Hedging Derivative Securities with Genetic Programming

Shu-Heng Chen[1]*, Wo-Chiang Lee[2] and Chia-Hsuan Yeh[1]

[1]AI-ECON Research Group, Department of Economics, National Chengchi University, Taiwan
[2]AI-ECON Research Group, Department of Finance and Banking, Tamsui Oxford University College, Taipie, Taiwan

ABSTRACT **One of the most recent applications of GP to finance is to use genetic programming to derive option pricing formulas. Earlier studies take the Black–Scholes model as the true model and use the artificial data generated by it to train and to test GP. The aim of this paper is to provide some initial evidence of the empirical relevance of GP to option pricing. By using the real data from S&P 500 index options, we train and test our GP by distinguishing the case in-the-money from the case out-of-the-money. Unlike most empirical studies, we do not evaluate the performance of GP in terms of its pricing accuracy. Instead, the derived GP tree is compared with the Black–Scholes model in its capability to hedge. To do so, a notion of tracking error is taken as the performance measure. Based on the post-sample performance, it is found that in approximately 20% of the 97 test paths GP has a lower tracking error than the Black–Scholes formula. We further compare our result with the ones obtained by radial basis functions and multilayer perceptrons and one-stage GP. Copyright © 1999 John Wiley & Sons, Ltd.**

Keywords: option pricing; Black-Scholes model; genetic programming; tracking error

MOTIVATION AND INTRODUCTION

Over the last few years, computational intelligence has been applied to the pricing of financial derivatives. For example, in artificial neural networks, applications can be found in Hutchinson, Lo, and Poggio (1994), Liu (1996), Lajbcygier *et al*. (1996), and Barucci, Cherubini and Landi (1997). As to genetic algorithms, a similar study has been conducted by Chen and Lee (1997a). Finally, in genetic programming, Noe and Wang (1997) and Trigueros (1997) pioneer this area of research.

Option is a financial contract that gives the right to buy (call option) or sell (put option) an asset for a specified price (the exercise price or the strike price) on or before a specified time. Option trading allows us to bet on future events and to reduce the financial risk. But what the contract is worth is anything but trivial. By using stochastic calculus, Black and Scholes (1973) established the cornerstone of modern option pricing theory.[1] The Black–Scholes model has led to many insights into the valuation of derivative securities. In brief, this model provides us with a formula for option pricing.[2] The basic variables included in the formula are:

* Correspondence to: Shu-Heng Chen, AI-ECON Research Group, Department of Economics, National Chengchi University, Taipei, Taiwan 11623.
E-mail: chchen@nccu.edu.tw

- The strike price of the option (E)
- The time to maturity (τ)
- The current market price of the stock (S)
- The (instantaneous) risk-free interest rate (r_f), and
- The volatility of the stock price (σ).

However, the most challenging issue lies in the choice of the function form which can put these five variables well together. By making six assumptions, Black and Scholes (1973) provided the following formula for option pricing.

$$C = S\Phi(d_1) - Ee^{-r_f\tau}\Phi(d_2) \tag{1}$$

where C is the call price,

$$d_1 = \frac{\ln\left(\frac{S}{E}\right) + r_f\tau}{\sigma\sqrt{\tau}} + \frac{1}{2}\sigma\sqrt{\tau}$$

$d_2 = d_1 - \sigma\sqrt{\tau}$, and $\Phi(d)$ is the cumulative distribution function for the standard normal distribution. This formula can apply to the case of European-style call option. By its derivation, the success or failure of the Black–Scholes model crucially depends on whether any of its six assumptions is violated. Among these six assumptions, the one under active debate is the dynamic process of the underlying asset price. For example, the geometric Brownian motion assumption on asset price has been challenged by empirical evidence (Lo and Mackinlay, 1988). It is not surprising then that the Black–Scholes model has been shown demonstrating systematic biases as in much empirical research (Hull and White, 1987; Tucker, 1990; Hull, 1993; Rubinstein, 1985, 1994; White, 1996).[3] To avoid the empirical biases of the Black–Scholes model, nonparametric pricing methods, which do not rely on restrictive parametric assumptions, are developed, and techniques derived from computational intelligence are also involved.

Nonparametric pricing methods are highly data-intensive, requiring large quantities of historical prices to obtain a sufficiently well-trained networks, trees or chromosomes. According to the data used, the literature can be classified into two kinds. The first assumes that the Black–Scholes model is the true model and uses the artificial data generated by the B–S model to train and to establish a nonparametric model. Barucci, Cherubini and Landi (1997), Noe and Wang (1997), and Trigueros (1997) are this type of applications. However, as mentioned above, when the assumptions behind the B–S model no longer hold, it does not seem to make too much sense to establish our nonparametric model upon this 'straw man'. In this case, the second kind of application, which is based on real rather than artificial data, seems to be more appropriate. Hutchinson, Lo, and Poggio (1994) and Lajbcygier *et al.* (1996) are among the few of this camp. In the second type of application, real data, such as the S&P 500 futures, is employed to train learning networks.[4] The performance of these learning networks is then compared with that of the B–S model in the holdout sample. One of the frequently used performance measure is the *call price error*, i.e.

$$|C_M - C_{\text{model}}| \tag{2}$$

where C_M is the market call price and C_{model} is the call price predicted by the model in question, either the B–S model (C_{BS}) or a nonparametric model.

The problem with this measure is that C_M is assumed as the 'true' price and our trained model is designed to fit this 'true' price. However, for the reason given below, we hesitate to take C_M as the true price. The call price observed in the market usually is set by using the B–S model as a reference price. Therefore, while C_M can be different from the call price estimated by the Black–Scholes formula, they can be highly correlated. For example, according to Hutchinson *et al.* (1994), the correlation coefficient of C_M/E and C_{BS}/E can be high up 0.85 (E is the strike price).[5] In this situation, if C_{BS} is wrong, then C_M, using C_{BS}as a reference, may also be wrong. Therefore, using the call price error as a performance measure to guide machine learning may run the risk of 'following the herd', and not being able to discover the true direction. Certainly, applying machine learning in this style can hardly be considered novel.

Given the possibility that C_M may not be the true price, this paper will not take $|C_M - C_{\text{model}}|$ as the performance measure. Instead, we turn to the key argument for deriving the Black–Scholes formula, i.e. the no-arbitrage principle.

This is the application of the law of one price to financial assets, and forms the basis of the theory of option pricing. This principle is concretized by the famous Black–Scholes partial differential equation (PDE), which says that the instantaneous difference between the return on a hedged option portfolio and the return on a bank deposit should be identical 0.[6] Solving this equation leaves us an dynamic strategy for hedging, which enables us to monitor and rebalance the hedged option portfolio in a way such that the Black–Scholes PDE can hold. Therefore, if our option pricing formula is correct, then the associated dynamic hedging strategy should satisfy the Black–Scholes PDE. A slight modification of this observation gives us a notion of the tracking error introduced by Hutchinson *et al*. (1994). In this paper, we shall base our performance evaluation on the tracking error.

