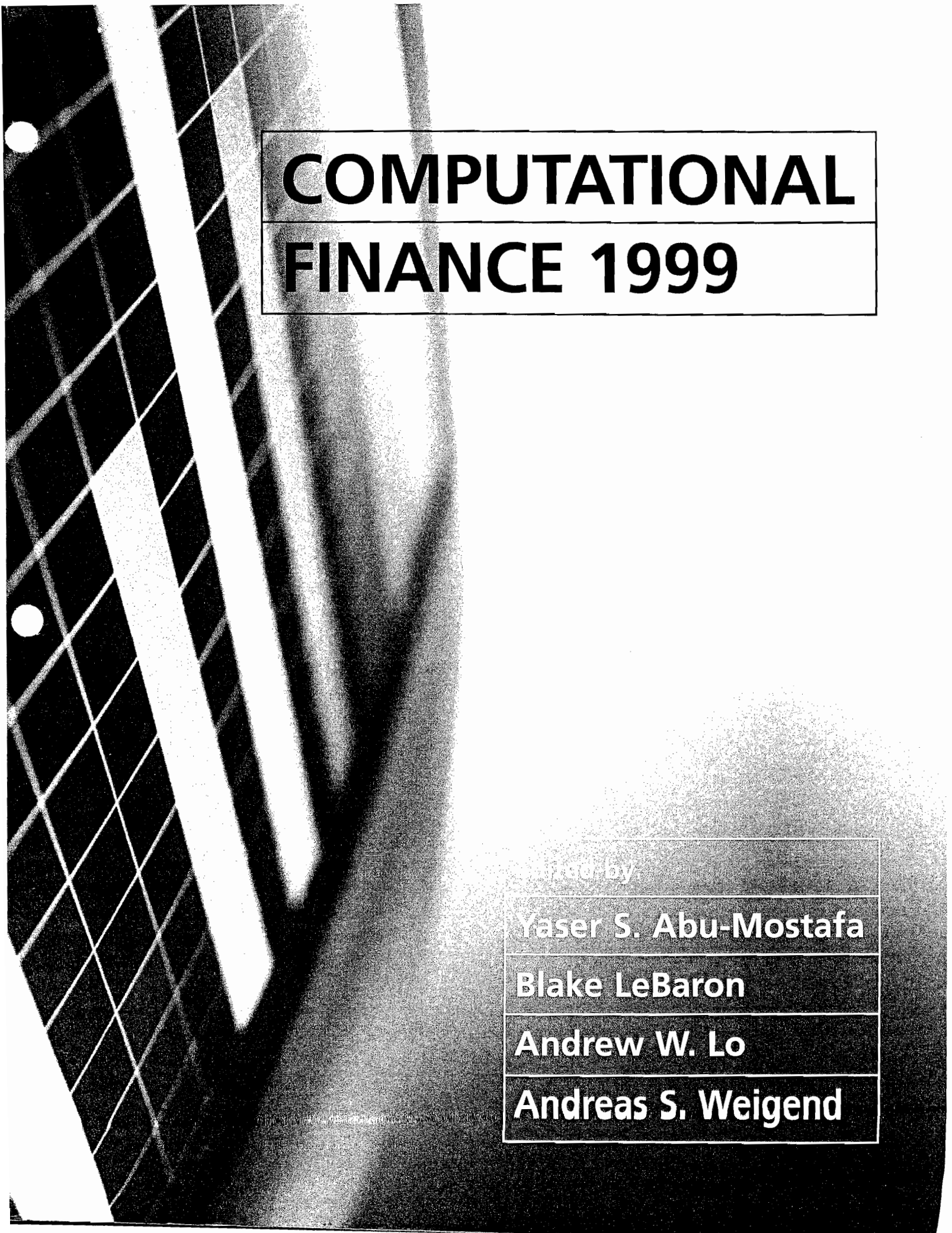

COMPUTATIONAL FINANCE 1999

edited by

Yaser S. Abu-Mostafa

Blake LeBaron

Andrew W. Lo

Andreas S. Weigend

38 Option Pricing via Genetic Programming

N. K. Chidambaran, Chi-Wen Jevons Lee, and Joaquin R. Trigueros

We propose a methodology of Genetic Programming to approximate the relationship between the option price, its contract terms and the properties of the underlying stock price. An important advantage of the Genetic Programming approach is that we can incorporate currently known formulas, such as the Black-Scholes model, in the search for the best approximation to the true pricing formula. Using Monte Carlo simulations, we show that the Genetic Programming model approximates the true solution better than the Black-Scholes model when stock prices follow a jump-diffusion process. We also show that the Genetic Programming model outperforms various other models when pricing options in the real world. Other advantages of the Genetic Programming approach include its low demand for data, and its computational speed.

The Black-Scholes model is a landmark in contingent claim pricing theory and has found wide acceptance in financial markets. The search for a better option pricing model continues, however, as the Black-Scholes model was derived under strict assumptions that do not hold in the real world and model prices exhibit systematic biases from observed option prices. We propose the Genetic Programming methodology for better approximating the elusive relationship between option prices, option contract terms, and properties of the underlying stock price.

Many researchers have attempted to explain the systematic biases of the Black-Scholes model as an artifact of its assumptions [Hull 1997]. The most often challenged assumption is the normality of stock returns[1]. Merton (1976) and Ball and Torous (1985) propose a Poisson jump-diffusion returns processes. French, Schwert and Stambaugh (1987) and Ballie & DeGennaro (1990) advocate GARCH [Bollerslev 1986] processes. It is not possible to derive closed-form analytical solutions in all cases, in which cases numerical solutions are used.

