Valuing Bermuda-Style Put Options using Symbolic Regression

By

Liam Johnston

Submitted in partial fulfillment of the
requirements for the degree of
Bachelor of Science with Honours in Computer Science
at
Saint Francis Xavier University
Antigonish, Nova Scotia
April 2023

Saint Francis Xavier University
Department of
Mathematics, Statistics and Computer Science

The undersigned hereby certify that they have read a thesis entitled "**Valuing Bermuda-Style Put Options using Symbolic Regression**" by **Liam Johnston** in partial fulfillment of the requirements for the degree of **Bachelor of Science with Honours**.

Dated: March 28, 2023

Supervisor: _____
Dr. James Alexander Hughes

Second Reader: _____
Dr. Milton King

St. Francis Xavier University

April 2023

Author:	Liam Johnston
Title:	Valuing Bermuda-Style Put Options using Symbolic Regression
Department:	Mathematics, Statistics and Computer Science
Faculty:	Science
Convocation	April 2023

Permission is herewith granted to Saint Francis Xavier University to circulate and to have copied for non-commercial purposes, at its discretion, the above title upon request of individuals or institutions.

Liam Johnston

to my parents, Chris and Rita

Abstract

Valuing stock options is an area in which extensive research has been conducted. This includes Kellogg and Charnes' work with decision tree and binomial lattice methods [10] and Carr and Madan's Fast Fourier Transform implementation [2]. This work is a comparative analysis of two different approaches for Bermuda-style put option valuation through obtaining regressions for simulated cash flows. These methods are the least-squares method implemented by Longstaff and Schwartz [12] and symbolic regression. The former is a deterministic method that uses a basis function whereas the latter is free to use any combination of independent variables, constants, and operators to determine the best predictor function for the dependent variable and was implemented in this work in a stochastic manner using genetic programming.

This research is applicable to options sellers who are looking to value their options and to options investors who are looking to value options that are available for purchase.

Acknowledgements

First I would like to acknowledge my supervisor, Dr. James Hughes. I am grateful for all the help he provided in this project. He has also been there since the beginning of my computer science journey. Prior to attending StFX, I had no previous programming experience and having Dr. Hughes as a professor for the introductory course opened my eyes to the world of computer science which led me to pursue it as my major.

Additionally, I would like thank Dr. Milton King for being my second reader on short notice. I am looking forward to getting the opportunity to work with him this summer.

Finally, I would like to thank my parents, Dr. Christopher Johnston and Dr. Rita Nigam. Although they do not have much knowledge of computer science, they are always there to listen to my issues and offer connections whom I can reach out to for help when necessary.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Essentially, stock options are a contract between two parties where the option holder has the right buy or sell a stock at an agreed upon price, known as the strike price, by a given expiration date. There are two types of options: put options, which lets the holder sell the stock at the strike price, and call options, which lets the holder buy the stock at the strike price. The former are useful if the holder expects the stock price to fall, as it will allow them to sell the shares at above market value, whereas the latter are useful if the holder expects the stock price to rise, as they can buy the shares at below market value.

Options also come in two major styles: European, where the holder can only exercise their options on the expiration date, and American, where the holder can exercise their option any time prior to the expiration date, which make them more valuable then the former [15]. However, options can also come in more exotic styles, such as Bermuda-style, where the holder can exercise the options only on specific dates prior to the expiration that are typically set in one-month increments [7].

Extensive research has been conducted on valuing stock options. Kellogg and Charnes

used decision tree and binomial lattice methods [10]. Additionally, Carr and Madan implemented a model using the Fast Fourier Transform [2].

This work is a comparative analysis of two different approaches for Bermuda-style put option valuation through obtaining regressions for simulated cash flows. These methods are the least-squares method implemented by Longstaff and Schwartz [12] and symbolic regression. Their differences are that the former is a deterministic method that uses a basis polynomial function whereas the latter is free to use any combination of independent variables, constants, and operators to determine the best predictor function for the dependent variable and is generally implemented stochastically using genetic programming (GP).

While the fact that Bermuda-style options can only be exercised generally once a month makes them less valuable than American-style options, this characteristic renders them better for stock price simulation as simulating a years worth of prices to value American-style options would require at least 365 time steps (since they can be exercised at any time, to determine exact precision, it would be necessary to simulate the price for all times when the stock price can change), whereas for Bermuda-style, it only requires 12 time steps.

Chapter 2

Genetic Programming Literature Review

Evolutionary algorithms (EA) are stochastic optimization techniques, which are employed for problem spaces that are too large to use deterministic techniques such as linear programming [6]. They take inspiration from the process of *natural selection* in order to determine a strong candidate solution to an optimization problem.

2.1 Background on Natural Selection

In the "Princeton Guide to Evolution", natural selection is described as a phenomenon that "occurs whenever there is a consistent, average difference in fitness (reproductive success) among sets of *individuals* among a given population that differ in some respect" [4].

An example of natural selection at work is the origin of giraffe's long necks [13]. The ancestors of modern giraffes were animals similar to deer or antelope, with necks of

ordinary length. Note that neck length of an animal can be referred to as a *phenotype*, which is defined as "the material organism, or some aspect of it, as contrasted with the information in the *genotype* providing the blueprint for the organism". The genotype is the information stored in the genes of one individual and can refer to anything from a single gene to all the genes, depending on context [4].

