Proceedings of the 2003 Winter Simulation Conference
S. Chick, P. J. Sánchez, D. Ferrin, and D. J. Morrice, eds.

GENETIC PROGRAMMING WITH MONTE CARLO SIMULATION FOR OPTION PRICING

N. K. Chidambaran

Rutgers Business School – Newark & New Brunswick
Rutgers University
Piscataway, NJ 08854, U.S.A.

ABSTRACT

I examine the role of programming parameters in determining the accuracy of Genetic Programming for option pricing. I use Monte Carlo simulations to generate stock and option price data needed to develop a Genetic Option Pricing Program. I simulate data for two different stock price processes – a Geometric Brownian process and a Jump-Diffusion process. In the jump-diffusion setting, I seed the Genetic Program with the Black-Scholes equation as a starting approximation. I find that population size, fitness criteria, and the ability to seed the program with known analytical equations, are important determinants of the efficiency of Genetic Programming.

1 INTRODUCTION

Genetic Programming has been proposed for determining the relationship between the options prices and the parameters that are believed to be important in determining the price. This advantage of a non-parametric approach such as Genetic Programming is that it requires minimal assumptions and can easily adapt to changing and uncertain economic environments. The implementation of a Genetic Program for option pricing involves various programming parameters that determine the efficiency of the genetic program. In this study I explore the effect of these programming parameters in determining the efficiency of Genetic Programming.

Theoretical option pricing models based on risk-neutral pricing theory, such as the seminal Black-Scholes model, rely on strict assumptions that do not hold in the real world. The Black-Scholes model, for example, has been shown to exhibit systematic biases from observed option prices (Rubinstein 1977, Macbeth and Merville 1979, Macbeth and Merville 1980) and researchers have attempted to explain the systematic biases as an artifact of its assumptions. The most often challenged assumption is the normality of stock returns. 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. While closed-form solutions for the option price cannot be obtained for all these models, pricing formulas can be obtained numerically.

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. Chidambaran, Lee, and Trigueros (1999) propose Genetic Programming to develop an adaptive evolutionary model of option pricing that is also data driven and non-parametric. They show that this method offers some advantages over learning networks. In particular, it 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 equations 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 equations, presumably these elements will be passed onto future generations of equations that can price options more accurately. The essence of the method is the selection of equation components, i.e. genetic traits, which parents pass on to the next generation. Since it is impossible to determine which element is the best ex-ante, the focus is on choosing parents that seem to be the fittest. The genes to be propagated to the next generation are thus selected on the basis of the pricing errors of the equations.

There are many factors that determine the efficiency of Genetic Programming. Important specifications include the size of the population, the method of selecting equations with their embedded "genetic traits" to serve as parents, the number of mutations that are allowed, the size of the data set used for training the program. In this paper, I vary these parameters in exploring the efficiency of the Genetic Program.

An important advantage of the Genetic Programming approach over other numerical techniques is its ability to incorporate known approximate solution as a starting point for the approximation. That is, we can seed the initial population of equations with a particular equation or individual. This has two effects on the efficiency of the program. One, I start with an individual member in the population that gives a good fit to the data. Two, the elements of this equation will add to the "gene pool" to be used in evolving future generations. In this paper, I include the Black-Scholes model in the initial gene pool. This approach can reach the true pricing model more efficiently as it begins the search from a locally optimum solution. I 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.

The paper proceeds as follows. In Section 2, I introduce genetic programming and highlight its advantages over other non-parametric methods. In Section 3, I assess the ability of Genetic Programming in learning the Black-Scholes model, given data that are simulated according to the assumptions of the Black-Scholes world. In Section 4, I construct a non-Black-Scholes world and show how Genetic Programming can adapt the Black-Scholes model to its specifications. In Section 5, I conclude.

2 GENETIC PROGRAMMING – A BRIEF OVERVIEW

Genetic Programming is a technique that applies the Darwinian theory of evolution to develop efficient computer programs. In this section I describe the mechanics of the approach and the various ways to improve its efficiency.

2.1 Basic Approach

Genetic Programming is an offshoot of Genetic Algorithms. Genetic Algorithms have been used to successfully develop technical trading rules by Allen and Karlajainen (1999) for the S&P 500 index and by Neely, Weller, and Dittmar (1997) for foreign exchange markets. Genetic Programming has also been used in heterogeneous multi-agent economies by Marimon, McGrattan and Sargent (1990), in multi-agent financial markets by Lettau (1997), and in multi-agent games by Ho (1996).