The rest of the paper is organized as follows. In the next section the tracking error is introduced. The data employed is described in the third section followed by the design of the two-stage genetic programming in the fourth section. The fifth section presents the experimental results, and the final section presents conclusions.

TRACKING ERRORS

Despite its computational convince, the measure $|C_M - C_{\text{model}}|$ is not ideal for telling us the practical value of any improvement in pricing accuracy that genetic programming might give us. As pointed out in Hutchinson *et al.* (1994), 'a more meaningful measure of performance for a given option pricing formula is the "tracking error" of various replicating portfolios designed to delta-hedge an option position, using the formula in question to calculate the hedge ratios or deltas' (p.868).

The idea of using tracking error as a performance measure of our model is based on the assumption of the no-arbitrage principle. By that principle, the expected value of a hedged option portfolio at the expiration date should be exactly zero. More specifically, suppose at date 0 we sell one call option and undertake the usual dynamic trading strategy in stocks and bonds to hedge this call during its life. If the option pricing model is correct, and if we can costlessly and continuously hedge, then at expiration the combined value of our stock and bond position should exactly offset the value of the call.

Mathematically, let $V(t)$ be the dollar value of the GP-based replicating portfolio at date t and let

$$V(t) = V_S(t) + V_B(t) + V_C(t) \tag{3}$$

where $V_S(t)$ is the dollar value of stocks, $V_B(t)$ the dollar value of bonds, and $V_C(t)$ the dollar value of call options held in the portfolio at date t. Moreover,

$$V_S(t) = S(t)Q_S(t) \tag{4}$$

where $S(t)$ is the stock price at date t, and $Q_S(t)$ is the shares of the stock held in the portfolio at date t.

Let us assume the initial composition of the GP-based portfolio at date 0 is assumed to be:

$$\alpha_0 = \begin{bmatrix} V_S(0) \\ V_C(0) \\ V_B(0) \end{bmatrix} = \begin{bmatrix} S(0)\Delta_{GP}(0) \\ -C_M(0) \\ -(V_S(0) + V_C(0)) \end{bmatrix} \tag{5}$$

where $C_M(0)$ is the market call option price, and $\Delta_{GP}(0)$ is the derivative of the GP pricing formula C_{GP} with respect to the stock price, i.e.

$$\Delta_{GP}(0) \equiv \frac{\partial C_{GP}(0)}{\partial S} \tag{6}$$

The portfolio position in $\alpha(0)$ represent the sale of one call option at date 0, priced according to the market price $C_M(0)$, and simultaneous purchase of $\Delta_{GP}(0)$ shares of stock at price $S(0)$. Since the stock purchase is wholly financed by the combination of riskless borrowing and proceeds from the sale of the call option, the initial value of the replicating portfolio is identically zero, and thus

$$V(0) = V_S(0) + V_B(0) + V_C(0) = 0 \tag{7}$$

Now, using C_{GP} to delta-hedge dynamically, we can have a continuous-time portfolio $\alpha_t (t \in R^+)$. However, since continuous-time delta-hedging is infeasible in practice, it has to be approximated by the discrete-time delta-hedging. In the following discussion, we, therefore, consider a sequence of portfolios α_{t_i} ($i = 0,1,2,\dots,n$). In other words, we divide the life of an option contract into n subintervals $\{[t_{i-1},t_i]\}_{i=1}^{n}$, where $t_n = T$. These intervals, as to be discussed later, are not necessarily equally spaced. Given the initial condition α_{t_0}, the GP-based portfolio α_{t_i} can be updated as follows:

$$\alpha_{t_i} = \begin{bmatrix} V_S(t_i) \\ V_C(t_i) \\ V_B(t_i) \end{bmatrix} = \begin{bmatrix} S(t_i)\Delta_{GP}(t_i) \\ -C_M(t_i) \\ e^{r_f(t_i - t_{i-1})}V_B(t_{i-1}) - S(t_i)(\Delta_{GP}(t_i) - \Delta_{GP}(t_{i-1})) \end{bmatrix} \tag{8}$$

where

$$\Delta_{GP}(t_i) = \frac{\partial C_{GP}}{\partial S}(t_i) \tag{9}$$

and $i = 1,2,\dots,n$.

$\Delta_{GP}(t)$ as defined above may not be computed analytically. Following Hutchinson *et al.* (1994), we approximate Δ by a first-order finite difference with an increment ∂S of size 1/1000 of the range of S,

$$\Delta_{GP}\left(\frac{S}{E},\tau\right) \approx \frac{C_{GP}\left(\frac{S}{E}(1+h),\tau\right) - C_{GP}\left(\frac{S}{E},\tau\right)}{\left(\frac{S}{E}\right)h} \tag{10}$$

where $h = 0.001$.

The tracking error of the replicating portfolio is then defined to be the value of the replicating portfolio $V(T)$ at expiration date T. From this, we obtain the following performance measure:

$$\xi \equiv e^{-rT}E[|V(T)|] \tag{11}$$

The quantity ξ is simply the present value of the expected absolute tracking error of the replicating portfolio. As mentioned above, if the option pricing formula is correct and all relevant assumptions are satisfied, then ξ should be zero, i.e. the difference between terminal value of the call and the terminal combined value of the stock and bond positions is identically 0. However, for the reasons listed below, ξ can be positive in practice.

- The option pricing formula employed is inaccurate.
- Even though the model is accurate, due to the discrete-time delta-hedging, ξ can be positive.[7]
- Third, since Δ_{GP} is numerically approximated by a first-order finite difference, this can also induce a positive ξ.
- Lastly, note that the ξ defined in equation (11) is an expectation value, which is unknown. In practice, it has to be estimated by the sample statistics, which are stochastic, and this creates another source of errors.

Since we expect to observe a positive ξ, we shall compare the GP tracking error with the tracking error of discrete delta-hedging under the exact Black–Scholes formula. Before we proceed further, note that the expectation of ξ is taken over the parameter space (T,E), and

$$\xi = 0 \tag{12}$$

implies

$$|V(T;E)| = 0, \ \forall\ T \text{ and } E \tag{13}$$

In other words, the conditional $\xi(\xi_{T,E})$ should also be 0 for all T and E. Equation (13) suggests that the performance comparison between GP and the Black–Scholes model can be made according to the following sample statistic:

$$\frac{\sum_{i=1}^{n_{T,E}} e^{-r_f T}|V(T;E)_i|}{n_{T,E}} \tag{14}$$

where $n_{T,E}$ is the number of the option contracts corresponding to the expiration date T and the strike price E, and $|V(T;E)_i|$ is the absolute tracking error of the ith contract. Equation (14) is the statistic we would like to calculate in this paper.