The difficulty in finding an analytical closed-form parametric solution has also led to non-parametric approaches. Rubinstein (1997) suggests that we examine option data for the implied binomial tree to be used for pricing options. Chidambaran and Figlewski (1995) use a quasi-analytic approximation based on Monte Carlo simulation. Hutchinson, Lo and Poggio (1994) build a numerical pricing model using neural networks. We apply Koza's (1992) Genetic Programming to develop an adaptive evolutionary model of option pricing. This method is well suited to the

1. Normality of stock returns has been repeatedly rejected. Indeed, Kim and Kon (1994) have ranked candidates for return distributions and found normality to be the least likely. Their rankings are: 1) Intertemporal dependence models (ARCH, GARCH), 2) Student t, 3) Generalized mixture of normal distributions, 4) Poisson jump, and 5) Stationary normal.

task and can operate on small data sets, circumventing the large data requirement of the neural network approach noted by Hutchinson, Lo, and Poggio (1994).

The philosophy underlying Genetic Programming is to replicate the stochastic process by which genetic traits evolve in offspring through a random combination of the genes of the parents, in the biological world. A random selection of functions of the option contract terms and basic statistical properties of the underlying stock price will have among them some elements that will ultimately make up the true option pricing formula. By selectively "breeding" the functions, these elements will be passed onto future generations of functions that price options more accurately.

Since it is impossible to ex-ante determine which element is the best, we instead focus on parent-selection, that is, the method of selecting equations to serve as parents for the next generation. Equations are chosen probabilisitically based on the pricing errors of the functions. We examine six alternative parent-selection methods: Best, Fitness, Fitness-overselection, Random, Tournament with 4 individuals and Tournament with 7 individuals. We find that the Fitness-overselection method seems to offer the best results for option pricing. We also explore the effect of varying other model parameters, such as the properties of the data set required to train the genetic programs, on model efficiency.

An important advantage of the Genetic Programming approach over other numerical techniques is its ability to incorporate known approximate solution into the initial "gene pool" to be used in evolving future generations, for example, we can include the Black-Scholes model. We illustrate how this approach quickly adapts the Black-Scholes model to a jump-diffusion process, where the Black-Scholes assumption of returns normality does not hold and for pricing options in the real world. We find that the Genetic Programming formulae beats the Black-Scholes equation in 9 out of 10 runs when the underlying stock prices are generated by a jump-diffusion process and in 10 out of 10 runs when we apply the analysis to the S&P Index options. The method also outperforms the Black-Scholes model for four out of five equity stock options in our sample.

38.1 Genetic Programming—A Brief Overview

Genetic Programming is a technique that applies the Darwinian theory of evolution to develop efficient computer programs[2]. We use a variant of Genetic Programming

2. Genetic Programming is an offshoot of Genetic Algorithms. Genetic Algorithms have been used to successfully develop technical trading rules by Allen and Karlajainen (1993) for the S&P 500 index and by Neely, Weller, and Dittmar (1997) for foreign exchange markets. In a paper similar

called Genetic Regression, where the desired program is a function that relates a set of inputs such as stock price, option exercise price, etc., to one output, the option price.

38.1.1 Basic Approach

The set of data on which the program operates to determine the relationship between input parameters and the options price is called the training set. The set of data on which the resulting formula is tested is called the test set. The procedure of the basic approach is as follows.

- Given a training set of matched inputs and outputs, a set of possible formulas is randomly generated. The formulas are represented as trees and we allow a maximum tree depth of 17[3]. Each formula is an individual and the set of individuals is called the population. The size of the population is held constant and is a control variable for optimizing the genetic program.
- Every individual in the population is evaluated to test whether it can accurately price options in the training data set. Based on individual fitness, a subset of the population is selected to act as the parents for the next generation of formulas.
- A pair of the parents generates a pair of offspring. Components of the parent formulas are crossed to generate offspring formula. A random point is selected in each parent tree. The sub-trees below that random point are switched between the two parent formulas. This operation creates a new pair of individuals, the offspring. It is possible that no crossover is performed and the parents themselves are placed in the new population (a clone). The process of selection and crossover is repeated until the new generation is completely populated.
- The steps above are repeated for a pre-specified number of times, or generations. Evolutionary pressure in the form of fitness-related selection combined with the crossover operator eventually produce populations of highly fit individuals. The best-fit individual is the solution to the option pricing problem.

---

in spirit to our study, Keber (1998) uses Genetic Programming to value American put options. Genetic Programming has also been used in multi-agent financial markets by Lettau (1997) and in multi-agent games by Ho (1996).

3. A 17 deep tree is a popular number used to limit the size of tree sizes Koza(1992). Practically, we chose the maximum depth size possible without running into excessive computer run times. Note that the Black-Scholes formula is represented by a tree of depth size 12. A depth size of 17, therefore, is large enough to accommodate complicated option pricing formulas and works well in practice.

38.1.2 Parent Selection Criteria

The method of selecting parents for the next generation can affect the efficiency of genetic programs. We analyze six different selection methods: Best, Fitness, Fitness-overselection, Random, Tournament with 4 individuals and Tournament with 7 individuals. These methods represent various attempts to preserve a degree of "randomnes" in the evolutionary process. In the Best method, individuals are ranked in terms of their fitness, ascending in order of the magnitude of their errors. The individuals with the smallest errors are thus picked to serve as parents of the next generation. In the Fitness method, individuals are selected randomly with a probability that is proportional to their fitness. In the Fitness-overselection method, 400 individuals are classified into two groups. Group 1 has 320 best-fit individuals and Group 2 has the remainder. Individuals are selected randomly with an 80% probability from Group 1 and a 20% probability from Group 2. In the Random method, the fitness of the individuals is completely ignored and parents are chosen at random from the existing population. Finally, in the Tournament method, n individuals are selected at random from the population and the best-fit individual is chosen to be a parent. We examine Tournament method with n=4 and n=7.