I use a variant of Genetic Programming called Genetic Regression, where the desired program is a function that relates a set of inputs such as share price, option exercise price, etc. to one output, the option price. 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 described as follows.

- Given a problem to be solved and a training set of matched inputs and outputs, a set of possible formulas is randomly generated. These formulas are functions of some or all of the independent variables and randomly generated constants. 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 modeling process.
- Every individual in the population is evaluated to test whether it can accurately price options in the training data set. I assign a fitness measure to select the surviving gene. A smaller mispricing for the training data set indicates a better fit.
- Based on a fitness measure, a subset of the population is selected to act as the parents for the next *generation* of the population of formulas.
- A pair of the parents generates a pair of offspring. Components of the parent formulas are *crossed* to generate offspring formulas. 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 individuals in the new population are tested to gauge their performance in pricing options. 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 operation eventually produces populations of highly fit individuals. I keep track of the best-fit individual found throughout this process and set it as the solution to the option pricing problem.

2.2 Parent Selection Criteria

The method of selecting parents for the next generation can affect the efficiency of genetic programs. I examine 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 randomness in the evolutionary process.

In the Best method, individuals are ranked in terms of their fitness, ascending in the order of 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. I examine Tournament method with n=4 and n=7.

2.3 Advantages of Genetic Programming

An important advantage of Genetic Programming is its capability of incorporating a known analytical approximation to the solution into the program. In this paper, I include the Black-Scholes model as an initial parameter, i.e. part of the initial gene pool, for the algorithm. Since the method begins with a known approximation, it increases the probability of finding the true pricing formula and reduces computing time.

Genetic Programming requires smaller training sets than Neural Networks, which is a popular alternative adaptive learning algorithm (see Hutchinson, Lo, and Poggio (1994) and Koza (1992)). Since most options, especially those that are deep-in-the-money and deep-out-of-the-money, are thinly traded, Genetic Programming is an ideal tool for option pricing,

The methodology can also be made robust to changing environmental conditions and can operate on data sets generated over a range of possible conditions. I make the population robust by stochastically changing the training sets in the middle of the evolution. Only individuals with the desirable characteristics that are well adapted to changing environments will survive. The problem of over-fitting, in particular, is easily resolved by this approach. Further, new formulas can evolve out of previously optimal solutions when the data set contains structural changes rather than requiring retraining from scratch like in learning networks. Since genetic programs are self-learning and self-improving, they are an ideal tool for practitioners.

2.4 Convergence Characteristics of Genetic Algorithms and Programs

Our implementation of the Genetic Programming is effectively a search over the space of functions that can be constructed from a user-defined set of base variables and operations. This space of functions is generally infinite. However, the Genetic Programming algorithms are aided by the fact that I *limit* the search space and that the search is a *parallel* search.

I control and limit the complexity of the problem by setting a maximum depth size of 17 for the trees used to represent formulas. A 17 deep tree is a popular number used to limit the size of tree sizes Koza (1992). Practically, I 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 in practice.

The search space is, however, still very large and it is computationally inefficient to examine every possible tree. The implicit parallelism of Genetic Programming, however, ensures that the search is efficient. The central idea behind the parallelism of Genetic Programming is that each of the formula elements defines *hyperplanes*, i.e. subregions of the search space. In the population of candidate formulas, all the elements are present, and the fitness of each formula is a function of how many of the elements of the true pricing formula is present in the individual being evaluated. All formulas that contain a particular element will have similar errors and an evaluation of the formulas in the population is a parallel search for the hyperplanes containing the elements that make up the true option-pricing model. For example, the Black-Scholes formula is:

$$C = SN(d1) - Xe^{-r\tau}N(d2)$$

where,

$$d1 = [\ln(S/X) + (r + \sigma^2/2)\tau]/\sigma\sqrt{\tau} \quad \text{and} \quad d2 = \mathrm{d1} - \sigma\sqrt{\tau}\,.$$

N (d1) and N (d2) are the cumulative standard normal values for d1 and d2, S is the current stock price, X is the exercise price, r is the risk free rate, τ is the option time to maturity and σ is the volatility of the underlying stock. I can treat the formula to be the point at which the hyperplanes containing the term S N (d1) and -X $e^{-r\tau}$ N (d2) intersect. Searching over a randomly generated set of formulas is, therefore, a parallel search over a set of hyperplanes.