Data Description

The data used to train and to test genetic programming are daily closing prices of S&P 500 index options. The data is available from Chicago Board Options Exchange.[8] These are standard European-style options, for which exercise can occur only on the option expiration date, and their payoffs are determined by the level of the S&P 500 index on the option maturity date. Each CBOE index option contract represents \$100 (the index multiplier) times the current value of the index. For example, when the index is at 550, the underlying dollar value of 1 index option contract is equal to \$55,000 ($550 \times \$100$).

Each observation in this dataset is defined by list of variables shown in Table 1. Among these variables, C_M, S_t, E and τ can be directly downloaded from the CBOE dataset. The risk-free interest rate (r_f) is approximated by using the 3-month Treasure-bill annual rate (R_f) on the day of the initial activity in that option. Within this dataset, we focus only on the year 1995 and use the sample of January 1995 as the training sample. There are 9012 observations in the whole sample and 758 observations in the training sample. The basic statistics of these variables in the in-sample period (January 1995) is summarized in Table 2.

A subsample of the data of 1995 is chosen to compute tracking errors. This consists of 36 strike prices, ranging from 430 to 625, and four expiration dates on March, June, September and December.[9] Table 3 is a summary of this subsample. The first and the sixth column in Table 3 give the strike prices, other columns give the number of observations with respect to the expiration dates specified in the head of these columns. The letters M, J, S and D denote the months March, June, September and December. For example, read the cell (430, M); the number is 8, which means there are eight observations whose strike price is 430 and expiration date is March. These eight observations represent contracts initiated on eight

Table 1 Data description

Variables	Content
C_M	The daily closing price of S&P 500 stock index option
S_t	The daily closing price of S&P 500 on the trading day
S_T	The daily closing price of S&P 500 stock index on the expiration date
E	The strike price of the index option
τ	The time to maturity in year
R_f	The 3-month Treasure-bill annual rate

Table 2 Basic statistics of data

Variables	Max	Min	Mean	S.D.
$C_{M,j}$	46	0.0625	8.76	9.77
$C_{M,95}$	166.34	0	13.69	18.81
S_j	470.42	459.11	465.16	3.79
S_{95}	621.69	459.11	530.69	40.51
E_j	550	430	475	22.31
E_{95}	625	430	529	40.68
$\frac{C_M}{E_j}$	0.10	0.0001	0.019	0.021
$\left(\frac{C_M}{E}\right)_{95}$	0.36	0	0.026	0.037
$\frac{S}{E_j}$	1.09	0.83	0.98	0.04
$\left(\frac{S}{E}\right)_{95}$	1.36	0.82	1.00	0.05
τ_j	0.964	0.015	0.226	0.238
τ_{95}	0.964	0	0.176	0.211
$R_{f,j}$	5.82	5.55	5.71	0.07
$R_{f,95}$	5.89	5.18	5.54	0.17

The subscript ‘j’ denotes January, and ‘95’ the whole year of 1995.

Table 3 Strike price and expiration date

E	M	J	S	D	E	M	J	S	D
430	8	0	0	0	440	8	6	0	0
445	9	0	0	0	450	38	28	23	20
455	29	9	0	0	460	47	13	3	4
465	47	19	0	0	470	50	28	8	0
475	51	55	36	36	480	48	52	9	0
485	49	65	4	4	490	49	75	19	0
495	41	65	5	0	500	38	91	87	80
505	18	59	33	17	510	15	77	45	10
515	5	74	36	26	520	10	57	61	23
525	14	77	109	108	530	0	59	78	66
535	0	42	64	58	540	0	29	82	63
545	0	20	51	36	550	0	57	122	130
555	0	0	70	44	560	0	0	68	93
565	0	0	61	45	570	0	0	61	78
575	0	7	87	121	580	0	0	53	105
585	0	0	48	74	590	0	0	33	104
595	0	0	0	54	600	0	0	64	63
610	0	0	0	11	625	0	0	0	27

different dates. In this case, there are 9, 19, 23 January, 9, 17, 22, 27 February and 17 March. Also, the number shown in the cell (450, M) is 38, which means that there are 38 observations whose strike price are 450 and expiration date is March. These 38 observations are corresponding to the contracts initiated on 3, 4, 6, 9, 11, 12, 16, 17, 18, 19, 23, 24, 27, 30, 31 January, 1, 2, 3, 7, 9, 10, 13, 14, 15, 16, 17, 21, 22, 23, 28 February and 3, 6, 7, 8, 13, 14, 16, 17 March.[10]

While our subsample consists of only four expiration months, it occupies almost one half of the whole dataset. In Table 4, we show the number of observations corresponding to each expiration month. The ratios shown in the third row are the relative sizes of the number of contracts expired on these four months. For example, December has the largest number of contracts expired—a total of 1500, which contributes to 16.64% of all the contracts initiated in 1995. By contrast, March has the smallest number of contracts expired, only 574, 6.37% of the total contracts. The basic statistics of this subsample are also given in Table 2.

THE DESIGN OF GENETIC PROGRAMMING

Terminal Set and Function Set

Genetic programming is an automatic parallel search procedure conducted over a predefined space Ξ. Generally speaking, the search space Ξ is composed of computer programs. In genetic programming, there is a standard way to represent these computer programs, i.e., the LISP S-Expression.[11] For example, consider the Black–Scholes option pricing model as a computer program. The LISP S-Expression of this model is shown in Box 1.

The LISP S-Expression shown in Box 1 consists mainly of elements from two sets. The first is called the terminal set (Γ_1), and the second is called the function set (Γ_2). In the example above,

$$\Gamma_1 \equiv \{S/E, \sigma, \tau, r_f, \text{Constants}\} \tag{15}$$

and

$$\Gamma_2 \equiv \{+,-,\times,\div,\text{Exp},\text{Log},\text{Sqrt},\Phi\} \tag{16}$$

The LISP S-Expression can also be depicted as a parse tree. To see this, the parse tree of the LISP S-Expression of the Black–Scholes model is given in Figure 1. From the figure we can see that all the leaves of the parse tree are corresponding to elements of the terminal set (Γ_1), whereas all the roots of the parse tree are elements of the function set (Γ_2). We use this example to illustrate that the search space Ξ is in effect spanned by Γ_1 and Γ_2 in the sense that Ξ is a collection of all parse trees that can be reached syntactically by Γ_1 and Γ_2.[12]

$$\Xi = \text{SpanTree}(\Gamma_1, \Gamma_2) \tag{17}$$

Therefore, the first part of the design of genetic programming is to determine the elements in Γ_1 and Γ_2.

In the light of Black–Scholes option pricing theory, three variables are included in the terminal set (Γ_1), namely, time to maturity (τ), the risk-free interest rate (R_f) and the ratio of the stock price to the strike price of the option (S/E) (Table 5). Our terminal set is different from the one employed in Noe and Wang (1997) and Trigueros (1997) in that the latter does not take the risk-free interest rate into account. In terms of the closure property, the option pricing function they considered is:

$$\frac{C}{E} = f\left(\tau, \frac{S}{E}, 1\right) \tag{18}$$

Table 4 Number of contracts expired and its relative size.

