going up but are prone to corrections after prolonged bull markets. This rule is quick to take a long position in bear markets, but is increasingly hesitant to do so when prices are high. (p. 263-264, ibid)."

Trade with A Delay. Remember that, earlier, [15.17] showed that the excess returns derived from their GP trading rules were not compensation for bearing systematic risk. Here, [15.2] also conducted a similar pursuit. They were searching for the potential explanations for the differences between "in" and "out" returns being statistically significant for the low and median transaction cost cases. To investigate this issue, the performance of the ex post trading rules was retested based on trades occurring with a delay of one day. This one-day delay was motivated by the well-known feature that there is low-order serial correlation in market indexes. This one-day delay should remove any first-order autocorrelation. It was found that, by using delayed signals, the average difference in returns during "in" and "out" days dropped from seven to two basis points. In the case of 0.1% transaction costs, a one-day delay removed all of the difference between returns during "in" and "out" days. They, therefore, concluded that the forecasting ability could be explained by low-order serial correlation in stock index returns.

Day Trading. However, this one-day delay is not applicable for the case of *day trading* based on high-frequency on-line data. [15.12] was the first to apply genetic programming to *day trading*. Using genetic programming, he established a trading system, known as a *single-day-trading-strategy* (**SDTS**). An SDTS is a strategy whereby a trader buys a stock at a price close to the *daily low* and sells at a price close to the *daily high*, regardless of the order in which trading occurs. The main task of GP is to forecast the daily low and high. The unique feature of the SDTS is to require the investor to close every position by the end of the trading day. This is mainly because trading is based on a one-day forecast. Holding no position overnight helps avoid greater gambling losses in subsequent days for which no forecast is available to base a decision on. For an empirical illustration, stocks of Chase Manhattan and General Motors from the NYSE and those of Dell Computer and MCI-Worldcom from the NASDAQ were selected to apply to test SDTS in [15.12]. For all four stocks, returns from trading based on the GP forecasts were higher than those based on a random walk.

It is worth noting here that GP was not applied to evolve trading strategy directly. Instead, it was applied to forecast the daily low and daily high. Based on the predicted daily low and daily high, the trading decision was made based on the intra-day tick-by-tick movement of the stock prices. Another difference which distinguished [15.12] from [15.2] was the trading target. What concerned [15.12] were the individual stocks of listed companies, rather than stock indexes.

15.3.3 Futures

Given the conflicting evidence on the capabilities of GP to find successful technical trading rules from the foreign exchange markets and the stock markets, [15.23] conducted another formal study of the issue: *is GP really useful in finding technical trading rules in financial markets?* It applied GP to find trading rules in S&P 500 futures, which are distinguished from their index counterpart by their sufficient marketability and liquidity. In addition, the transaction costs in futures trading are also significantly *lower* than the costs in equity trading. This consideration is important because one possible explanation for the difference between GP performance in the foreign exchange markets and the stock markets is the *transaction costs*. So far, there has been no empirical study showing whether their break-even transaction costs are the same. Therefore, by applying GP to find trading rules in futures, [15.23] is able to shed more light on whether, in addition to the foreign exchange markets ([15.17], [15.18]), GP can be useful in another liquid market with lower transaction costs. [15.23] used the daily data of the S&P 500 index, S&P 500 futures, and 3-month T-bill rates from 1983 to 1998.

Hedging. Given futures as the application domain, [15.23] further extended the application of GP from *just trading* to *hedging*, which is certainly infeasible for those earlier studies without the introduction of the futures market. This extension is empirically relevant since many investors use futures to hedge positions in the spot market, and this extension tests whether GP could be used for finding *hedging strategies*. Hence, in [15.23], GP was applied to generate trading rules for both the index and futures.

To do so, [15.23] encoded the trading rules with GP in a dramatically different way as opposed to [15.17], [15.18] and [15.2]. For the latter, the trading rules were generated by the function set, usually just the Boolean-valued functions and real-valued functions, and the terminal set of no imposed structure. However, for [15.23], one solution candidate or trading rule is represented as four hierarchical trees. Among them, the *third tree* generates a signal to be used in the *futures market*, and the *fourth tree* generates a signal to be used in the *spot market*. The root of these two sub-trees is restricted to the function *if-then-else*. Furthermore, the output of these two sub-trees is transformed into the range [0,1] by the sigmoid function. Moreover, by further dividing the range [0,1] into a few segments, one can actually decide how many units to buy or sell. This flexibility was also not shared by those earlier studies, which can only signal the time to buy and sell (one unit).