The true option pricing formula will consist of many different elements that form a set of hyperplanes and these is called its *schemata.* The individual sub-regions formed by the hyperplanes are the *schema*. If an individual equation contains elements that represent a superior region of the search space, it will generally be reflected as better fitness for the equation. This will increase the individual's chance to reproduce and pass on its schema to the next generation. When used to solve problems that involves a search for the sequence of elements that make up a gene, or any problem that involves a search for a sequence of numbers, Holland (75) and Koza (92) and show that the schemata of the Genetic Algorithm search process is extremely efficient and the algorithm converges. In this paper, I implicitly test whether such an approach will also work when searching for a closed-form option-pricing model.

3 GENETIC PROGRAMMING IN A BLACK-SCHOLES WORLD

In this section, I test the capacity of Genetic Programming to learn the Black-Scholes model, paralleling the study by Hutchinson, Lo, and Poggio (1994). Data to train the Genetic Programming is generated through Monte-Carlo simulation. For each data set, price paths of the underlying stock with initial value S_0 =50 are simulated for 504 days (24 months * 21 days/month). Stock returns are assumed to follow a diffusion process $dS(t)/S(t) = \mu dt + \sigma dW(t)$ with annual continuously compounded expected return μ=0.10, standard deviation σ=0.20 and risk-free rate r=0.05. Stock price at time t is calculated as:

$$S(t) = e^{\sum_{i=1}^{t} Z_t} \quad ; \quad t = 1,.., 504.$$

I next generate a sample of call options for each stock price realization. CBOE rules (Hull (1993)) were used to create call options with varying strikes and maturity for each day of the simulated price path. Option prices are derived for each simulated option, using the Black-Scholes equation. I thus have a sample of simulated options data. I adopted many of the simplifications suggested by Hutchinson, Lo, and Poggio (1994) in generating the data sample, for example, I hold the annual volatility σ and riskless rate r constant throughout. Figure 1 shows a stock price path generated by Geometric Brownian motion and Figure 2 shows the distribution of associated option prices.

Table 1 describes the specifications of the Genetic Programming model. I use the four basic mathematical operations, the log function, the exponential function, the square root function, and the cumulative Normal distribution. The basic division operation is protected against division by zero and the log and square root functions are protected against negative arguments. The current stock price, option exercise price, option intrinsic value, and option time-to-maturity are input parameters. The functional representation of a formula is assumed to be 17-step deep.

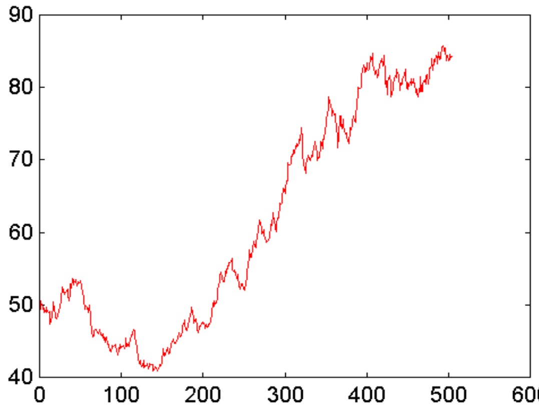

Figure 1: Sample Geometric Brownian Stock Price Process Using Monte Carlo Simulation

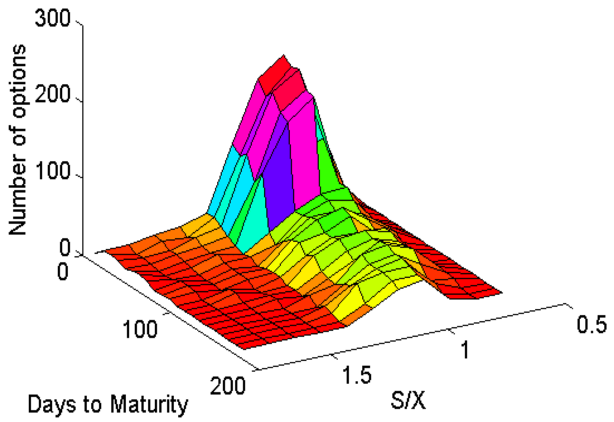

Figure 2: Distribution Of Option Prices Derived From The Stock Price Path In Figure 1