A *mutation* in the genotype of certain individuals in the population of the giraffe's ancestors would have caused them to have abnormally large necks in relation to their peers. However, because their food was located at the top of tall trees, the animals with longer necks had an advantage over those with shorter necks. This caused the longer-necked animals to reproduce more, therefore in the next generation, longer necks would have been more prevalent. The many generations of mutation and reproduction over millions of years resulted in the modern giraffe.

2.2 Genetic Programming

Genetic algorithms (GAs) borrow the concepts of natural selection and apply them to problem solving. They were first developed by John H Holland in the 1970s and this work was applied to applications such as developing algorithms that learned optimal pressure control for gas pipeline systems [8]. John R Koza expanded on the concepts of the GA by creating Genetic Programming (GP), which is a method of evolving computer programs, generally represented as trees, to generate solutions [11]

In this work, a GP algorithm was implemented for the purpose of symbolic regression. Here, an individual is a candidate solution for the problem which, in the case of symbolic regression, is a mathematical function. The individual's phenotype is the function itself and it's genotype is the dependent variables, constants, and mathematical operators

that make up the function. Additionally, the individual's fitness is how well the function suits the given sets of dependent and independent variables. This is usually measured by taking the *mean squared error* between every set of independent variables' respective dependent variable given by the dataset with its output given by function. The smaller the mean squared error, the better the fitness of the individual. The population consists of all the candidate solutions that exist in the GP algorithm.

2.2.1 Genetic Operators

Genetic operators are used to modify the existing individuals in the population in order to find ones with stronger fitness levels. While algorithm developers can be as creative as they deem in designing their genetic operators since there are no firm laws as to the definition of a GP algorithm, the main genetic operators are selection, crossover, and mutation.

Selection is the process of determining which individuals "survive" a generation and therefore are able to become "parents" for the subsequent generation. There are a wide variety of selection methods that are used in GP algorithms.

Crossover borrows from the concept of sexual reproduction. It selects two individuals from a generation and combines their genotypes to create "offspring" or individuals of the subsequent generation. As with selection, there are also many crossover methods that can be created when designing a GP algorithm.

A key concept when choosing selection and crossover methods is to ensure that the population remains diverse. This is because a population with similar genotypes will continue to produce offspring with similar genotypes. This has the potential to cause the algorithm to converge on a local maximum or minimum when there are still solutions

with better fitness levels.

Mutation is the manipulation of the genotype of a single individual. It exists to create diversity in the population and has the potential to find new genotypes that generate phenotypes with stronger fitness levels such as in the giraffe example.

When implementing genetic operators, it is standard that the individual with the highest fitness level in the population is always selected for the next generation and not modified through crossover or mutation. This is to ensure that the best fitness level cannot decrease as more generations are executed.

2.2.2 Flow of Execution

When combining all the aforementioned concepts, a GP algorithm can be designed. The general flow of execution is shown in Figure 2.1.

Each loop that executes the selection, crossover, mutation, and evaluation steps can be referred to as a generation. The termination clause is usually either a fitness threshold, meaning the algorithm terminates when the best individual reaches an arbitrary fitness level, or a generation threshold, meaning the algorithm terminates after an arbitrary number of generations. The algorithm returns the candidate solution with the highest fitness level as its most optimal solution. Generally, evaluating the fitness of the entire population is a bottleneck for performance.

2.3 Modular Enhancements

The following are modular enhancements to the GP algorithm implemented in this work.

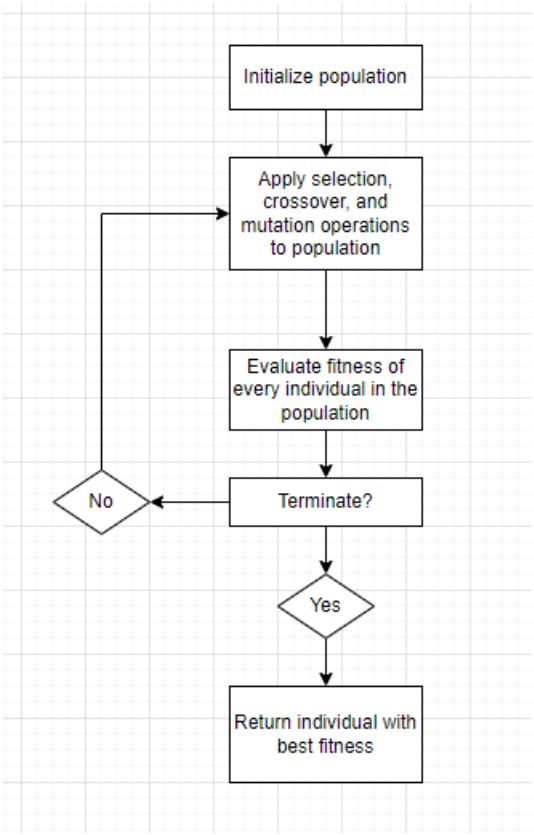

Figure 2.1: General flow of execution of a GP algorithm

2.3.1 Island Model

The GP algorithm implemented in this work uses the *island model*, meaning that during each "migration", the population splits into a given number of "islands". The genetic operators are then applied to the populations of each island independently from the populations of other islands over a number of generations.