38.2 Performance Analysis in a Jump-Diffusion World

We first examine how well the Genetic Programming model can adapt and outperform the Black-Scholes model under controlled conditions. We choose a jump-diffusion world described by Merton (1976), since the closed form solution for the option prices in a jump-diffusion world is known and is a convenient benchmark. We can therefore measure the pricing errors of the Genetic Programming model and the Black-Scholes model.

The jump-diffusion process is a combination of a Geometric Brownian diffusion process and a Poisson jump process and can be written as:

$$\frac{dS(t)}{S(t)} = (\mu - \lambda k)dt + \sigma dW(t) + dq \tag{38.1}$$

where dq is the Poisson-lognormal jump process. The Poisson process determines when a jump occurs, with the jump size being lognormally distributed.

We simulate the price path of daily stock prices over a 24 month period with the initial price set at $S_0 = 50$. Each month is assumed to have 21 trading days. The diffusion parameters m (mean) and σ(standard deviation) were set at 10% and 20% respectively, and jump parameters k(jump size), λ (jump rate), and δ (standard

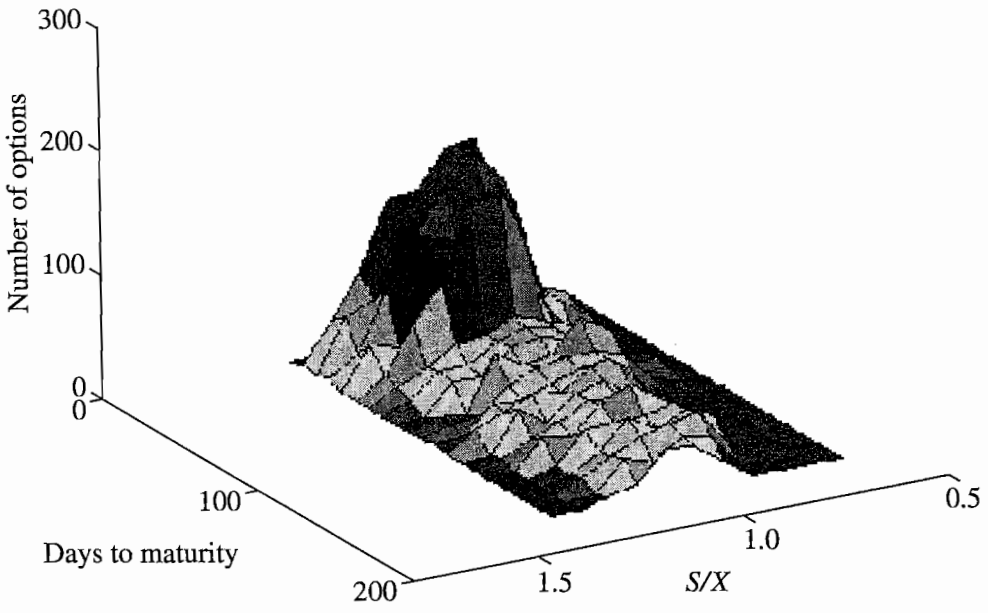

Figure 38.1
Distribution of options simulated in the jump-diffusion world

deviation of the log-jumps), were set at 0.02, 25, and 0.05 respectively. These values are well within the range estimated by stock price data. Thus 504 stock prices, $S(t)$, are simulated using random daily returns $z_t \sim N[(m - \sigma^2/2 - k)/252, \sigma/252]$ and $n(t) \sim Poisson[\lambda t]$ jumps, each of magnitude Y_j (where $lnY_j \sim N[ln(1+k) - 0.5\delta^2, \delta]$) for each $t \in 1 \ldots 504$:

$$S(t) = S_0 e^{\sum_{i=1}^{t} z_t}\ Y(n(t)), \quad t = 1, \ldots, 4 \tag{38.2}$$

where, $Y(0) = 1$ and $Y(n(t)) = \prod_{i=1}^{n(t)} Y_i,\ n(t) > 0$

We use CBOE rules to create call options from the simulated stock price path. Figure 38.1 shows the distribution of option prices in a jump-diffusion world.

Options are priced using Merton's (1976) jump-diffusion formula, truncated at the point when the marginal contribution of additional terms is negligible.

$$F(S, X, r, \sigma, \tau, \lambda, k, \delta) = \sum_{n=0}^{\infty} \frac{e^{-\lambda'\tau}(\lambda'\tau)^n}{n!} C_{B-S}(S, X, r_n, v_n, \tau) \tag{38.3}$$

where,$\lambda' = \lambda(1+k)$, $r_n = r - \lambda k + \frac{nln(1+k)}{\tau}$, $v_n = \sigma^2 + \frac{n\delta^2}{\tau}$, and τ is the option's time to maturity. C_{B-S} is the Black-Scholes option value given by,

$$C_{B-S}(S, X, r, \sigma, \tau) = SN(d_1) - Xe^{-rT}N(d_2) \tag{38.4}$$

Table 38.1
Genetic Programming Model Specification in a Jump-Diffusion World - Training Variables.