Table 1: Training Variables and Arithmetic Operations

Name	Source	Definition
S	Option Contract	Stock price
X	Option Contract	Exercise price
S/X	Part of Black-Scholes	Option moneyness
σ	Option Contract	Time to maturity (years)
max(S-X)	Boundary Condition	Option intrinsic value Max (S-X,0)
+	Standard arithmetic	Addition
-	Standard arithmetic	Subtraction
*	Standard arithmetic	Multiplication
%	Standard arithmetic	Protected Division: x%y = 1 if y = 0; = x/y otherwise
Exp	Black-Scholes component	Exponent: $\exp(x) = e^x$
Plog	Black-Scholes component	Protected Natural log: plog(x) = ln(|x|)
Psqrt	Black-Scholes component	Protected Square root: psqrt(x) = sqrt(|x|)
Ncdf	Black-Scholes component	Normal Cumulative Distribution Function

I implement ten trial runs, i.e. a Genetic Programming option pricing formula. Table 2 shows the genetic programming parameters used in each run. For each training set the price path of a stock with starting value S_0=50 was simulated through 24 21-day months as described earlier. 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. Formula popu-

Table 2: Genetic Programming Parameters

Fitness Criterion	Sum of absolute dollar errors and percentage errors
Population Size	100 - 50,000
Sample Size	5% - dynamically sampled
Number of Generations	100 - 1,000
Mutations	10%-50%

lations were exposed to dynamically sampled subsets, about 5% of the entire sample. The data set is stochastically changed in the middle of training run to prevent overfitting. I find that evaluating the population formulas on such stochastic subsets of the data set resulted in reduced training times and better out-of-sample performance. Only robust formulas can survive the constantly changing environment and pass on their "traits" to the next generation.

I varied population size from 100 to 50,000 individual formulas and varied the level of mutations between 10% and 50%. Each trial was run up to 1,000 generations.

The criterion for selecting the surviving formulas is a linear combination of the absolute pricing errors and the percentage pricing errors. I found (Chidambaran 2003) that the formulas consistently made relatively small absolute errors when pricing out-of-the-money options and relatively large absolute errors when pricing in-the-money options. The pattern in the magnitudes of the percentage error is just the opposite. Linear combination of these two error measurements leads to a more efficient selection rule.

For example, if the true price of an option is \$2.00 and one of the Genetic Programming formulas gives a price of \$2.20, then percentage error is small (10%) but dollar error is \$0.20, which is economically significant. On the other hand, if the true price is \$0.10 and our formula gives a price of \$0.07, dollar error is small (\$0.03) but the price is off by 30%. Our error measure is then 30 (10% + \$0.20) in the first case and 33 (30% + \$0.03) in the second case.

In the classic Genetic Programming fashion, I define the fitness of a formula to be:

$$\frac{1}{1+\sum_{i=1}^{\text{number of fitness cases}} \varepsilon_i}$$

where ε_i is the training error for the ith case. This training error is defined as the sum of percentage and dollar errors if the Black-Scholes value was greater than \$0.01 and just the dollar error if the Black-Scholes value was less than \$0.01.

It should be noted that the restrictions on Genetic Programming are far fewer than those required for Neural Networks. Only the variables needed for pricing options have to be specified. I need not make assumptions on the smoothness or complexity of the formulas beyond the maximum allowable depth (tree size) for representing a formula.

I measure the performance of Genetic Programming on an out-of-sample two-dimensional options grid of option maturities and strike prices. Each cell in the table is the average pricing errors across ten different Genetic Programming formulas.

My results indicate a pattern of performance of the Genetic Programming model consistent with Chidambaran, Lee, and Trigueros (1999). First, the dollar pricing errors are small for short-maturity options as opposed to long-maturity options. However, the percentage pricing errors are just the opposite. Obviously this is because the magnitude of option prices varies substantially across option maturities. Second, the errors vary across the option strike. Once again this is because option prices are very small for out-of-the money options and much higher for in-the-money options. The fitness criterion I use balances the two effects by minimizing a combination of the absolute and percentage pricing errors. I found that this allows me to better control the pricing errors for out-of-the money and in-the-money options without adversely affecting the errors for at-the-money options.

While all the parent selection methods yield similar qualitative results, they vary widely in efficiency. Results indicate that the Fitness-overselection method and the Tournament method (n=7) providing the best results and that Genetic Programming gives a good numerical approximation to the Black-Scholes model.

4 PERFORMANCE ANALYSIS IN A JUMP-DIFFUSION WORLD

The Genetic Programming approach can incorporate any known analytical approximation into its algorithm. It is flexible to adapt to changing and unknown economic environments. In this section, I illustrate how the Genetic Programming model can adapt and outperform the Black-Scholes model in a jump-diffusion world described by Merton (1976). Since the closed form solution for the option prices in a jump-diffusion world is available, I can measure the pricing errors from the Genetic Programming model and the Black-Scholes model in such a world. The performance analysis highlights the salient features of the Genetic Programming approach to option pricing.