As with all features of a GP algorithm, the migration mechanism can be implemented in various manners. Gozali and Fujimura describe the benefits and drawbacks of various migration mechanisms in their paper *Localized Island Model Genetic Algorithm in Population Diversity Preservation* [5]. In this work, migration consisted of combining the island's populations and randomly redistributing the individuals.

2.3.2 Individual Representation

As previously noted, individuals in a symbolic regression GP algorithm are mathematical functions. Equation 2.1 shows a possible individual for a symbolic regression problem with 3 dependent variables:

$$f(v1, v2, v3) = \frac{1.92(abs(-7.84 - v3))}{(v1 * v2) + cos(v2)} \tag{2.1}$$

While GP algorithms usually represent individuals as trees, this implementation represents them as an *acyclic graph*. Figure 2.2 depicts the acyclic graph representation of Equation 2.1. Each gene in the function can be one of four types: independent variable, constant, unary operator, or binary operator. In Figure 2.2, these types are represented as circles, ellipses, squares, and diamonds respectively. As you can see, the structure on the left cannot be classified as a tree given that the node labelled *v2* has

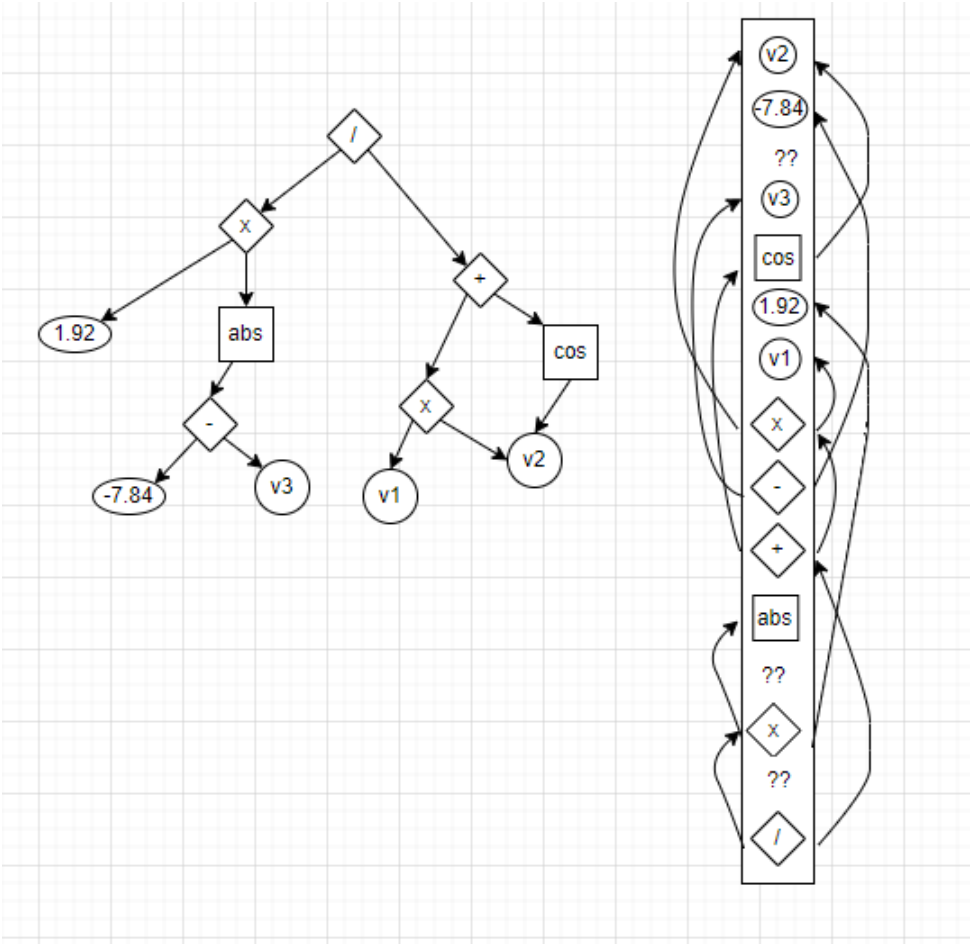

Figure 2.2: Acyclic graph and array representations of an individual

two parent nodes.

The reasons for opting with the acyclic graph representation are that it scales better, it has a lighter encoding, and avoids bloat, which is the tendency of individuals to grow arbitrarily large without significant fitness improvements [9]. The structure on the right of Figure 2.2 depicts how the acyclic graph may be stored in an array. The gene *v2* is referenced twice, which allows for a lighter encoding.

2.3.3 Fitness Predictors

The GP algorithm implemented in this work also made use of *fitness predictors*. Fitness predictors are not evaluated with the entire dataset. Instead an adjustable subset of approximately 10% of the entire set is used. They evolve alongside the main population and their fitness is determined by a measure of how well they approximate the entire dataset. The concept was created by Schmidt et al [14].

The purpose of fitness predictors is twofold. Only evolving using a subset of the dataset greatly reduces the number of computations the algorithm is required to perform. This also reduces the overfitting since the points used to evaluate fitness are changing every generation.