Name	Source	Definition
S	Option Contract	Stock price
X	Option Contract	Exercise price
S/X	Option Contract	Option moneyness
τ	Option Contract	Time to maturity (years)
Max(S-X)	Boundary Condition	Intrinsic value Max (S-X,0)
Black-Scholes	Analytical Model	Black Scholes value
+	Standard arithmetic	Addition
−	Standard arithmetic	Subtraction
*	Standard arithmetic	Multiplication
%	Standard arithmetic	x%y = 1 , if y = 0 = x/y , otherwise
Exp	Black-Scholes	Exponent: $exp(x) = e^x$
plog	Black-Scholes	$plog(x) = ln(\|x\|)$
psqrt	Black-Scholes	$psqrt(x) = sqrt(\|x\|)$
Ncdf	Black-Scholes	Normal CDF

where $d_1 = (ln[S/X] + (r + \sigma^2/2)T)/(\sigma\sqrt{T})$, $d_2 = d_1 - (\sigma\sqrt{T})$. $N[d_1]$ and $N[d_2]$ are the cumulative standard normal distribution values at d_1 and d_2.

Table 38.1 presents the set of operations and variables used to develop the Genetic Programs. Note that we include the Black-Scholes option value as a possible component of the tree. This serves as a good starting point and information from known analytical models can thus be used to find a better solution. We do correct for the volatility estimate an investor would have used based on a history of prices generated by a combination of the diffusion and jump processes. This reflects the approach of a naive investor who is unaware of the true nature of the returns' underlying process when using the Black-Scholes model to price the option. The estimated call option value with the modified Black-Scholes model is therefore (Merton 1976):

$$C_{AdjB-S} = C_{B-S}(S, X, r, \sigma_{adj} = \sqrt{\sigma^2 + \lambda\delta^2}, \tau) \tag{38.5}$$

Table 38.2 and Table 38.3 present the size of the training sets used and the algorithm training criteria. We implement an additional step in setting the size of the population and the number of generations. Using an independent set of 25% of the options as a training set, we determine that a minimum population size of 5000 functions and 10 generations is needed to get a formula that outperforms the Black-Scholes model. We use these parameters on ten new 25% subsets of the

Table 38.2
Genetic Programming Model Specification in a Jump-Diffusion World - Size of Training Sets. For each training set (option pricing formula), the price path of a stock with beginning value $S_0 = 50$ was simulated through 24 21-day months. Options were created according to CBOE rules and valued using the Black Scholes formula. Each training set consisted of the daily values of these options.

Training Set	1	2	3	4	5	6	7	8	9	10
Data Points	311	350	364	308	288	420	318	387	409	319

Table 38.3
Genetic Programming Model Specification in a Jump-Diffusion World - Training Parameters. Genetic Programming algorithm training parameters used in the non-Black-Scholes world where stock prices are generated by a jump-diffusion process.

Fitness Criterion	Sum of absolute dollar and percentage errors
Population Size	5,000
Number of Generations	10

options created to develop the genetic program for option pricing in a jump-diffusion world.

The formulas generated by the genetic program are adaptations of the Black-Scholes model, for example, one of the formulas generated is:

$$C(S,X,\tau) = \sqrt{C_{B-S} * \left[0.11734 + \sqrt{0.95461 * C_{B-S} * (C_{B-S} + \tau)}\right]} \tag{38.6}$$

where, C_{AdjB-S} is the adjusted Black-Scholes model given in Equation 38.5 and τ is the option's time to maturity.

We examine the performance of our genetic program on ten out-of-sample test sets of option data. The Fitness-overselection method gives the lowest pricing errors and beats the modified Black-Scholes model in each of the ten out-of-sample tests. The next best performance is the Tournament method with n=7. The Genetic Programming model based on Fitness-overselection clearly outperforms the Black-Scholes model in out-of-sample tests[4]. The Genetic Programming formula performs better than the Black-Scholes model for each of the 10 out-of-sample test sets.

An important criticism usually leveled at complex numerical methodologies is, Can the method perform any better than a simple linear regression model? We

4. The only measure in which the original Black-Scholes model ever beats the Genetic Programming formula is in the training-set sum of percentage errors, and this occurs for only 1 out of the 10 Genetic Programming formulas. However, the error is large enough to blow up the average percentage error. We attribute this fluke to our decision to ignore during training all percentage errors for options worth less than $0.01.

Table 38.4
Average Absolute Pricing Errors of the Genetic Programming models (GP), the Black-Scholes model (BS), and Linear Models in a Jump-Diffusion World. Pricing errors are presented for six Genetic Programming formulas using alternate methods for generating new populations from the previous generation and for four linear models that are a function of the initial stock price, exercise price, and time to maturity. Each cell in the table presents the average pricing errors over ten sets of stock and option prices and for the entire sample of options generated in each set. Parameter values used to generate stock price and options data and the Genetic Programming parameters are given in Table 38.1.

	GP	B-S	1-stage, with B-S	1-stage, no B-S	2-stage, with B-S	2-stage, no B-S
Best	0.0655	0.0888	0.0350	0.9357	0.0158	0.4049
Fitness	0.0517	0.0888	0.0350	0.9357	0.0158	0.4049
Fitness-Overselection	0.0393	0.0888	0.0350	0.9357	0.0158	0.4049
Random	0.0704	0.0888	0.0350	0.9357	0.0158	0.4049
Tournament, Size = 4	0.0534	0.0888	0.0350	0.9357	0.0158	0.4049
Tournament, Size = 7	0.0464	0.0888	0.0350	0.9357	0.0158	0.4049

therefore run single-stage and two-stage linear regression with and without Black-Scholes model as an independent variable. The two-stage model represents separate equations for in-the-money and out-of-the-money options.