Month	C	F	I	L	Total
N	574	1064	1420	1500	4558
Size	6.37%	11.81%	15.76%	16.64%	50.58%

Total number of contracts initiated in 1995 is 9012.

Box 1 The LISP S-Expression of the Black–Scholes Model

$(-\ (\times\ S/E\ (\text{NCDF}\ (\div\ (+\ (\ln S/E)$
$(\times\ (+\ r_f\ (\div\ (\times\ \sigma\ \sigma)2))$
$\tau)(\times\ \sigma\ (\sqrt{\ }\ \tau))))) \ (\times\ (\text{Exp}\ (\times\ -r_f\ \tau))$
$(\text{NCDF}\ (\div\ (+\ (\ln S/E)(\times\ (+\ r_f\ (\div\ (\times\ \sigma\ \sigma)2))$
$\tau))\ (\times\ \sigma\ (\sqrt{\ }\ \tau))))))$

Ignoring the risk-free interest rate r_f can be

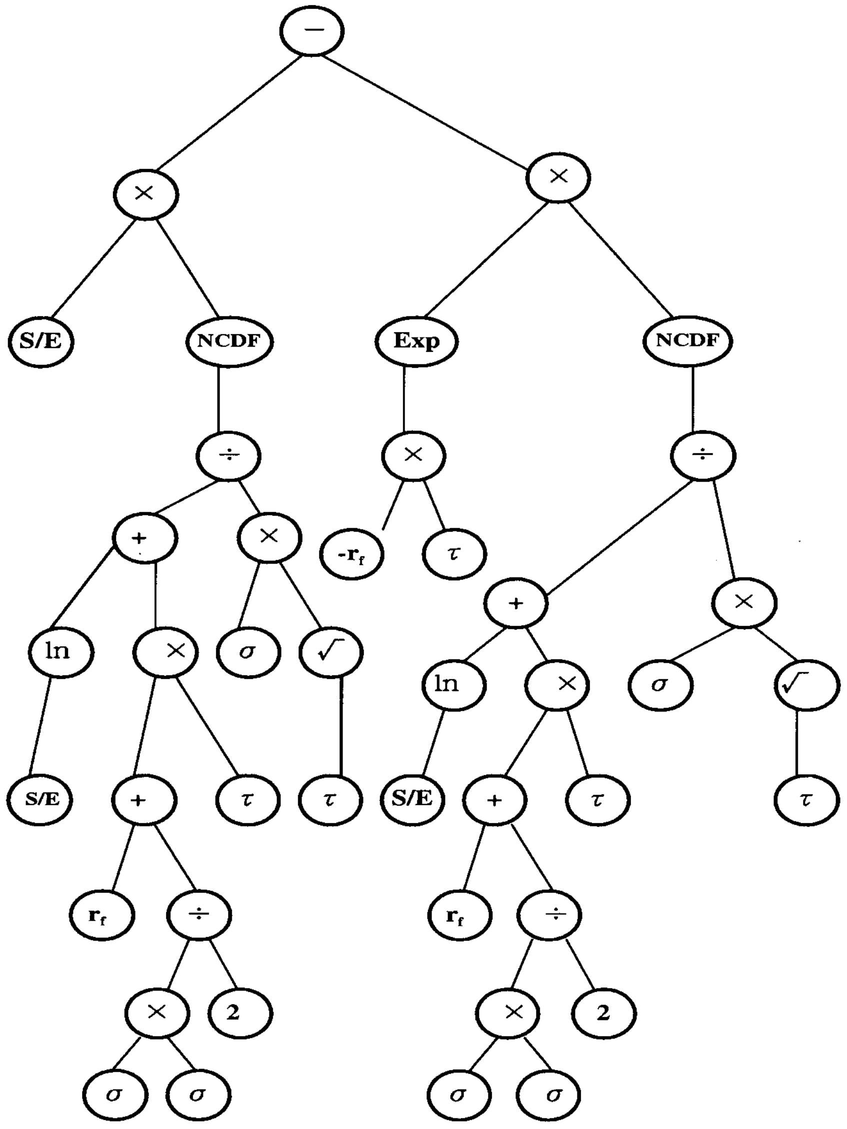

Figure 1 The parse tree of the Black–Scholes model

justified only if r_f is constant over the sample period. However, as we have seen from Table 2, R_f is not constant. Therefore, we decide to include it in our terminal set.

The same argument suggests that we should also include the variable volatility in the terminal set, for a preponderance of evidence points to volatility being time-varying (Bollerslev, Chou, and Kroner, 1992). But for the B–S model, the only input that is unobservable is the future volatility of the underlying asset. Therefore, to include it, the volatility input has to be modeled. However, modeling volatility is not a simple task and we are not ready to do it at this stage. We would leave this possibility to future research.[13] Furthermore, as usual, our terminal set also includes the ephemeral random floating-point constant R ranging over the interval [–9.99, 9.99]. By doing this, we are essentially using genetic programming

Table 5 Tableau for genetic programming

Population size (N)	500
Method of generation	Ramped half and half
Maximum depth for new individuals	6
Terminal set (Γ_1)	$\{\tau, R_f, S/E, R\}$
Function set (Γ_2)	{+,–,×,%,sin,cos,Exp,Rlog}
Truncation values	0, 0.2
Selection scheme	Proportionate selection
Criterion of fitness (F)	Sum of squared errors
Number of trees generated by reproduction	49
Number of trees generated by crossover	250
Number of trees generated by mutation	100
Number of new lives (immigrants)	100
Number of trees generated by elitism	1
Probability of mutation	0.0033
Mutation mode	Point mutation
Termination criterion	200-generation evolution
Maximum length of the tree	17
Probability of leaf selection under crossover	0.5
Maximum number in the domain of Exp	1700

R appearing the terminal set is the ephemeral random floating-point constant ranging over the interval [–9.99, 9.99].

to evolve constants. Whether this is an effective way to generate constants has been well noticed and discussed by the GP society. For example, Evett and Fernandez (1998) show that direct perturbation may help the discovery of numeric constants in genetic programming. This idea, while not being pursued in this paper, is certainly an interesting direction to attempt in the future study.

Our choice of the function set is also different from the one considered by Noe and Wang (1997) and Trigueros (1997), both of which are model-driven. In their studies, the B–S formula is assumed to be the true model and the goal is to test whether GP can discover this model. Our approach, in contrast, is data-driven and there is no reason to bind us to the B–S model. We therefore consider a very general function set (Table 5) and do not use those specific functions in the B–S model such as the normal cumulative distribution function $\Phi(.)$ used by Noe and Wang (1997) and Trigueros (1997). However, in future studies, we would like to see whether a larger terminal set may help. Another related choice of the function set is the truncation parameter. This parameter functions as follows:

$$\frac{C_{GP}}{E} = \begin{cases} \frac{C_{GP}}{E}, & \text{if } 0 \leq \frac{C_{GP}}{E} \leq 0.2, \\ 0.2, & \text{if } \frac{C_{GP}}{E} > 0.2, \\ 0 & \text{if } \frac{C_{GP}}{E} < 0 \end{cases} \tag{19}$$

The parameter values 0 and 0.2 are determined empirically by the data. In other words, from empirical data, the C–E ratio (the Call Price/Strike Price ratio) was between 0 and 0.2. (See Table 2 above.) We therefore explicitly take this factor into account.

