Pricing Exotic Options with Fast Approximation Algorithms *

Sudarshan P. Gururaj

April 29, 2001

Abstract

This paper examines the problem of pricing exotic options accurately and quickly by approximation algorithms. The analytical Black-Scholes Formula or the Cox-Rubinstein-Ross binomial tree method accurately, and sometimes exactly, price simple, or plain vanilla, options. Exotic options, particularly path-dependent options, cannot be priced by these standard techniques and must be priced by approximations in practice. In this paper, we discuss the difficulty of pricing Asian and basket options, and survey several pricing algorithms, including the bucketed tree traversal algorithm and its variants developed by Akcoglu, et. al. We then present implementations of bucketed tree traversals and its recursive improvement, along with a comparison of the accuracy and running times with other available algorithms in literature.

1 Introduction

Financial derivatives, securities whose value depends on the value of other securities, have become increasingly more important in financial markets. Sophisticated investors and financial institutions use derivatives to protect against possible principal investment losses in stocks, commodities, and other assets. Hedge funds, arbitrageurs, and investment managers also use derivatives to assume large positions in investments at little cost. Derivatives include such contingent securities as futures, forwards, options, and swaps. Since such securities involve many underlying factors, including underlier price, time, interest rates, and other financial variables, it may be difficult to determine a fair market value.

Options, especially those with equity underliers, are among the most popular financial derivatives. An option is a contract specifying the right to purchase or sell an asset for a particular price, called the strike price, during or at a particular time during the life of the contract. Calls, options to purchase, and puts, options to sell, can be traded in two styles: European or American. European options can only be exercised at the expiration of the option contract, while American options can be exercised between the time an option is written and its expiration. [9]

As traders, hedgers, and speculators have used financial derivatives throughout the past thirty years, they have also developed more sophisticated options with complex conditions on the strike price and methods of execution to suit their particular investment needs. Furthermore, some options have been written on several assets at once, allowing traders to make positions on baskets of assets or the relative performance of different assets. Such sophisticated options, known as exotic options, have become more complex and more prevalent in financial markets as investors try to

---

*Based on research work performed with and algorithms developed by Karhan Akcoglu, Ming-Yang Kao, and Shuba Raghavan in Fast Pricing of European Asian Options with Provable Accuracy: Single-stock and Basket Options Special thanks to my parents for their support; to Prof. Rust and Prof. Kao for all their help, guidance, and inspiration; and to Karhan Akcoglu and Petros Drineas for their insight.
achieve increasingly more diverse investment objectives. For simplicity and generality, this paper only considers options that are based on single or multiple stocks, although traders write options on fixed income, foreign currencies, and commodity futures instruments as well. [11]

The popularity of options and other derivatives increased with two parallel developments in financial theory: the Black-Scholes option pricing formula and Stephen Ross's Arbitrage Pricing Theory. Until the publication of the Black-Scholes theory in 1973, most options were traded over-the-counter and with little knowledge of determining the fair value. Both theories are based on the idea that any financial security must have a value equal to the value of a portfolio of securities with similar cash flow streams. This breakthrough in financial theory led to the greater popularity of options and more efficient and sophisticated financial markets. While the Black-Scholes formula provided a closed-form valuation of simple options, exotic options are not as easy to price. Exotic options, loosely defined, are options whose payoff at expiration is a complex function of different factors, including the asset price and the option's strike. Numerical pricing techniques that arise from the Black-Scholes pricing theory provide good approximations, but often do not run in sufficient time to be of use in real-time trading systems. Furthermore, current financial theory often does not specify running times nor analytic error bounds that such approximations inherently possess. Such information is valuable in practice for traders who must price options in real-time, responsive markets. [4]

Option pricing theory has applications beyond investment management. Many financial contracts and regulations involve decisions that have financial values and can be analyzed as embedded options. Other situations in the social sciences, such as elections and laws, are considered to be real options, contingent decisions that have structures that are similar to options contracts. Therefore, option pricing theory also has larger applications in decision theory and in the social sciences.

The purpose of this paper is to investigate the empirical performance of bucketed tree traversal approximation algorithms developed by Akcoglu, et, al. in the pricing of Asian and basket options, two kinds of exotic options. Since the Black-Scholes pricing formula does not apply and the CRR binomial method is too cumbersome for such options, approximation techniques are necessary. Most computational approximation algorithms, such as bucketed tree traversal, are based on the structure of CRR binomial tree. This paper examines implementation of such algorithms and compares their accuracy and running time performance with other approximation pricing methods. Additionally, we present analytical expressions of the accuracy and running times of the algorithms, unlike most existing pricing methods and techniques in finance literature. Section 2 briefly describes the basics of traditional option pricing theory. Section 3 presents the difficulty of pricing exotic options with traditional techniques reviewed earlier. Section 4 discusses Asian options, presents two versions of the bucketed tree traversal algorithm, and provides some experimental results that demonstrate the superiority of such algorithms over available methods. Section 5 presents extension of ideas in Asian option pricing to the pricing basket options. We conclude with a summary of our results and a discussion of further research.