Table 38.4 presents the pricing errors for the Genetic Programming formulas, the Black-Scholes equation, and for the linear models. The absolute and average errors for the Genetic Programming formulas are the average pricing error for all options which is again averaged across the ten Genetic Program runs. Results are presented for all six parent-selection methods considered. The modified Black-Scholes equation has the largest errors compared to all other models. The linear models give very good results when we include the naive Black-Scholes model as an independent variable with the two-stage linear model giving the lowest errors. Among the Genetic Programming formulas, the Fitness-overselection parent selection method provides the smallest absolute pricing error and one of the smaller percentage pricing errors. The magnitudes are comparable to the two-stage linear model with Black-Scholes as an independent variable[5].

5. The linear models that have the Black-Scholes model as an independent variable however have one major draw back – the partial derivatives of the pricing equation are equal to the Black-Scholes partial derivatives with a constant adjustment term. In a related paper (see Chidambaran, Lee, and Trigueros (1998)), we test the hedging effectiveness of the Genetic Program model and the Black-Scholes model by constructing a hedge portfolio of the Option, Stock, and Bonds. The

Table 38.5
Mean Absolute Pricing Errors of Genetic Programming models (GP), the Black-Scholes model (BS), and Neural Networks (NN) in a Jump-Diffusion world. The numbers in each cell are the average pricing errors from the models across 10 test sets. For each test set, the error value for each cell is calculated by taking the average pricing errors over five options. Rows in the table represent days-to-maturity and columns represent the degree-of-moneyness, S/X.

		Moneyness S/X					
Maturity		0.9	0.95	1	1.05	1.1	1.15
	BS=	0.05	0.08	0.03	0.03	0.01	0.00
5 days	GP=	0.03	0.02	0.12	0.06	0.01	0.01
	NN=	0.23	0.18	0.12	0.21	0.23	0.26
	BS=	0.08	0.09	0.00	0.04	0.01	0.00
10 days	GP=	0.03	0.02	0.08	0.07	0.03	0.02
	NN=	0.17	0.08	0.12	0.22	0.22	0.26
	BS=	0.13	0.12	0.07	0.02	0.00	0.01
30 days	GP=	0.03	0.03	0.02	0.03	0.05	0.04
	NN=	0.09	0.17	0.17	0.19	0.21	0.26
	BS=	0.15	0.14	0.10	0.06	0.03	0.01
45 days	GP=	0.04	0.04	0.03	0.02	0.03	0.04
	NN=	0.17	0.18	0.16	0.18	0.22	0.26
	BS=	0.16	0.15	0.12	0.09	0.05	0.03
60 days	GP=	0.04	0.05	0.03	0.03	0.03	0.04
	NN=	0.19	0.16	0.15	0.19	0.21	0.26
	BS=	0.19	0.18	0.16	0.13	0.10	0.07
90 days	GP=	0.06	0.06	0.05	0.05	0.05	0.06
	NN=	0.14	0.14	0.18	0.20	0.21	0.28

We further evaluate the performance of the Genetic Programming formula by comparing its pricing errors with that of the Black-Scholes model and Neural Networks for options of various maturities and moneyness. The details of the Neural Networks we use are reported in Chidambaran, Lee, and Trigueros (1998). Results are reported for the network that gives the best results among the various normalization and initialization schemes considered.

Table 38.5 shows the absolute pricing errors and the percentage pricing errors for the Genetic Programming formula developed with Fitness-overselection, for the Black-Scholes model, and for the best Neural Network, on an out-of-sample two-dimensional grid. Each cell in the table represents the average across ten out-of-sample test data sets and the value in each cell is the average over 5 options.

hedging performance is calculated to be the deviation from zero in the portfolio value. Overall, the Genetic Program beats the Black-Scholes model in over 50% of the cases.