Chapter 3

Data

Equation 3.1 was used to generate simulations of a stock price over a given number of time steps. It was derived from the Black-Scholes model [1] and is based on geometric Brownian motion [3].

$$P(t+1) = P(t) * e^{(r-0.5\sigma^2)h+z\sigma\sqrt{h}} \tag{3.1}$$

where

- *P(t)* is the price of the stock at time step t
- z is a number randomly selected from a standard normal distribution
- r is the expected annual rate of return
- σ is the stock price volatility i.e. the standard deviation of r
- h is the length of the time step measured in years (i.e. h=1/12 indicates that a month occurs in between time steps)

Further parameters used to generate datasets that are not present in Equation 3.1 are *P(0)*, *N*, and *T*, which are the initial stock price, the number of simulations, and the number of time steps per simulation respectively. Additionally, the lifetime of the option in years is equal to $T * h$ and the discount rate, which is the adjustment for the value of money over time, is equal to e^{-rh}.

This equation can be used to generate simulations of real stocks by setting the parameters *P(0)*, r, and σ to values that exhibit the characteristics of the cash flow of the chosen stock[1]. In this work, research was not conducted to base values of *P(0)*, r, and σ on real stock prices. Instead, parameter values were generated from a normal distribution with the mean and standard deviation values for each parameter being arbitrarily selected. The only parameter held to a constant value was h, which was set to *1/12* to keep the standard of Bermuda-style options. Five simulations from a dataset are shown in Figure 3.1.

30 datasets in total were generated. Table 3.1 shows the mean values used to generate the parameter values and the parameter values for the 10 datasets.

[1]Note that N, T, and h are always arbitrarily selected as they are not traits of a stock

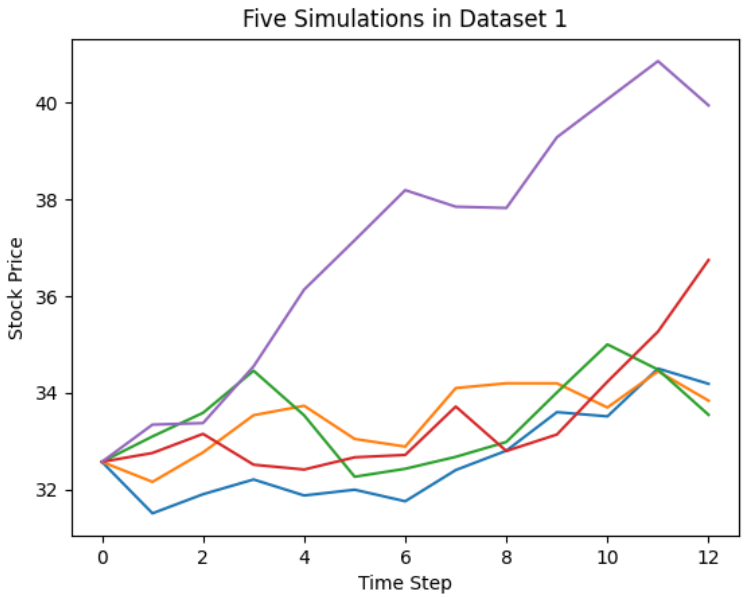

Figure 3.1: Five simulations from Dataset 1

Dataset	*P(0)*	*N*	*T*	*r*	σ
Mean	36.00	1000	12	0.0600	0.2000
1	32.57	909	13	0.0696	0.0811
2	36.94	1018	13	0.0421	0.2050
3	20.91	1179	14	0.0590	0.2284
4	35.49	1067	11	0.0720	0.2122
5	52.06	970	9	0.0552	0.1454
6	46.85	1056	15	0.0480	0.1884
7	40.40	818	14	0.0590	0.0959
8	50.97	1053	15	0.0693	0.1864
9	51.00	1170	16	0.0489	0.1695
10	52.13	957	8	0.0485	0.1676

Table 3.1: Combinations of Parameter Values Used to Generate Datasets

Chapter 4

Methods

This chapter takes a deeper dive into the specific implementation of the two employed techniques.

4.1 Least-Squares Method

The least-squares method was implemented using *numpy.polyfit(X, Y, deg)*, where X is an array of independent variables, Y is an array of dependent variables, and *deg* is the degree of the basis polynomial function[1]. For a polynomial of degree d, it returns a vector of coefficients [c_d, c_{d-1}, c_{d-2}, ..., c_1, c_0] that minimizes the squared error between $f(x) = c_d x^d + c_{d-1} x^{d-1} + c_{d-2} x^{d-2} + ... + c_1 x + c_0$ and the y value in Y associated with the given x value in X. Note that the regression is deterministic, meaning the same inputs will yield the same output. In this work, the degree of the basis polynomial function was selected to be 2.

[1]Official documentation can be viewed at `https://numpy.org/doc/stable/reference/generated/numpy.polyfit.html`

4.2 Symbolic Regression

The GP algorithm was implemented in Java and made use of the modular enhancements described in Chapter 2. The code can be viewed at `https://github.com/liamj1616/jGP`.

4.2.1 Parameters

Table 4.1 shows the adjustable parameters used by the algorithm and their selected values.

It must be noted that it is best practice run a stochastic algorithm with the same set of parameters at least 30 times on a single dataset and research should also be conducted on how adjusting the parameter values affects results. However, due to the heavy amount of computation required in this algorithm, it can take up to 15 minutes for it to generate a single valuation for datasets described in Chapter 3, therefore time constraints only allowed for one run on each dataset with a single arbitrarily selected set of parameter values.