Automatically Defined Functions. The most striking feature of [15.23]'s representation of the trading rule is the employment of automatically defined functions (ADFs). The first two sub-trees are automatically defined functions. The first sub-tree is a Boolean-valued ADF, and the second sub-tree is a real-valued ADF. An ADF is one way of capturing the idea of *subroutine* and *reusable codes* in computer programming. The advantage of ADF is that the

same complex function can be called from multiple nodes by a single reference, so lower algorithmic complexity can be achieved. The ADF was proposed by John Koza in [15.16], but the economic and financial applications of it were not available until very recently. [15.23] was the first to bring the attention of economists to it.

Rolling Forward Procedure. To avoid the problem of overfitting, [15.23] also adopted the *rolling forward procedure* as [15.2] did. By this procedure, we consider an investor at the beginning of 1987. The investor had 3 years of data available, and he or she defined the training period to be 1984 and 1985 and the validation period to be 1986. Ten trials were run with periods specified in this way, and 10 trading programs were generated. These 10 programs were used to trade in 1987, the out-of-sample period, and the results were reported. In this way, the editor rolled forward until 1998, which means that a total of twelve rolling forward steps were taken and 120 independent trials were implemented.

What was not made clear in [15.2] was the significance of the *rolling forward scheme*. As [15.23] mentioned, an early version of [15.2] did not use this scheme. Instead, they used one training period, one validation period, and one out-of-sample period, and they found that GP-generated rules outperformed the buy-and-hold strategy in the out-of-sample period. Once they used the rolling forward scheme to prevent data-snooping bias in the selection of time periods, the results were no longer conclusive. In an early draft of [15.23], data-snooping bias also occurred. There, the training period was defined as April 1986 to September 1987 and the validation period as 1988 and 1989. Then GP-generated rules consistently beat the buy-and-hold strategy in the *only* out-of-sample period, 1990 to 1993. However, when the rolling forward scheme was adopted, things became, once again, inconclusive.

In both [15.2] and [15.23], it was found that, under the *rolling forward scheme*, the GP-generated rules could not consistently outperform the *buy-and-hold strategy. The performance of the GP-generated trading rules varied greatly for different years.* For example, in its experimental series "Trading in Futures Only," [15.23] found that 7 out of 10 rules outperformed the buy-and-hold strategy in 1992, whereas in 1995, only 1 out of 10 rules outperformed the buy-and-hold strategy. One may wonder why, if GP really has the capability of discovering, it fails to discover the buy-and-hold strategy. The answer to this is that the training period and the out-of-sample period may not always share such similar features. If they did share the same similar features and the returns to the buy-and-hold strategy were high for the training period, then GP would eventually learn to use a buy-and-hold strategy. This was the case for 1998, when the returns in the previous 3 years were high. Five of the 10 trading rules generated from 1995 to 1997 were just buy-and-hold.

When the objective was return only and both the index and futures markets were allowed to trade, the trading rules generated by GP were more likely to lead to trade in the futures market because the transaction costs

there were substantially lower. In the spot market, the low average number of position changes, from 1.6 to 6.3, indicated that the trading rules did not lead to trade much beyond the buy-and-hold strategy. In the futures market, the average number of position changes ranged from 4 to 68.5, which was much higher than the number of position changes for the buy-and-hold strategy, i.e. two. GP recognized that transaction costs were lower in the futures market and generated trading rules that more often than not resulted in trade in the futures market while the buy-and-hold strategy was adopted most of the time in the spot market.

15.4 Concluding Remarks

In this chapter, we have reviewed the two early applications of genetic programming to financial data mining and knowledge discovery. The application to time series modeling shows that GP has some power for tracking and predicting the complex dynamic behavior of financial time series, whereas the application to trading strategies indicates the potential of GP in relation to market-timing capability. For the application to trading strategies, one has generally started with a set of primitives, which may characterize state-of-the-art knowledge. From these primitives, we then let GP set in motion an evolutionary process and discover something novel for us. This style of application reveals an interesting and challenging way of doing finance in the future. As a matter of fact, recently, this style of application has just been extended to *option pricing*. For example, in their study of option pricing formulas, [15.9] included the Black-Scholes model as a primitive. The GP-evolved option pricing formulas for the equity options were hence adaptations of the Black-Scholes model. It is our belief that further progress in the theory of finance can be made in this way.