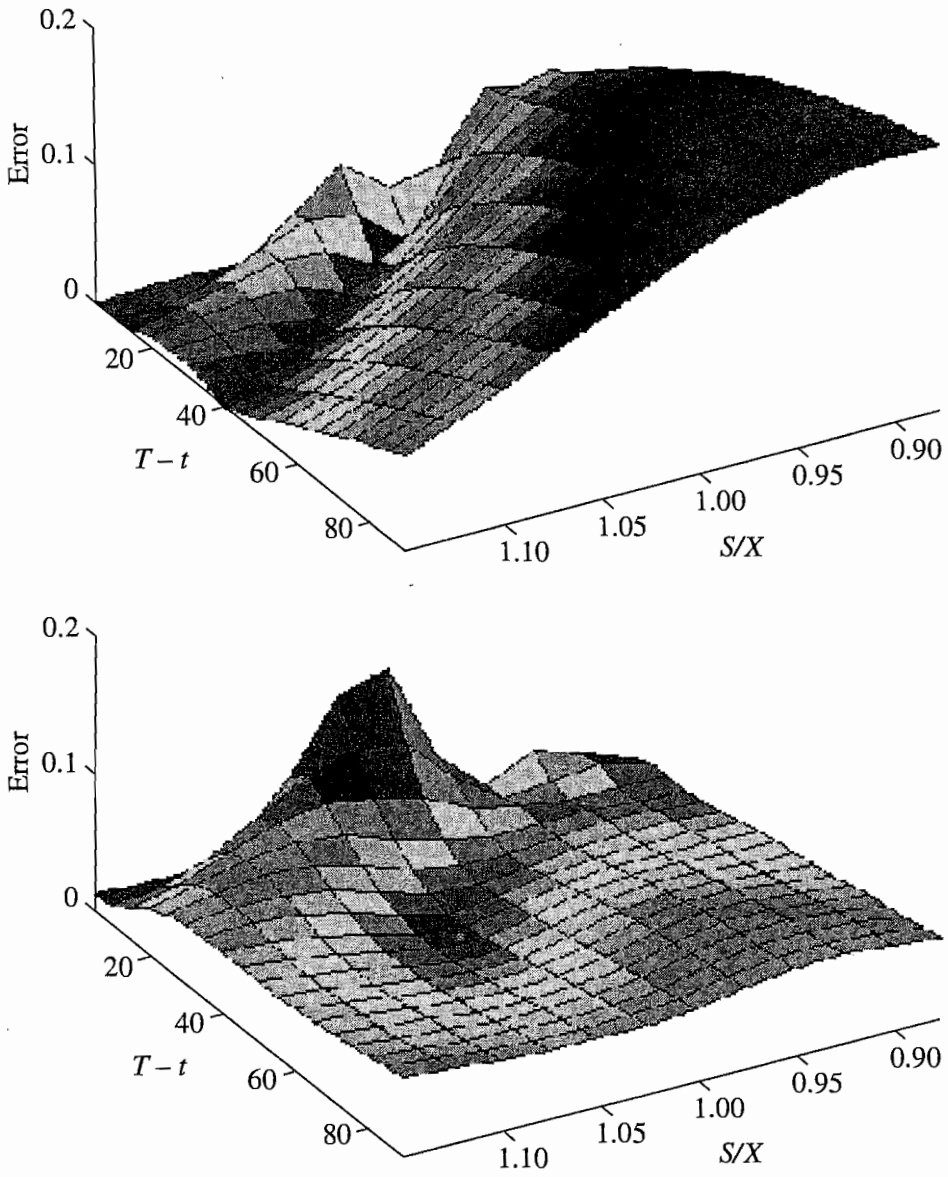

Figure 38.2
Mean absolute pricing error for the Black-Scholes (top figure) and Genetic Programming (bottom figure) models.

Figure 38.2 plots the absolute pricing errors for the Black-Scholes equation and the Genetic Programming formulas.

The Genetic Programming formula tends to do better with in-the-money and short-maturity options whereas the Black-Scholes model seems to perform relatively better with out-of-the money and long-term options. This result is consistent

with the notion that the jump term influences prices of short-maturity in-the-money options more, relative to long-term and out-of-the-money options. Genetic Programming beats Neural Networks overall, however in 9 out of the 72 cases considered Neural Networks do show marginally lower errors.

To take advantage of Genetic Programming's ability to learn with small training sets [Koza 1992] and reduce computational time, we tested its performance using random samples of 5% of the options generated in the simulation, without updating the algorithm parameters. We found that the training formulas with the smaller data sets resulted in only a minimal reduction in out-of-sample performance (see Chidambaran, Lee, and Trigueros (1998)). Our tests support the notion that Genetic Programming needs only small training sets in order to arrive at a good solution.

38.3 Application in the Real World

We next apply Genetic Programming to price real-world options data. Call options data for the S&P 500 Index and 5 different stocks were obtained from the Berkeley Options Data Base (BODB). BODB's data is time stamped to the nearest second and ensures a good match between the values of an option and its underlying asset[6]. Option prices are set to be the average of the bid and ask prices. The option's time to maturity is set to be the number of trading days between the trade date and the expiration date of the option. Interest rates from the term structure of zero-coupon treasuries (Bliss 1997) were used to calculate the risk-free rate between two calendar dates.

We use a two step process to develop the Genetic Program. We first determine the optimal set of algorithm parameters using a training and validation step. That is, we vary algorithm parameters when training the program and test the performance on a validation data set of options prices from a later date. The parameters that give the best results in the validation step are the optimal algorithm parameters. These parameters are then used in the next stage when the genetic programs are

6. Raw BODB records are screened as follows. We do not include records from the first 2,500 seconds after 8:30 am or in the last 2,500 seconds before 3:00pm and required at least 300 seconds within a 1% deviation for the underlying index/equity price. We also reject data when the option bid-ask spread is greater than $0.25 or is greater than 5% of option value. The first restriction eliminates artificial pricing that may occur due to the structure of the market at the beginning and the ending of the day. The second restriction is to allow the options market to adjust to changes in underlying asset value. The third gives us a tighter handle on the option's equilibrium price.

developed. A separate training data set and out-of-sample data set of options prices are used for the subsequent training/test step.

Ten Genetic Programming formulas were developed using ten sets of training/validations sets and ten sets of training/test data sets. The training/validation data sets were created by randomly sampling April 3-4, 1995 screened S&P 500 Index Option data. The training/test data sets are separately created from April 6-10, 1995 screened S&P 500 Index Options data. All out-of-sample validation and test data occurred later in time than the training data. Training sets contained a mere 50 points each and training time do not exceed 3 minutes per formula.

The sets of operations, functions and variables allowed in our formulas are those used in the jump-diffusion world (given in Table 38.1), augmented by the risk-free rate and historical volatility. As in Hutchinson, Lo, and Poggio (1994), we estimate the S&P 500 Index volatility by computing the standard deviation of the 60 most recent continuously compounded daily S&P 500 returns using 3.00pm (CST) prices. We adjusted for dividends by subtracting the present value of actual dividends between the record date t_0 and the option maturity date T.