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

$$dS(t)/S(t) = (\mu - \lambda k)dt + \sigma dW(t) + dq$$

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

I 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 μ (mean) and σ(standard deviation) were set at 10% and 20% respectively, and jump parameters *k*(jump size), λ (jump rate), and δ(standard deviation of the log-jumps), were set at 0.02, 25 and 0.05 respectively. This translates into 25 expected jumps per year, each inducing an expected percentage change of 2% on the stock price. The variance of the log-jumps is 0.05. These values are well within the range estimated by stock price data. Thus, 504 stock prices, S(t), are simulated using random daily returns $z_t \sim N((\mu-\sigma^2/2-k)/252, \sigma/252)$ and $n(t) \sim \text{Poisson}(\lambda t)$ jumps, each of magnitude Y_j (where $\ln Y_j \sim N(\ln(1+k)-0.5\delta^2, \delta)$), for each t in {1…504}:

$$S(t) = S_0 e^{\sum_{i=1}^{t} z_i} Y(n(t)),\ t = 1,..,504$$

where,

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

I use CBOE rules to create call options from the simulated stock price path. This ensures that there is a sufficient sample of at-the-money and near-the-money options at all points. The methodology however can result in only a few deep in-the-money and deep out-of-the-money options in the sample. I find that the performance of the genetic program is sensitive to the number of options at various moneyness levels in the sample.

Options are priced using Merton's (1976) jump diffusion formula given below. Terms in the sum increase and then decrease in magnitude due to the distribution of the attached probabilities. I truncated at the point when the marginal contribution of additional terms is negligible - all terms in the decreasing segment of the series whose marginal contribution was less than 0.00001% were dropped, as well as all terms beyond the 1000th in the sum.

$$F(S,X,r,\sigma,\tau,\lambda,k,\delta) = \sum_{n=0}^{\infty} \frac{e^{-\lambda'\tau}(\lambda'\tau)^n}{n!} f_n(S,\tau)$$

where,

$$\lambda' = \lambda(1+k)$$
$$f_n = Black - Scholes(S, X, r_n, v_n, \tau)$$
$$r_n = r - \lambda k + \frac{n\ln(1+k)}{\tau}$$
$$v_n = \sigma^2 + \frac{n\delta^2}{\tau}$$
$$\tau = \text{Time to maturity (T - t)}.$$

The set of operations and variables used to develop the Genetic Programs in the Jump Diffusion setting is the same as that for the Black-Scholes setting shown in Table 1.

The significant innovation in this step of the study is to seed the initial population with the Black-Scholes equation. This provides a good starting point for finding a solution and is a way in which I can adapt known analytical approximations to find a better approximation. I, however, correct for the volatility estimate that an investor would have calculated using a history of observed prices, which is a combination of the variances of the diffusion and jump processes. This reflects the approach of a naive investor who is unaware of the true nature of the underlying stock price process when using the Black-Scholes model to price option. The estimated call option value with the modified Black-Scholes model is (Merton 1976):

$$C = BS(S, X, r, \sqrt{\sigma^2 + \lambda\delta^2}, \tau).$$

I use the same population sizes, convergence criteria, and sample sizes in the jump-diffusion setting as I used in the Black-Scholes setting.

I also address an important criticism usually leveled at complex numerical methodologies. Can the method perform any better than a simple linear regression model? While linear regression of the options price on variables such as the options strike and current stock price can result in an equation that gives small errors within the sample, it is obvious that out-of-the sample option values will be priced with larger errors. It is, however, a useful benchmark. I, therefore, run single-stage and two-stage linear regressions with and without Black-Scholes model as an independent variable. The two-stage model represents separate equations for in-the-money and for out-of-the-money options.

The linear models that have, 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. The true test of any option pricing model is its performance in hedging and the constant adjustment to the option price sensitivity with respect to the stock price, the option *delta*, will not work in practice. Genetic Programming allows general adjustments to the option-pricing model and is not subject to this problem. Note that if the linear model is indeed the best model, the Genetic Program theoretically should be able to find it and no generality is lost.

I test the hedging effectiveness of the Genetic Program formula by constructing a hedge portfolio of the option, stock, and a riskless bond. The amount of stocks in the portfolio is chosen as usual to be the *delta* amount, where delta is determined by taking the first partial of the Genetic Programming formula with respect to the stock price. I estimate the performance of the hedge over ten samples of

100 paths for options of varying maturities and strike prices. The hedging performance in each path is calculated to be the deviation from zero in the portfolio value. I also similarly determine the hedging performance of the Black-Scholes model over the same 100 paths. My preliminary results indicate that the Genetic Program beats the Black-Scholes model in over 50% of the cases.