GENETIC OPERATORS

Initialization Scheme

Once Γ_1 and Γ_2 are determined, genetic programming will generate a sequence of finite subsets of Ξ, say, $\Sigma_1, \Sigma_2, \cdots$ by applying the operation of Darwinian selection, crossover and mutation. Typically the cardinality (N) of each subset (population size) is exogenously given and is fixed throughout the entire generation process. In this paper, we set N to 500. This initial population of 500 tress is randomly generated. The initialization scheme we use is the 'ramped half and half' method detailed in Koza (1992). Under this scheme, equal numbers of trees are generated using a maximum initial depth that ranges from 2 to 6, so that 20% (100) of all initial trees are generated under the condition that the maximum depth is equal to 2, another 20% are generated under the condition that the maximum depth is equal to 3, etc. on up to a depth of 6. For each of the five maximum depth levels, 50% (50) initial tress are generated using the full method and the other 50% (50) are generated using the grow method.[14] Thus, under our initialization scheme, the full method is used to create one-half (250) of the initial trees and the grow method is used to create the other half (250) of the initial trees, as we have indicated in Table 5.

Selection Scheme

The dynamic process of Σ_t is driven by a series of genetic operations, each of which is run according to a specific selection scheme. Among many well-known schemes, we choose the proportionate selection scheme, also known as the roulette-wheel selection scheme. This scheme requires a probability measure over $\Sigma_t, p_{i,t}$, where $p_{i,t}$ is the probability that the ith tree at Generation t will be selected to generate Σ_{t+1}. Moreover, $p_{i,t}$ is positively related to the performance (fitness) of the parse tree $gp_{i,t}$, $gp_{i,t} \in \Sigma_t$. In other words, the better the parse tree performs, the more likely it can be influential on the generation of Σ_{t+1}.

The fitness criterion considered in this paper is a function of the inverse of the sum of squared errors (SSE), i.e.

$$f_{i,t} = \frac{1}{1 + SSE_{i,t}} \tag{20}$$

where

$$SSE_{i,t} = \sum_{k=1}^{K} \frac{(C_{M,k} - C_{i,k})^2}{K}. \tag{21}$$

$C_{M,k}$ is the kth observation of the true market price, and $C_{i,k}$ is the kth call price predicted by the ith parse tree. The probability measure, also known as the normalized fitness value, is then given by

$$p_{i,t} = \frac{f_{i,t}}{\Sigma_{i=1}^{N} f_{i,t}} \tag{22}$$

Unlike Noe and Wang (1997), we do not regularize the complexity of the evolved GP programs by adding any penalty term to the performance measure. In a noisy environment, this may run the risk of overfitting, but, as we shall indicate below, our GP does not tend to overfit.

Genetic Operations

The normalized fitness values $p_{i,t}$ are used to determine the next generation of parse trees $\Sigma_{i,t+1}$ from the current generation $\Sigma_{i,t}$ through application of the three primary genetic operators, i.e. reproduction, crossover, and mutation. We now describe these three genetic operators.

- *Reproduction* makes the copies of individual parse trees. The criterion used in copying is the normalized fitness value $p_{i,t}$. If $gp_{i,t}$ is an individual in the population Σ_t with the normalized fitness value $p_{i,t}$, it will be copied into the next generation with probability $p_{i,t}$. The operation of reproduction does not create anything new in the population and the offspring generated by reproduction constitute only part of the population Σ_{t+1}. As specified in Table 5, reproduction is performed on only 9.8% (49 out of 500) of the population. The rest of the offspring are generated by the other operators, such as crossover and mutation.
- The *crossover* operation for the genetic programming paradigm is a sexual operation that starts with two parental parse trees which are randomly selected from population Σ_t in accordance with the normalized fitness described above. Next, by exchanging the parts of these parents, two offspring are produced. This exchange begins by randomly and independently selecting one point in each parental parse tree using a uniform distribution described below.

  By the syntax of LISP, each point (atom) of a parse tree could be either a leaf (terminal) or a root (function). Therefore, the point (atom) selected could either be a leaf or a root. As specified in Table 5, the probability that the crossover point is a root or a leaf is the same, i.e. one half. Given that a root or a leaf is to be the point chosen for crossover, the probability that any root or leaf is chosen as the crossover point is uniformly distributed. For example, if the crossover point is to be a root, and then there are three roots in the parse tree, the probability that any one of the three roots is chosen for the crossover point is one-third (1/3). Unlike reproduction, the crossover operation creates new individuals in populations. As specified in Table 5, 50% (250 out of 500) of the new-generation population is created in this way.

- The operation of *mutation* also allows new individuals to be created. It begins by selecting a parse tree $gp_{i,t}$ from the population Σ_t based on $p_{i,t}$. Once a particular $gp_{i,t}$ is selected, mutation is a process of a random change of the value of a point (atom) within $gp_{i,t}$. Each point (atom) has a small probability of being altered by mutation, which is independent of other points (atoms). As specified in Table 5, the probability used throughout this paper is 0.0033. The altered individual is then copied into the next generation of the population. 20% (100 out of 500) of the new-generation population is created in this way.

Non-genetic Operations

The three operators combined create 79.8% of the population Σ_{t+1}. The rest of the parse tress are created by using the immigration operator and the elitism to be introduced below.

(1) *Immigration:* Another 20% of the offspring are immigrants that are created by randomly generating 50 parse trees. The motivation behind using the immigration operator is similar to that of using the mutation operator. They are all designed to equip the system with enough adaptability to avoid being trapped into a local optimum. The difference between mutation and immigration is that the latter will function completely independent of the ancestors while the former does not. Hence, immigration allows the system to be even more flexible.

(2) *Elitism:* Finally, the elitist selection is also introduced. This operation always save the best discovered solution from the previous generation. The use of elitism is mainly motivated by Rudolph (1996), who provides the conditions under which evolutionary algorithms with an elitist selection rule will converge to the global optimum of some function whose domain may be an arbitrary space.

Some Comments

The design of genetic programming is summarized in Table 5. There are some differences in the setup of the selection scheme and genetic operations between our paper and Noe and Wang (1997) and Trigueros (1997). For example, Trigueros (1997) does not have the mutation operator and Noe and Wang (1997) used the tournament selection scheme rather than the proportionate selection scheme. Needless to say, there are lots of variants one can play with. To encourage interested readers to have their own trial, the computer program used to conduct the experiments reported in this paper can be downloaded from the website: http://econo.nccu.edu.tw/ai/staff/csh/Software.htm

EXPERIMENT RESULTS

Motivated by Chen and Lee (1997b), we also divide the samples into two parts, namely, in-the-money ($S/E > 1$) and out-of-the-money ($S/E < 1$).[15] GP was first run eighteen times each of the two parts. Based on the MSE (Mean Squared Error) criterion, we choose the best performer in the training phase.[16] For the case in-the-money, the best one is Program 7, and for the case out-of-the-money, Program 3. By uniting these two programs together ($7 \cup 3$), the performance of genetic programming is compared with the Black–Scholes model in their performance of delta-hedging.