The formulas generated by the genetic program for the index options were adaptations of the Black-Scholes model. For example, one formula is,

$$C(S, X, \tau) = C_{B-S} + 3\tau \tag{38.7}$$

where, C_{B-S} is the Black-Scholes formula and τ is the option's time to maturity.

Table 38.6 presents the average absolute pricing error and the average percentage pricing error for the 10 Genetic Programming formulas and for the Black-Scholes model when pricing the S&P Index options. The out-of-sample performance of these Genetic Programming adaptations of Black-Scholes model is remarkable: 9 out of the 10 Genetic Programming formulae beat the Black-Scholes model in both average absolute and percentage pricing errors.

For testing the performance of Genetic Programming in pricing equity options, we choose five stocks that had options volume of at least 1500 contracts and which never paid cash dividends. The stocks are: Best Buy Company Inc., Broderbund Software Inc., CompUsa Inc., Digital Equipment Corporation, and Novellus Systems Inc. For each stock, we develop ten Genetic Programming formulas. The training/validation data sets are constructed using BODB records for April 3-4, 1995. The training/test data sets are constructed from options traded during the period April 6-13, 1995.

The formulas generated by the genetic program for the equity options were also adaptations of the Black-Scholes model. In most cases, the formulas are of the form,

$$C(S, X, \tau) = C_{B-s} + Constant * \tau \tag{38.8}$$

Table 38.6
This table shows the mean absolute pricing errors for ten Genetic Programming Formulas, the Black-Scholes model, and Neural Networks, on ten out-of-sample data sets of the S&P 500 Index (SPX) option and five equity options. Each formula came from a separate training set and was evaluated on a separate test set, each set having a sample of 50 options. The parameter search was performed using April 3-4 data to find algorithm parameters that give formulas with good out-of-sample performance. All training and test sets for SPX came from April 6-10 BODB data and those for five equity options came from April 6-13.

Run #	GP	B-S	NN	GP	B-S	NN
		S&P 500			Best Buy	
1	1.9555	3.2301	1.2371	0.0927	0.1181	0.0583
2	3.0323	4.6736	1.1773	0.0605	0.0859	0.0633
3	2.3617	3.3248	1.2008	0.1150	0.0794	0.0771
4	1.5177	3.2884	1.0456	0.0996	0.1019	0.0583
5	2.4803	3.5442	2.0528	0.1262	0.0868	0.0743
6	2.0220	3.0111	0.7804	0.1046	0.0895	0.0621
7	2.3358	3.1802	1.6938	0.1012	0.1145	0.0527
8	2.1766	3.3159	1.2111	0.1189	0.0786	0.0701
9	1.6305	3.2846	1.0720	0.0855	0.1011	0.0601
10	3.0803	4.1809	0.9376	0.1181	0.0780	0.0766
Average	2.2590	3.5030	1.2409	0.1022	0.0933	0.0653
		Broderbund			Comp USA	
1	0.1346	0.1346	0.7407	0.1475	0.1892	0.1640
2	0.1255	0.1255	0.6099	0.1919	0.1856	0.1396
3	0.1074	0.1074	0.5695	0.2145	0.1487	0.2774
4	0.1418	0.1415	0.5975	0.1622	0.1870	0.1395
5	0.1448	0.1448	0.6821	0.1592	0.1656	0.1620
6	0.1279	0.1279	0.6916	0.2133	0.1843	0.2652
7	0.1253	0.1253	0.6114	0.1665	0.1856	0.1211
8	0.1211	0.1211	0.5610	0.1502	0.1886	0.1583
9	0.1475	0.1475	0.7252	0.1796	0.1922	0.1453
10	0.1074	0.1158	0.6349	0.1519	0.1870	0.1724
Average	0.1283	0.1291	0.6424	0.1737	0.1814	0.1745
		DEC			Novellus	
1	0.1595	0.1595	0.4301	0.1423	0.2531	0.4756
2	0.1462	0.1462	0.4140	0.1976	0.2840	0.5227
3	0.1435	0.1435	0.4544	0.2249	0.2850	0.4574
4	0.1148	0.1148	0.4056	0.1817	0.2692	0.5035
5	0.0880	0.1223	0.4336	0.1882	0.2772	0.4485
6	0.1254	0.1254	0.4076	0.2049	0.2344	0.4389
7	0.1236	0.1236	0.4302	0.1668	0.3217	0.4507
8	0.1286	0.1286	0.3329	0.2006	0.1979	0.5819
9	0.1454	0.1454	0.4343	0.1953	0.3096	0.5859
10	0.1063	0.1063	0.4105	0.1617	0.2973	0.4420
Average	0.1281	0.1316	0.4153	0.1864	0.2729	0.4907

where, C_{B-S} is the Black-Scholes formula and τ is the option's time to maturity. The constant takes values from 1 to 4 depending on the stock underlying the option that is being priced.

Table 38.6 also presents the average absolute pricing errors and average percentage pricing errors for the ten Genetic Programming formulas, the Black Scholes formula, and for Neural Networks, when pricing equity options. When there is no difference between the errors for the Genetic Programming formula and the Black-Scholes model, it indicates that the Genetic Program converged on the Black-Scholes model. Except for Best Buy, the Genetic Programming method produced formulas that outperform the Black-Scholes model on average, though the results are not as strong as the case of S&P Index options. We attribute the results for Best Buy to the fact that the data set used for the parameter search is much smaller than the data sets used for the other four stocks. The resulting training and validation sets were thus not independent enough to yield an insight on satisfactory parameters.