4.2.2 Algorithm Structure

The pseudocode of the implemented algorithm is shown in Algorithm 1. Note that Lines 12 and 13 state that if the *fitness value* of the invididual *bestOverCurrentMigration* is less than the *fitness value* of the individual *bestOverAllMigrations*, then *bestOverCurrentMigration* becomes *bestOverAllMigrations*.

Parameter	Value	Description
NUMBER_RUNS	1	the number of times the algorithm is executed on a given dataset
NUMBER_MIGRATIONS	100	the number of migrations during a single execution of the algorithm
NUMBER_ISLANDS	7	the number of islands that the total population splits into during a migration
GENERATIONS	10100	the number of generations where genetic operators are applied to the island's population
POPULATION_SIZE	101	the number of individuals *per island*, meaning the total population size is equal to *POPULATION_SIZE* multiplied by *NUMBER_ISLANDS*
NUMBER_GENES	40	the number of constants, variables, and mathematical operators in the individual, which is a mathematical function
CROSSOVER_RATE	0.8	the likelihood of two individuals performing a crossover operation
MUTATION_RATE	0.1	the likelihood of a mutation operation being applied to an individual
NUMBER_PREDICTORS	10	the number of fitness predictors evolving alongside the population
PREDICTOR_SIZE	0.25	the percentage of the dataset being used to evolve the fitness predictors

Table 4.1: GP parameters and their selected values

Algorithm 1 GP algorithm pseudocode

```
1: Set parameters
2: Read data and set cashFlowMatrix
3: n ← N
4: while n > 2 do
5:     theData ← cash flow simulations at times T_n and T_{n−1}
6:     i ← 0
7:     while i < NUMBER_RUNS do
8:         Initialize bestCurrentMigration and bestOverAllMigrations
9:         Generate population and split the individuals into islands
10:        Initialize trainers and predictors
11:        j ← 0
12:        while j < NUMBER_MIGRATIONS do
13:            Run Genetic Operators on Islands
14:            bestCurrentMigration ← individual with best fitness
15:            if bestCurrentMigration < bestOverAllMigrations then
16:                bestOverAllMigrations ← bestCurrentMigration
17:            end if
18:            Shuffle population and re-distributed individuals to islands
19:            j ← j + 1
20:        end while
21:        Output bestOverAllMigrations's fitness value
22:        Output bestOverAllMigrations's function
23:        i ← i + 1
24:    end while
25:    Update cashFlowMatrix
26:    n ← n − 1
27: end while
28: Output the valuation and the average MSE across all regressions
```

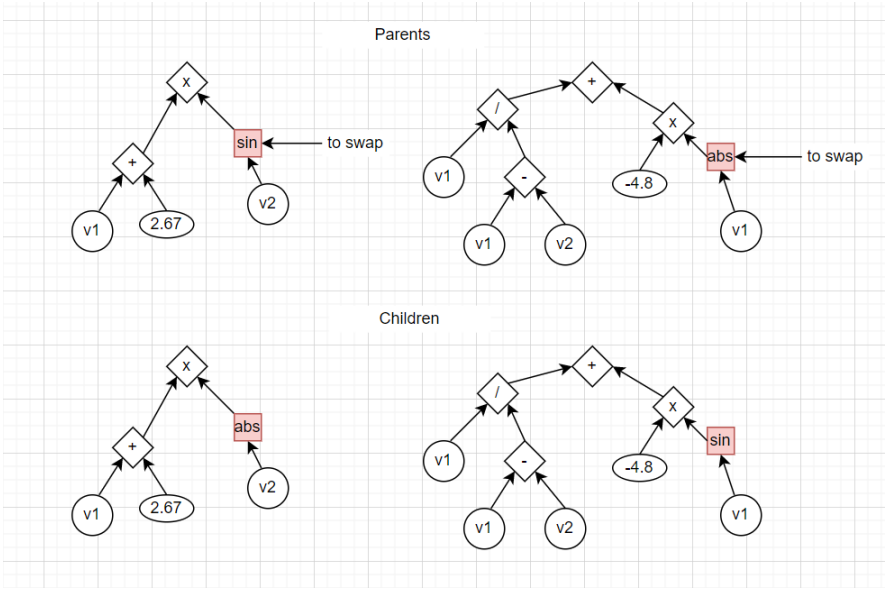

Figure 4.1: Crossover Operation

4.2.3 Islands

The algorithm was implemented in Java to make use of the built-in *Thread* class. Each island is its own instance of a Thread which allows them to run concurrently and therefore improves the algorithm's efficiency. The crossover and mutation operators are applied to each island's population over the given number of generations. Crossover involves the two parent individuals swapping genes of the same type to form two children. It is demonstrated in Figure 4.1.

Mutation is similar to crossover except it involves switching a gene in a single individual to a randomly selected different gene of the same type.

Chapter 5

Valuation Example

This Chapter walks through how a put option valuation is obtained using both the least-squares method and symbolic regression on a small example dataset, which can be viewed in Table 5.1. The dataset was generated using the parameter values initial stock price $P(0) = 36$, number of simulations $N = 8$, number of time steps per simulation $T = 3$, expected annual rate of return $r = 0.06$, stock price volatility $\sigma = 0.2$, and length of time step $h = 1/12$, which yields a discount rate of $e^{-(0.06)(1/12)} = 0.995$.