References

15.1 Allen, F., Karjalainen, R. (1993) Using Genetic Algorithms to Find Technical Trading Rules. Rodney L. White Center for Financial Research, The Wharton School, Technical Report, 20–93

15.2 Allen, F., Karjalainen, R. (1999) Using Genetic Algorithms to Find Technical Trading Rules. Journal of Financial Economics, Vol. 51, no. 2, 245–271

15.3 Bessembinder, H., Chan, K. (1995) The Profitability of Technical Trading Rules in the Asian Stock Markets. Pacific Basin Finance Journal, Vol. 3, 257–284

15.4 Brock, W.A., Lakonishok, J., LeBaron, B. (1992): Simple Technical Trading Rules and the Stochastic Properties of Stock Returns. Journal of Finance, Vol. 47, 1731–1764

15.5 Bauer, R. J. Jr., Liepins, G. E. (1992) Genetic Algorithms and Computerized Trading Strategies. In: O'Leary, D. E., Watkins, R. R. (Eds.), Expert Systems in Finance, North Holland

15.6 Bauer, R. J. Jr. (1994) Genetic Algorithms and Investment Strategies. Wiley
15.7 Chen, S.-H., Yeh, C.-H. (1997) Toward a Computable Approach to the Efficient Market Hypothesis: An Application of Genetic Programming. Journal of Economic Dynamics and Control, Vol. 21, no. 6, 1043–1063
15.8 Chen, S.-H., Kuo, T.-W. (2002) 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, 55–77
15.9 Chidambaran, N., Lee, C.-W. J., Trigueros, J. (2002) Option Pricing via Genetic Programming. In: Chen, S.-H. (Ed.), Evolutionary Computation in Economics and Finance, Physica-Verlag, 383–397
15.10 Johnson, H. E., Gilbert, R. J., Winson, K., Goodacre, R., Smith, A. R., Rowland, J. J., Hall, M. A., Kell, D. B. (2000) Explanatory Analysis of the Metabolome using Genetic Programming of Simple, Interpretable Rules. Genetic Programming and Evolable Machines, Vol. 1, no 3, 243–258
15.11 Kaboudan, M. A. (1999) A Measure of Time Series's Predictability Using Genetic Programming Applied to Stock Returns. Journal of Forecasting, Vol. 18, 345–357
15.12 Kaboudan, M. A. (2002) GP Forecasts of Stock Prices for Profitable Trading. In Chen, S.-H. (Ed.), Evolutionary Computation in Economics and Finance. Physica-Verlag, 359–381
15.13 Kaboudan, M. A. (2001) Genetically Evolved Models and Normality of Their Fitted Residuals. Journal of Economic Dynamics and Control, Vol. 25, no. 11, 1719–1749
15.14 Koza, J. (1992) A Genetic Approach to Econometric Modelling. In: Bourgine, P., Walliser, B. (Eds.) Economics and Cognitive Science. Pergamon Press, 57–75
15.15 Koza, J. (1992a) Genetic Programming: On the Programming of Computers by Means of Natural Selection. The MIT Press.
15.16 Koza, J. (1994) Genetic Programming II : Automatic Discovery of Reusable Programs. The MIT Press.
15.17 Neely, C., Weller, P., Ditmar, R. (1997) Is Technical Analysis in the Foreign Exchange Market Profitable? A Genetic Programming Approach. Journal of Financial and Quantitative Analysis, Vol. 32, no. 4, 405–427
15.18 Neely, C. J., Weller P. A. (1999) Technical Trading Rules in the European Monetary System. Journal of International Money and Finance, Vol. 18, no. 3, 429–458
15.19 Szpiro, G. G. (1997a) Forecasting Chaotic Time Series with Genetic Algorithms. Physical Review E, 2557–2568
15.20 Pereira, R. (2002): Forecasting Ability But No Profitability: An Empirical Evaluation of Genetic Algorithm-Optimized Technical Trading Rules. In: Chen, S.-H. (Ed.), Evolutionary Computation in Economics and Finance. Physica-Verlag, 287–310
15.21 Szpiro, G. G. (1997b) A Search for Hidden Relationships: Data Mining with Genetic Algorithms. Computational Economics, Vol. 10, no. 3, 267–277
15.22 Szpiro, G. (2002) Tinkering with Genetic Algorithms: Forecasting and Data Mining in Finance and Economics. In: Chen S.-H. (Ed.) Evolutionary Computation in Economics and Finance. Physica Verlag.
15.23 Wang, J. (2000) Trading and Hedging in S&P 500 Spot and Futures Markets Using Genetic Programming. Journal of Futures Markets, Vol. 20, no. 10, 911–942