Neural Network pricing errors, though higher, are comparable to those of Genetic Programming formulas for the S&P 500 index option. On the other hand, Neural Network pricing errors are a magnitude higher than that of Genetic Programming formulas when pricing equity options. Equity options are more thinly traded and there is less data available to train Neural Networks and Genetic Programs in comparison to the S&P 500 index option.

Note that for Broderbund and DEC, a majority of the Genetic Programs converge on the Black-Scholes model as the best possible pricing formula. This highlights the advantage of the Genetic Programming approach – it can easily converge on existing known models, if they are indeed the best solutions. By including known analytical solutions in the parameter set, we thus increase the efficiency of Genetic Programming by using it to improve on existing solutions.

38.4 Conclusion

In this paper we have developed a procedure to apply the principles of Genetic Programming to option pricing. Our results, from controlled simulations and real world data, are strongly encouraging and suggest that Genetic Programming work well in practice.

Genetic Programming is a non-parametric data driven approach and, using options data, extracts the implied pricing equation. Genetic Programming thus overcomes the need to make specific assumptions about the stock price process. Re-

searchers have attributed the systematic biases in Black-Scholes prices to the assumption that stock prices follow a diffusion process. We show that Genetic Programming formulas beat the Black-Scholes model in 10 out of 10 cases in a simulation study where the underlying stock prices were generated using a jump-diffusion process. They work almost as well in pricing S&P Index options with genetic programs beating the Black-Scholes model in 9 out of 10 cases. For equity options, genetic programs beat or match the Black-Scholes model for 4 of the 5 stocks considered.

The Genetic Programming method requires less data than other numerical techniques such as Neural Networks (Hutchinson, Lo, and Poggio (1994)). We show this by simulation studies that use smaller subsets of the data and by using both genetic programs and neural networks to price relatively thinly traded equity options. The time required to train and develop the genetic programming formulas is also relatively short.

Genetic Programming can incorporate known analytical approximations in the solution method. For example, we use the Black-Scholes model as a parameter in the genetic program to build a better option pricing model. The flexibility in adding terms to the parameter set used to develop the functional approximation can also be used to examine whether factors beyond those used in this study, for example, trading volume, skewness and kurtosis of returns, and inflation, are relevant to option pricing. Finally, since the Genetic Programming method is self-learning and self-improving, it is an ideal too for practitioners.

References

Allen, F. and Karjalainen, R., 1999, "Using Genetic Algorithms to find technical trading rules," *Journal of Financial Economics,* Vol. 51(2), (February).

Ball, C.A. and Torous, W.N., 1985, "On jumps in common stock prices and their impact on call option pricing," *Journal of Finance,* Vol. 40 (March).

Ballie R. and DeGennaro, R., 1990, "Stock returns and volatility," *Journal of Financial and Quantitative Analysis,* Vol. 25 (June).

Black, F. and Scholes, M., 1972, "The valuation of option contracts and a test of market efficiency," *Journal of Finance,* Vol. 27 (May).

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

Bliss, R., 1997, "Testing term structure estimation methods", *Advances in Futures and Options Research,* Vol 9(1).

Bollerslev T., 1986, "Generalized Autoregressive conditional Heteroskedasticity," *Journal of Econometrics,* Vol. 31 (April).

Chidambaran, N. K., Lee, C. W. J., and Trigueros, J., 1998, "An Adaptive Evolutionary Approach to Option Pricing via Genetic Programming," Working paper, New York University.

Chidambaran, N. K. and Figlewski S., 1995, "Streamlining Monte Carlo Simulation with the Quasi-Analytic Method: Analysis of a Path-Dependent Option Strategy," *Journal of Derivatives,* Winter.

French, K. R., Schwert, G.W., and Stambaugh, R.F., 1987, "Expected stock returns and volatility," *Journal of Financial Economics,* Vol. 19 (September).

Ho, T. H., 1996, "Finite automata play repeated prisoner's dilemma with information processing costs," *Journal of Economic Dynamics and Control,* Vol. 20 (January-March)

Hull, J., 1997, *Options, Futures, and Other Derivative Securities,* 3rd Ed., (Prentice-Hall, Englewood Cliffs, New Jersey).

Hutchinson, J., Lo A., and Poggio, T., 1994, "A Nonparametric approach to the Pricing and Hedging of Derivative Securities Via Learning Networks," *Journal of Finance,* Vol. 49 (June).

Keber, C., 1998, "Option valuation with the genetic programming approach," Working paper, University of Vienna.

Kim, D. and Kon, S. J., 1994, "Alternative models for the conditional heteroscedasticity of stock returns," *The Journal of Business,* Vol. 67 (October).

Koza, J. R., 1992, *Genetic Programming,* (MIT Press, Cambridge, Massachusetts).

Lettau, M., 1997, "Explaining the facts with adaptive agents," *Journal of Economic Dynamics and Control,* Vol. 21.

Merton, R. C., 1973, "Theory of rational option pricing," *Bell Journal of Economics and Management Science,* Spring.

Merton, R. C., 1976, "Option pricing when underlying stock returns are discontinuous," *Journal of Financial Economics,* Vol. 3 (January-March).

Neely, C., Weller P., and Dittmar R., 1997, "Is Technical Analysis in the Foreign Exchange Market Profitable? A Genetic Programming Approach," *Journal of Financial and Quantitative Analysis,* Vol. 32(4), pp.405-426.

Rubinstein, M., 1997, "Implied Binomial Trees", *Journal of Finance,* Vol. 49.

Trigueros, J. 1997, "A Nonparametric Approach to Pricing and Hedging Derivative Securities Via Genetic Regression," Proceedings of the Conference on Computational Intelligence for Financial Engineering, March.