Since delta-hedging needs the market prices of call options (C_M), the intervals of discrete delta-hedging are designed in a way such that all values of C_M are observable. For example, consider the cell (430, M) in Table 3. There are eight observations in this cell, and the initiated dates of them are 9, 19, 23 January, 9, 17, 22, 27 February and 17 March. In this case, delta-hedging can only be performed on 19, 23 January, 9, 17, 22, 27 February and 17 March. Only in these seven days are C_M available, and the values of the replicated portfolio V_t can be evaluated in accordance with equation (8). Due to this data restriction, the frequency of delta-hedging is exactly the same as the number of observations in the cell (430, M). For convenience, we denote the number appearing in the cell (E, P) by $n_{E,P}$ ($P = M,J,S,D$). Also, with this data restriction, the delta-hedging interval is not constant. To see this, let us denote the eight market days mentioned above by $i_{430,M}$ ($i_{430,M} = 0,1,2,\ldots,7$). Then, in terms of τ (the time to maturity in year), it is easy to see that $\tau_{0_{430,M}} = 0.1944$, $\tau_{1_{430,M}} = 0.1626$, $\tau_{2_{430,M}} = 0.1547$, $\tau_{3_{430,M}} = 0.1031$, $\tau_{4_{430,M}} = 0.0793$, $\tau_{5_{430,M}} = 0.07142$, $\tau_{6_{430,M}} = 0.0595$, and $\tau_{7_{430,M}} = 0$. The difference between the two consecutive values of τ,

$$|\tau_{(i-1)_{430,M}} - \tau_{i_{430,M}}|$$

gives the following six lengths of the delta-hedging interval, namely, 0.0317, 0.0079, 0.0515, 0.0238, 0,00794, 0.0119, and 0.0595. Tracing the data sequence in this manner, the delta-hedging intervals for other cells (E,P) is defined. Applying equation (11) to the hedging intervals of each cell (E,P), we can calculate ξ_E as:

$$\hat{\xi}_{E,P} = e^{-R_f\%(\tau_{0_{E,P}} - \tau_{n_{E,P}})}|V(\tau_{0_{E,P}} - \tau_{n_{E,P}}; E)| \tag{23}$$

Given equation (23) above, the delta-hedging error of GP and the Black–Scholes model, $\hat{\xi}_{E,P}$, is computed for all Es and Ps available in Table 3. The results are given in Tables 6 and 7.

To make these results more readable, we divide the $\hat{\xi}_{E,P}$ obtained by the GP program '7 ∪ 3' by the one obtained by the Black–Scholes model, i.e.

$$\rho_{E,P} = \frac{\hat{\xi}_{E,P}|_{7\cup 3}}{\hat{\xi}_{E,P}|_{BS}}. \tag{24}$$

If $\rho_{E,P}$ is less than 1, then it means that GP outperforms the B–S model in terms of delta-hedging at the specified strike price and expiration month. On the other hand, it means that the B–S model is superior to GP in delta-hedging. The results of ρs are exhibited in Table 8. From Table 8, we can see that out of 97 tracks of options, GP can outperform the B–S model in 18 cases. Approximately, this empirical study shows that GP can have 20% chance to beat the B–S model. How impressive or significant is this result? To answer this question, it would be quite useful to look at the empirical results established by other studies.

In the literature, there are only two studies producing delta-hedging errors, one is Trigueros (1997) and the other is Hutchinson *et al.* (1994). Trigueros (1997) applied the same technique, genetic programming, but did not distinguish the case in-the-money from the case out-of-the-money. The data employed by Trigueros is the simulated B–S data rather than the real data. In his simulation, the parameter volatility was set to be 0.2., and the riskless rate was held constant at 0.2 and 0.05 respectively.[17] With these and other few parameters, a path $\{S_t\}$ with $S_0 = 50$ is simulated through 24 21-day months, i.e. 504 observations. CBOE rules as explained in Hull (1993) were used to create C_{BS}s and Es as stock price moved and existing options expired. The training set size ranged between 5000 and 7000 points. But only 2% of were actually used. By this procedure, the highest winning rate obtained in Trigueros is 17.6%.

Following a similar procedure, while with a much larger size of training sets, Hutchinson *et al.* (1994) found that the winning rate of their study is at best 38% for the radial basis function and is 28% for the multilayer perceptrons, at the case $E = 50$ and $T = 0.25$.

In these two studies, since the data is simulated from the B–S model, the results may be biased toward the support of the B–S model. Therefore, these low winning rates may not be surprised. In the second part of Hutchinson *et al.* (1994), they also tried the real data. The data for their empirical analysis are daily closing price of S&P 500 futures and options for the 5-year period from January 1987 to December 1991. They divide the S&P 500 data into 10 nonoverlapping six-month subperiods for training and testing. The total number of data points per subperiod ranged from 4454 to 8301, with an average of 6246. They trained a separate learning network on each of the first 9 subperiods, and tested those networks only on data from the immediately following sub-

Table 6. Absolute delta-hedging errors: GP 7 ∪ 3

E	M	J	S	D	*E*	M	J	S	D
430	5.681	NA	NA	NA	440	3.676	12.160	NA	NA
445	3.295	NA	NA	NA	450	1.110	10.529	21.113	30.808
455	0.323	8.045	NA	NA	460	1.663	7.107	26.802	30.083
465	2.283	4.414	NA	NA	470	1.943	4.504	18.534	NA
475	1.055	1.865	9.664	19.626	480	1.844	3.166	17.080	NA
485	8.897	0.389	15.892	28.681	490	0.538	0.060	17.015	NA
495	1.561	1.817	15.086	NA	500	0.824	2.504	3.622	10.906
505	0.819	1.074	6.664	40.040	510	0.496	0.359	3.463	31.886
515	0.246	1.921	3.340	15.353	520	0.707	3.316	0.892	11.740
525	0.807	7.556	4.366	0.721	530	NA	2.634	1.758	12.060
535	NA	0.099	2.440	3.858	540	NA	2.275	4.977	3.348
545	NA	0.700	7.444	2.638	550	NA	0.448	3.414	2.802
555	NA	NA	12.920	6.057	560	NA	NA	6.944	1.136
565	NA	NA	1.532	0.523	570	NA	NA	8.999	3.042
575	NA	0.854	2.489	8.482	580	NA	NA	0.139	12.284
585	NA	NA	1.595	9.762	590	NA	NA	2.140	1.561
595	NA	NA	NA	0.482	600	NA	NA	3.313	4.478
610	NA	NA	NA	7.606	625	NA	NA	NA	6.642