I 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. I chose Neural Networks for benchmarking results as it is the closest to genetic programming in its philosophy and is a data-driven non-parametric methodology. Other option pricing methods such as GARCH models require assumptions of the underlying process and parameter values. My preliminary results indicate that Genetic Programming beats Neural Networks in all cases.

To take advantage of Genetic Programming's ability to learn with small training sets (Koza 1992) and reduce computational time, I tested its performance using random samples of 5% and 25% of the options generated in the simulation. I found that the training formulas with the smaller data sets resulted in only a minimal reduction in out-of-sample performance. Our tests dramatically support the notion that Genetic Programming needs only small training sets in order to arrive at a good solution.

5 CONCLUSION

In this paper I have examine the impact of the various programming parameters in developing a Genetic Option Pricing Program. I find that program design, in particular population size, level of mutation, and sample size, is important in determining the accuracy and efficiency of the Genetic Program.

A careful specification of the programming parameters improves the efficiency of the Genetic Program. I find that smaller data sets that are stochastically changed leads to quicker convergence. I also find that larger population sizes and a higher level of mutations leads to quicker convergence of the program.

The Genetic Programming method has many advantages over other numerical techniques. First, it is a non-parametric data driven approach and requires minimal assumptions. I thus avoid the problems associated with making specific assumptions regarding the stock price process. The Genetic Programming method uses options price data and extracts the implied pricing equation directly.

Second, the Genetic Program can be seeded with the Black-Scholes equation in developing a better model. The ability to seed a Genetic Program is similar to choosing starting values using all available information in a standard optimization exercise. The final solution can be considered to be an adaptation of the Black-Scholes model to conditions that violate the underlying assumptions.

Third, the Genetic Programming method requires less data than other numerical techniques such as Neural Networks (Hutchinson, Lo, and Poggio (1994)).

Finally, the flexibility in adding terms to the parameter set used to develop the functional approximation can 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. The self-learning and self-improving feature also makes the method robust to changes in the economic environment. The Genetic Option Pricing Program is fast, 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).

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.

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

Chance, D. M., 1986. "Empirical tests of the pricing of index call options," *Advances in Futures and Options Research*, Vol. 1.

Chidambaran, N. K., 2003. Genetic Programming with Monte Carlo Simulation for Option Pricing, *Unpublished Working Paper*, Rutgers University.

Chidambaran, N. K., C. H. Jevons Lee, and Joaquin Trigueros, 1999. An Adaptive Evolutionary Approach to Option Pricing via Genetic Programming, in: *Computational Finance -- Proceedings of the Sixth International Conference, editors: Y. S. Abu-Mostafa, B. LeBaron, A. W. Lo, and A. S. Weigend,* Cambridge, MA: MIT Press.

Chidambaran, N. K. and S. Figlewski, 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).

Holland, J. H. 1975. *Adaptation in natural and artificial systems*, The University of Michigan Press, Ann Arbor.

Hull, J., 1993. *Options, Futures, and Other Derivative Securities*, 2nd 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).

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.

Macbeth, J. D. and Merville, L. J., 1979. "An empirical estimation of the Black-Scholes call option pricing model." *Journal of Finance*, Vol. 34 (December).

Macbeth, J. D. and Merville, L. J., 1980. "Tests of the Black-Scholes and Cox call option valuation models" *Journal of Finance*, Vol. 35 (May).

Marimon, R., McGrattan, E., Sargent, T.J., 1990. "Money as a medium of exchange in an economy with artificially intelligent agents." *Journal of Economic Dynamics and Control*, Vol. 14.

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

Neely, C., P. Weller, and R. Dittmar, 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., 1985. "Nonparametric Tests of Alternative Option Pricing Models." *Journal of Finance*, Vol. 40 (June).

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

AUTHOR BIOGRAPHY

N. K. CHIDAMBARAN is an Assistant Professor of Finance in the Rutgers Business School – Newark and New Brunswick at Rutgers University. He has a B.Tech. degree in Chemical Engineering from IIT, Bombay, and M.Phil. and Ph.D. degrees in Finance from New York University. His research interests are in the areas of Corporate Finance and Financial Derivatives. His email address is <chiddi@rci.rutgers.edu>.