Simulation	t = 0	t = 1	t = 2	t = 3
1	36.00	36.76	35.54	35.04
2	36.00	36.83	36.86	43.39
3	36.00	30.65	28.48	29.72
4	36.00	35.45	37.74	35.83
5	36.00	32.95	30.04	22.76
6	36.00	34.55	37.68	38.49
7	36.00	37.36	39.79	44.96
8	36.00	33.59	32.84	33.28

Table 5.1: Stock Price Cash Flow Matrix

For this example, the strike price was set to \$40. If the stock option was of European-style, meaning early exercise is not possible, the option valuation would be obtained using Equation 5.1.

$$\frac{0.995^3(6 * 40 - (35.04 + 29.72 + 35.83 + 22.76 + 38.49 + 33.28))}{8} = 5.5263 \qquad (5.1)$$

Two noteworthy points regarding Equation 5.1:

- 0.995^3 comes from the fact that because the European option can only be exercised at the expiration date, which in this case is time step 3, this is the discounted value of the stock option at time step 3.
- Only options that are *in the money* are considered in the valuation equation. For put options, *in the money* refers to strike price > stock price. For call options, *in the money* refers to stock price > strike price. However, the equation is still divided by N as the more simulations where the option is not *in the money*, the less valuable the option.
- 6×40 is from the fact that there are 6 simulations where the option is in the money and the value of the option is strike price - stock price

For options that allow for early exercise, in order to optimally exercise stock options, it must be determined whether it is optimal to either exercise or continue to hold at each time step. As a result, regression is used to determine a conditional expectation function $E(Y|X)$ at time step t, where X is the value of the stock at time step t and Y is the expected discounted value of the stock at time step $t + 1$. For put options, if

$$\textit{strike price - } X > E(Y|X),$$

then the decision should be made to exercise the option and if

$$\textit{strike price - } X < E(Y|X)$$

then the decision should be made to hold the option.

Regression to determine $E(Y|X)$ takes an iterative approach, beginning with regressing time step $T-1$, to time step T, which in this example is regressing from time step 2 to time step 3. X is the stock price at time step $T-1$ and Y is the discounted value of the stock price subtracted from the strike price at time step T. Only options that are *in the money* in time step 2 are considered in the regression. Table 5.2 shows the values used for regression from time step 2 to time step 3 to obtain $E(Y|X)$. Because all stock prices in X are *in the money*, they are all included in Table 5.2. The y values for simulations 2 and 7 are $0 * 0.995$ because the stock price is not *in the money* at these points.

Simulation	X	Y
1	35.54	4.96 * 0.995
2	36.86	0.00 * 0.995
3	28.48	10.28 * 0.995
4	37.74	4.17 * 0.995
5	30.04	17.24 * 0.995
6	37.68	1.51 * 0.995
7	39.79	0.00 * 0.995
8	32.84	6.72 * 0.995

Table 5.2: Regression from time step 2 to time step 3

Simulation	strike Price - X	Y	$E(Y\|X)$		Squared Error	
			SR	LSM	SR	LSM
1	4.46	4.93	3.13	4.75	3.24	0.0324
2	3.14	0.00	5.53	3.08	30.5809	9.4864
3	11.52	10.23	3.98	13.63	39.0625	11.56
4	2.26	4.15	-2.54	1.97	44.7561	4.7524
5	9.96	17.15	20.60	11.67	11.9025	30.0304
6	2.32	1.50	5.820	2.04	0.0083	0.0364
7	0.21	0.00	3.22	-0.63	10.3684	0.3969
8	7.16	6.69	6.78	8.15	0.0081	1.8769
MSE					17.4909	7.2714

Table 5.3: $E(Y|X)$, squared error and mean squared error (MSE) between Y and $E(Y|X)$ from least-squares method (LSM) and symbolic regression (SR) regressing from time step 2 to time step 3

This is where the differences of the two approaches matter. The least-square method uses a basis function for its regression. It obtains a regression function of

$$E(Y|X) = -(6.14 * 10^{-4})x^2 - 1.22x + 48.84$$

Running symbolic regression on this dataset using the implemented GP algorithm described in Chapter 4 obtained the function

$$E(Y|X) = tan(15.53x) + 5 - (7.13/x)$$

Table 5.3 shows the results of $E(Y|X)$ based on these regression functions and their squared errors against individual y in Y as well as the mean squared error against Y.

The next step is to determine from the regression if it is optimal to exercise at time step 2. As previously mentioned, this is the case when *strike price* - $X > E(Y|X)$. Table 5.3 shows that for symbolic regression, it is optimal to exercise at time step 2 for simulations 1, 3, 4, and 8, and for the least-squares method, it is optimal to exercise at

time step 2 for simulations 2, 4, 6, and 7. Additionally, although symbolic regression states that the option should be exercised at time step 3 for simulations 2 and 7, because the stock is not *in the money* at these times, the option cannot be exercised.

After performing this regression, the cash flow matrices can be updated. Given that the models yielded different optimal exercise times, they yield different updated cash flow matrices. Tables 5.4 and 5.5 show the updated cash flow matrices after the regression from time step 2 to time step 3 for symbolic regression and the least-squares method, respectively.