Table 7. Absolute delta-hedging errors: Black–Scholes model

E	M	J	S	D	*E*	M	J	S	D
430	1.294	NA	NA	NA	440	0.113	2.330	NA	NA
445	0.347	NA	NA	NA	450	1.531	1.285	2.785	5.219
455	1.963	3.259	NA	NA	460	2.224	3.167	3.105	0.224
465	2.117	3.848	NA	NA	470	1.875	2.979	1.721	NA
475	1.198	3.420	5.743	7.312	480	0.297	1.373	0.464	NA
485	1.008	2.534	0.717	0.945	490	0.763	2.905	1.698	NA
495	1.054	3.340	1.744	NA	500	0.568	2.180	4.658	7.082
505	0.365	0.563	1.798	21.136	510	0.135	0.941	1.629	14.122
515	0.103	0.285	0.063	1.007	520	0.065	1.731	0.972	2.372
525	0.066	1.596	0.115	3.872	530	NA	2.235	0.836	6.802
535	NA	1.531	0.200	0.877	540	NA	0.947	4.010	2.300
545	NA	0.297	5.467	0.245	550	NA	0.208	0.578	1.862
555	NA	NA	5.913	0.025	560	NA	NA	4.115	1.656
565	NA	NA	1.582	0.435	570	NA	NA	5.887	1.817
575	NA	1.009	0.761	0.564	580	NA	NA	2.310	0.814
585	NA	NA	0.995	3.070	590	NA	NA	0.830	0.979
595	NA	NA	NA	0.004	600	NA	NA	0.184	0.909
610	NA	NA	NA	4.352	625	NA	NA	NA	4.949

period. Their Table XIX shows that the learning networks exhibit less hedging error than the estimated Black–Scholes formula in a substantial fraction of the options tested. The highest winning rate is 64.9% observed in the July to December 1990 testing period for the multilayer perceptrons, and the lowest winning rate is only 21.8% in the July to December 1991 testing period.

CONCLUSIONS

At present, existing studies are so limited and so variant that it is difficult to give a

Table 8. Absolute delta-hedging errors: (Genetic Programming Model/Black–Scholes Model)

E	M	J	S	D	*E*	M	J	S	D
430	4.389	NA	NA	NA	440	32.388	5.219	NA	NA
445	9.476	NA	NA	NA	450	**0.725**	8.194	7.579	5.903
455	**0.164**	2.468	NA	NA	460	**0.747**	2.244	8.630	134.303
465	1.078	1.146	NA	NA	470	1.036	1.511	10.766	NA
475	**0.881**	**0.545**	1.682	2.683	480	6.198	2.305	36.771	NA
485	8.819	**0.153**	22.162	30.327	490	**0.705**	**0.020**	10.016	NA
495	1.481	**0.544**	8.647	NA	500	1.451	1.148	**0.777**	1.540
505	2.244	1.907	3.706	1.894	510	3.671	**0.381**	2.125	2.257
515	2.376	6.728	52.273	15.240	520	10.852	1.914	**0.917**	4.947
525	12.166	4.733	37.807	**0.186**	530	NA	1.178	2.101	1.773
535	NA	**0.065**	12.155	4.398	540	NA	2.402	1.241	1.455
545	NA	2.356	1.361	10.763	550	NA	2.156	5.904	1.504
555	NA	NA	2.184	242.293	560	NA	NA	1.687	**0.686**
565	NA	NA	**0.968**	1.201	570	NA	NA	1.528	1.673
575	NA	**0.846**	NA	15.026	580	NA	NA	**0.060**	15.076
585	NA	NA	1.603	3.179	590	NA	NA	2.578	1.594
595	NA	NA	NA	102.6382	600	NA	NA	17.956	4.921
610	NA	NA	NA	1.747	625	NA	NA	NA	1.341

For all the cases which genetic programming beats the Black–Scholes model in hedging performance, the cell fields are emphasized by bold type.

thorough evaluation of our GP-based hedging portfolios.

First, in light of the Hutchinson *et al.* (1994), a test based on a single year seems to be too limited. We may say that the low winning rate obtained in this paper is due to the bad luck as what happened in Hutchinson *et al.* (1994), when they ran the test over the July to December 1990 testing period. Therefore, in the future, we would like to extend this study to cover a larger database. But, the side-issue about the instability in machine learning techniques should also be explored so that the end-user can be informed about when these machine learning techniques can be helpful.

Second, compared with the existing literatures, our results seems to perform better than those of Trigueros. But based on the limited experiments, we cannot be sure whether this improvement comes from the separation of the case in-the-money and the case out-of-the-money.

Finally, the use of machine learning technique in delta-hedging is still at a premature stage, and more empirical studies are needed to motivate useful designs so that we can drive this research toward more complicated derivatives, such as American options and exotic options, in the future.

Acknowledgements

This paper was presented at the Workshop on the Application of Machine Learning and Data Mining in Finance of the 10th European Conference on Machine Learning (ECML'98), Chemnitz, Germany, April 24, 1998. The authors are grateful to the workshop participants and one anonymous referee for the helpful comments. Research support from NSC grant No. 87-2415-H-004-001 is gratefully acknowledged.

References

Abelson H, Sussman GJ, Sussman J. 1985. *Structure and Interpretation of Computer Programs*, MIT: Cambridge, MA.

Barucci E, Cherubini U, Landi L. 1997. Neural networks for contingent claim pricing via the Galerkin method. In Amman H, Rustem B, Whinston A. (eds). *Computational Approaches to Economic Problems*. Kluwer: Amsterdam; 127–142.

Black F, Scholes M. 1973. The pricing of options and corporate liabilities. *Journal of Political Economy* **81**.

Bollerslev T, Chou R, Kroner K. 1992. ARCH modeling in finance: a review of the theory and empirical evidence. *Journal of Econometrics* **5**: 5–59.

Chen S-H. 1998. Modeling volatility with genetic programming: a first report. *Neural Net World* **8**, No. 2: 181–190.

Chen S-H, Lee W-C. 1997a. Option pricing with genetic algorithms: the case of European options. In Back T (ed.). *Proceedings of 1997 International Conference on Genetic Algorithms*, Morgan Kaufmann Publishers: San Francisco; 704–711.

Chen S-H, Lee W-C. 1997b. Option pricing with genetic algorithms: separating out-of-the-money from in-the-money. In *1997 IEEE International Conference on Intelligent Processing Systems* Vol.1, 110–115.

Chen S-H, Yeh C-H, Lee W-C. 1998. Option pricing with genetic programming. In Koza J, Banzhaf W, Chellapilla K, Deb K, Dorigo M, Foegl D, Garson M, Goldberg D, Iba H, Riolo R. (eds). *Proceedings of the Third Annual Genetic Programming Conference (GP'98).* University of Wisconsin, Madison, Wisconsin, 22–25 July 1998. Morgan Kaufmann Publishers: San Francisco; 32–37.

Evett M, Fernandez T. 1998. Numeric mutation improves the discovery of numeric constants in genetic programming. In Koza J, Banzhaf W, Chellapilla K, Deb K, Dorigo M, Foegl D, Garson M, Goldberg D, Iba H, Riolo R. (eds). *Proceedings of the Third Annual Genetic programming conference (GP'98)*. University of Wisconsin, Madison, Wisconsin, 22–25 July 1998. Morgan Kaufmann Publishers: San Francisco; 66–71.

Figlewski S. 1996. Finance, engineering, and financial engineering. The keynote speech given at the 1996 IEEE/IAFE Conference on Computational Intelligence for Financial Engineering, 24–26 March, New York City.

Houthakker HS, Williamson PJ. 1996. *The Economics of Financial Markets*. Oxford University Press: Oxford.

Hull J. 1993. *Options, Futures and Other Derivative Securities*, 2nd edn. Prentice Hall: Englewood Cliffs, NJ.

Hull J, White A. 1987. The pricing of options on assets with stochastic volatilities. *Journal of Finance* **42**: 281–300.

Hutchinson J, Lo A, Poggio T. 1994. A nonparametric approach to pricing and hedging derivative structure via learning networks. *Journal of Finance* **49**: 851–889.

Koza J. 1992. *Genetic Programming: On the Programming of Computers by Means of Natural Selection*, MIT: Cambridge, MA.

Lajbcygier P, Boek C, Flitman A, Palaniswami M. 1996. Comparing conventional and artificial neural network models for the pricing of options on futures. *NeuroVe$T Journal* **4**, No. 5: 16–24.

Liu M. 1996. Option pricing with neural networks. In Amari S-I, Xu L, Chan L-W, King I, Leung K-S 1996 (eds), *Progress in Neural Information Processing, Vol. 2*. Springer-Verlag: New York; 760–765.

Lo A, Mackinlay C. 1988. Stock market prices do not follow random walks: evidence from a simple specification test. *Review of Financial Studies* **1**: 41–66.

Noe TH, Wang J. 1997. The self-evolving logic of financial claim prices. Paper presented in the Third International Conference on Computing in Economics and Finance, Stanford, California, 30 June–2 July 1997.

Rubinstein M. 1985. Nonparametric tests of alternative option pricing models using all reported trades and quotes on the 30 most active option classes from August 23, 1976, through August 31, 1978. *Journal of Finance* **40**: 455–480.

Rubinstein M. 1994. Implied binomial trees. *Journal of Finance* **49**: 771–818.

Rudolph G. 1996. Convergence of evolutionary algorithms in general search space. *Proceedings of 1996 IEEE International Conference on Evolutionary Computation*, IEEE Press: New York; 50–54.

Tucker AL. 1990. *Financial Futures, Options and Swaps.* West: St Paul, M.N.

Trigueros J. 1997. A nonparametric approach to pricing and hedging derivative securities via genetic regression. *Proceedings of the IEEE/IAFE 1997 Conference on Computational Intelligence for Financial Engineering*, IEEE Press: New York; 1–7.

White H. 1996. The Lecture Note in the International Conference on Neural Information Processing (ICONIP'96), Hong Kong Convention and Exhibition Center, Wan Chai, Hong Kong, 24–27 September 1996.

[1]Had he not died of cancer in 1995, at the age of 57, Fischer Black would doubtless have shared the 1997 Nobel Prize with Myron S. Scholes and Robert C. Merton.

[2]For those who are not familiar with the basic option pricing theory, a quick overview can be found in Chen and Lee (1997a).

[3]The existence of pricing biases for the Black–Scholes model has been well documented. The best known bias is the *volatility smile*, which means that if only one volatility is used to price options with different strikes, pricing errors will be systematically related to strikes. The smile has also been shown to depend on options' maturities. For example, short-maturity out-of-the-money calls on equities have market prices much higher than the Black–Scholes model would predict.

[4]Options are traded on individual stocks, stock indexes, interest rate instruments, precious metals indexes, foreign currencies, and future contracts. The date used in Hutchinson *et al.* (1994) is daily closing prices of S&P 500 *futures options*, which should be distinguished from those of S&P 500 *index options* used in this paper.

[5]As Figlewski (1996) pointed out, whether the B–S model is the true model does not matter as long as people on Wall Street believe that it is the 'true' model.

[6]In plain language, it says that it is impossible to make an instantaneous riskless profit by buying and selling options and shares and at the same time borrow or lend money.

[7]A good example can be found in Hutchinson *et al.* (1994). On pages 870–871, they show that the RBF network pricing formula can yield a smaller ξ than the Black–Scholes formula, even though the latter is indeed the correct pricing formula.

[8]This is the only exchange specializing in options, and currently the leader in volume of trading. Originally part of the Chicago Board of Trade, it has become independent. For details, see the web page of CBOE: http://www.cboe.com/index/spx/spxbasic.html

[9]An option for any particular month has its maturity on the third Friday of that month. Each option is assigned to one of three maturity cycles. Options on some stocks at first had maturities on the January–April–July–October cycle, others on the February–May–August–November cycle, and the remainder on a March–June–September–December cycle. Because interest turned out to be concentrated on nearby maturities, however, the current month and the following month have been inserted into the list in each cycle. The readers who are unfamiliar with the basics of stock options is referred to Houthakker and Williamson (1996).

[10]This dataset is available upon request.

[11]For a nice introduction on the LISP language, the interested reader is referred to Abelson, Sussman and Sussman (1985).

[12]To avoid ill-defined behavior, some function modifications and restrictions are necessary. For example, the division (%) and the logarithm function (Log) used are all protected. The protected division operator protects against division by zero by returning the value 1 if its denominator argument is 0; otherwise, it returns the value from dividing its first argument (the numerator) by its second argument (the denominator). Similarly, the protected natural logarithm function (RLog) avoids nonpositive arguments by returning the natural logarithm of the absolute value of its argument, and returning the value 0 if its argument is 0. The exponential function, which takes the argument x and returns the value e^x, allows a maximum argument value of 1,700 as indicated in Table 5. These types of modifications are quite standard in the GP literature. See, e.g., Koza (1992).

[13]While a model of volatility based on genetic programming has been proposed in Chen (1998), its advantages over existing volatility models are yet to be found.

[14]In the full method, the initial trees have the property that every path from root to endpoint is of full (i.e. maximum) depth. In the grow method, initial trees can be of various depths, subject to the constraint that they not exceed the maximum depth. See Koza (1992, pp. 92–93) for details.

[15]In our dataset, it happens that there is no observation of the case at-the-money ($S/E = 1$).

[16]The MSEs of these 18 runs can be found in Chen, Yeh and Lee (1998). There we find that the best performer in the training phase is also the best one in the testing phase in terms of the MSE criterion. The rank correlation results are all positive.

[17]In 1995, the historical volatility is 7.67%, and the interest rate ranges from 0.0518 to 0.0589.