Simulation	t = 0	t = 1	t = 2	t = 3
1	36.00	36.76	4.46	0.00
2	36.00	36.83	0.00	0.00
3	36.00	30.65	11.52	0.00
4	36.00	35.45	2.26	0.00
5	36.00	32.95	0.00	17.24
6	36.00	34.55	0.00	1.51
7	36.00	37.36	0.00	0.00
8	36.00	33.59	7.16	0.00

Table 5.4: Updated Cash Flow Matrix for symbolic regression after regressing from time step 2 to time step 3

Note that if it has been determined that for a simulation it is more optimal to exercise than continue the option at time step 2, the value the stock price at time step 3 is set to 0 in the updated cash flow matrix. This is due to the fact that once the option has been exercised, future stock price values have no effect on the value of the option. Furthermore, if is has been determined that for a simulation it is more optimal to continue than exercise at time step 2 but time step 3 is not in the money, then both

Simulation	t = 0	t = 1	t = 2	t = 3
1	36.00	36.76	0.00	4.96
2	36.00	36.83	3.14	0.00
3	36.00	30.65	0.00	10.28
4	36.00	35.45	2.26	0.00
5	36.00	32.95	0.00	17.24
6	36.00	34.55	2.32	0.00
7	36.00	37.36	0.21	0.00
8	36.00	33.59	0.00	6.72

Table 5.5: Updated Cash Flow Matrix for least-squares method after regressing from time step 2 to time step 3

values are set to 0 in the updated cash flow matrix.

Now, the next regression is performed. In this simple example, it is also the final one. No regression is performed from time step 0 to time step 1 since Bermuda-style options do not allow for exercising on the day the option has been purchased [7].

Table 5.6 shows the values that will be used to perform this regression, which are derived from the cash flow matrices updated after the first regressions. Note that again, all stock prices are *in the money* so they are all included, and that the X values are the same for both regressions, but the Y values differ due to the difference in results obtained by the previous regression.

The least-squares method obtains the function

$$E(Y|X) = -0.81x^2 + 57.82x - 1029.21$$

with a mean-squared error of 0.5681, while symbolic regression obtains the function

$$E(Y|X) = e^{e^{cos(-0.9449-x-10.5779)}}$$

Simulation	X	SR Y	LSM Y
1	36.76	4.46*0.995	0.00*0.995
2	36.83	0.00*0.995	3.14*0.995
3	30.65	11.52*0.995	0.00*0.995
4	35.45	2.26*0.995	2.26*0.995
5	32.95	0.00*0.995	0.00*0.995
6	34.55	0.00*0.995	2.32*0.995
7	37.36	0.00*0.995	0.21*0.995
8	33.59	7.16*0.995	0.00*0.995

Table 5.6: Regressions for symbolic regression (SR) and least-squares method (LSM) from time step 1 to time step 2

with a mean-squared error of 17.99.

The updated and final cash flow matrices for symbolic regression and the least-squares method after this regression can be viewed in Tables 5.7 and 5.8 respectively.

Simulation	t = 0	t = 1	t = 2	t = 3
1	1.00	3.24	0.00	0.00
2	1.00	3.17	0.00	0.00
3	1.00	9.35	0.00	0.00
4	1.00	4.54	0.00	0.00
5	1.00	0.00	0.00	17.24
6	1.00	0.00	0.00	1.51
7	1.00	2.64	0.00	0.00
8	1.00	6.41	0.00	0.00

Table 5.7: Updated Cash Flow Matrix for symbolic regression after regressing from time step 1 to time step 2

Pseudocode for calculating the valuation of the option from the final cash flow matrix

Simulation	t = 0	t = 1	t = 2	t = 3
1	1.00	0.00	0.00	4.96
2	1.00	3.16	0.00	0.00
3	1.00	0.00	0.00	10.28
4	1.00	4.54	0.00	0.00
5	1.00	0.00	0.00	17.24
6	1.00	5.45	0.00	0.00
7	1.00	2.64	0.00	0.00
8	1.00	0.00	0.00	6.72

Table 5.8: Updated Cash Flow Matrix for least-squares method after regressing from time step 1 to time step 2

can be viewed in Algorithm 2. Note that multiplying the value of the cash flow matrix by $discountRate^j$ prior to adding it to the sum is accounting for the change in the value of money over time.

Applying Algorithm 2 to the final cash flow matrices yields a valuation of 5.9604 for symbolic regression and 6.7928 for the least-squares method. Additionally, their total mean-squared errors are 17.7405 and 3.9511 respectively. Note that both of these valuations are higher than the valuation of 5.5263 if the option was European-style.

5.1 Applicability

After seeing how data is generated and how to apply these models to generated datasets, a logical question that arises is how can options sellers and investors apply this research? The first step for both parties would be to gather information regarding the stock to determine the appropriate parameter values to generate the simulations.

Options sellers could then run these models on the generated dataset to determine a

Algorithm 2 Pseudocode for obtaining a valuation from the final cash flow matrix

```
sum ← 0
i ← 0
while i < N do
    j ← 1
    while j < T do
        if cashflowMatrix[i][j]! = 0 then
            sum ← sum + cashflowMatrix[i][j] ∗ discountRate^j
            break
        end if
        j ← j + 1
    end while
    i ← i + 1
end while
valuation ← sum/N
Output valuation
```

valuation for their options. Options investors could run this model given a stock where they are considering purchasing options to determine if the valuation they obtain is different than the seller's valuation.

Selecting appropriate parameters for the data generation is the most critical aspect as this ensures that the regression models are fitting to cash flows that may actually occur.

Chapter 6

Results and Discussion

When running the algorithms, the strike price was kept constant at $40 as it was in the example. Table 6.1 shows the valuations and mean squared errors from symbolic regression and the least squares method on 10 of the datasets, as well as the averages across all 30 datasets.

While valuations are relatively similar, Table 6.1 shows that the mean squared error for the least-squares method is lower across the 10 shown datasets in relation to symbolic regression. In fact, the mean-squared error is actually lower for the least-squares method across all 30 datasets. This may seem counterintuitive; shouldn't the freedom exhibited by symbolic regression lead to lower regression errors? While it is highly likely that there are functions in the search space of symbolic regression that produce regressions with lower errors than in the search space of the least-squares method, the issue is that the GP algorithm has to find these functions. Because the search space in symbolic regression is so large, this may not occur, which is what happened in this work. It is possible that different sets of parameter values can allow the GP algorithm to find more accurate regressions.

Dataset	Valuation		Mean Squared Error	
	SR	LSM	SR	LSM
1	7.2347	6.8404	1.3914	0.5885
2	4.0691	4.971	7.4623	3.064
3	18.9786	17.0611	2.0843	0.6732
4	4.6874	5.6055	5.7150	3.6529
5	0.0179	0.02350	1.1417	0.0996
6	0.8026	0.9063	4.8739	1.9912
7	0.6868	0.7141	1.0581	0.8033
8	0.2883	0.3446	4.664	1.0693
9	0.1818	0.2807	2.872	0.7246
10	0.0335	0.03087	1.6051	0.2987
Average	3.8740	3.9395	4.4968	2.4430

Table 6.1: Valuations and mean-squared error of symbolic regression (SR) and least squares method (LSM) across 10 datasets along with the average across all 30 datasets

Chapter 7

Future Work

There is much room for future work in this study. As previously mentioned, given that GP is a stochastic algorithm, the same set of parameters should be run at least 30 times on a given dataset to increase validity of results. Additionally, research should be conducted on how adjusting the parameter values affects the results. Furthermore, different genetic operator implementations could also be tested. In terms of the least-squares method, different degree polynomials can also be tested to determine the effect this has on results.

Both approaches can also be studied on call options. This was not studied in this work as it only causes small edits to the processes, therefore does not have a massive impact on results.

It was also previously noted that the parameters of the data generation function could be set to model real stocks. The most common use of stock options are call options given to employees by start-up businesses as means of compensation [15]. This is due to the fact that they have yet to generate sufficient profit to guarantee compensation. Datasets could be generated with parameters that model those of specific

start-ups, then the valuation models studied here could be applied. This is similar to how Kellogg and Charnes [10] focused their stock options valuation research on startup biotechnology companies, as they often have high valuations although they are years away from being able to generate any product revenue since the products are in early stages of development. The difficulty with modelling start-up cash flows is that it would be very tough to obtain an accurate dataset, given that the r and σ values would be difficult to accurately guage.

Bibliography

[1] Fischer Black and Myron Scholes. The pricing of options and corporate liabilities. *Journal of political economy*, 81(3):637–654, 1973.

[2] Peter Carr and Dilip Madan. Option valuation using the fast fourier transform. *Journal of computational finance*, 2(4):61–73, 1999.

[3] Kinder Chen. Black-scholes model and monte carlo simulation. `https://kinder-chen.medium.com/black-scholes-model-and-monte-carlo-simulation-d8612ac4519b`, January 2021.

[4] Douglas J. Futuyama. Natural selection and adaptation. In *The Princeton Guide to Evolution*, pages 189–191. Princeton University Press, 2014.

[5] Alfian Akbar Gozali and Shigeru Fujimura. Localized island model genetic algorithm in population diversity preservation. *Atlantis Highlights in Engineering (AHE)*, 2:122–128, 2019.

[6] A Hanif Halim and I Ismail. Combinatorial optimization: comparison of heuristic algorithms in travelling salesman problem. *Archives of Computational Methods in Engineering*, 26(2):367–380, 2019.

[7] Adam Hayes. Bermuda option: What they are, examples, pros and cons. https://www.investopedia.com/terms/b/bermuda.asp, April 2022.

[8] John H Holland. Genetic algorithms. *Scientific american*, 267(1):66–73, 1992.

[9] James Hughes. Finding nonlinear relationships in functional magnetic resonance imaging data with genetic programming. *Western Libraries Electronic Thesis and Dissertation Repository*, 2018. Available at https://ir.lib.uwo.ca/etd/5491.

[10] David Kellogg and John M Charnes. Real-options valuation for a biotechnology company. *Financial analysts journal*, 56(3):76–84, 2000.

[11] John R Koza. Genetic programming as a means for programming computers by natural selection. *Statistics and computing*, 4:87–112, 1994.

[12] Francis A. Longstaff and Eduardo S. Schwartz. Valuing american options by simulation: A simple least-squares approach. In *The Review of Financial Studies*, volume 14, pages 113–147, 2001.

[13] Michael Marshall. Natural selection. https://www.newscientist.com/definition/natural-selection/.

[14] Michael D. Schmidt and Hod Lipson. Coevolution of fitness predictors. *Evolutionary Computation, IEEE Transactions on*, 12(6):736–749, 2008.

[15] Roger Wohlner and Andrew Goldman. What are stock options and how do they work? https://www.wealthsimple.com/en-ca/learn/what-is-a-stock-option, March 2023.