2 Review of Standard Option Pricing Techniques

2.1 Asset Pricing Theories and the Black-Scholes Formula

An option is the right to purchase or to sell an asset for a particular price during a particular period of time. As with other derivatives, options are relatively difficult to price in comparison to other purely financial instruments, such as bonds. The payoff function at expiration for a call option is $\max(S - X, 0)$ where $S$ is the stock price at option expiration and $S$ is the strike price, while the
payoff for a put option is \( max(X - S, 0) \). Puts are related to calls for non-dividend paying stocks by the put-call parity relation, \( C + X e^{-rT} = P + S_0 \) where \( C \) and \( P \) are the values of the call and the put, respectively, and \( S_0 \) is the initial stock price. In this paper, we only consider pricing call options, since put options can be priced in a similar fashion or be priced as a function of the value of the call option with a similar parity relation. [9]

Before the development of arbitrage pricing techniques, financial theory primarily used the discounted cash flow method for pricing securities, such as simple bonds with easily understood cash flows. The discounted cash flow method for pricing securities requires forecasting expected cash flows and discounting those cash flows by a discount rate that reflect the risk inherent in the security, known as the opportunity cost of capital. As with other derivatives, several factors affect the expected cash flow and the opportunity cost of capital throughout the life of the option. [4]

The breakthrough in options pricing results from the principle of arbitrage. Arbitrage is the simultaneous purchase and sale of essentially the same asset resulting in risk-free profit. In an efficient market, two assets that have the same cash flows and affected by the same factors should have the same price. This assumption of an arbitrage-free market arises from the assumption of the Efficient Markets Hypothesis, developed by Fama and French in the late 1960s. The Efficient Markets Hypothesis can be stated in three forms:

1. Weak form: It is impossible to make consistently superior profits by studying past returns. Prices will follow a random walk.

2. Semistrong form: Prices immediately respond to any public information, which therefore implies that prices reflect not just past prices but all other published information.

3. Strong form: Prices reflect all the information that can be acquired through analysis and insider information.

The Efficient Markets Hypothesis and the resulting assumption of no arbitrage suggest that options can be priced by pricing a portfolio of securities with returns equal to the cash flows provided by options. This idea underlies the Black Scholes options pricing formula in particular and Stephen Ross’s Arbitrage Pricing Theory in general. [4]

Stock prices are generally assumed to follow a continuous-time Markov process. The probability distribution of a stock’s price at any time does not depend on the particular history of prices before a given instant. [12] The assumption of a Markov process for stock prices is a direct result of the weak form of the Efficient Markets Hypothesis. Furthermore, stocks are assumed to grow usually at a certain growth rate with some random, normally distributed disturbances, following the geometric Brownian form

\[
\frac{dS}{S} = \mu \, dt + \sigma \, dz
\]

where \( S \) is the stock’s price, \( \mu \) is the average growth rate, \( \sigma \) is the standard deviation of the price, and \( dz \) is a Weiner process with \( dz = \phi \sqrt{dt} \) such that \( \phi \) is a random variable with standard normal distribution. The assumption that stock prices follow a geometric Brownian process leads to the observation that stock prices are distributed lognormally. [9]

From the observations about the movement of stock prices and Ito’s Lemma, which states a function of a random variable with geometric Brownian behavior also follows geometric Brownian motion, Fischer Black and Myron Scholes proved that the value of a plain vanilla call option on a stock is described by

\[
dV = \left( \frac{\partial V}{\partial S} \mu S + \frac{\partial V}{\partial t} + \frac{1}{2} \frac{\partial^2 V}{\partial S^2} \sigma^2 S^2 \right) dt + \frac{\partial V}{\partial S} \sigma S dz,
\]
assuming markets are efficient, no transaction costs exist, and that the risk-free interest rate \( r \) and the volatility \( \sigma \) are known. Since markets are assumed to be efficient, the equation results from equating the value of the call option to a portfolio of the underlying stock and a bond at the risk-free rate weighted appropriately. Since the portfolio and the call option have the same cash flow, the prices of the two securities are equivalent. [14]

The Black-Scholes-Merton differential equation does not involve any variables, such as the mean return of the stock, that are affected by the risk preferences of the investor. Since this fundamental equation of options pricing does not consider investor’s risk preference, most option pricing theories assume that investors are risk neutral. This assumption of risk neutrality is, in fact, artificial and needed only to simplify determining the solution to the differential equation. The same result are obtained in a risk-averse model as in the riskneutral model, because the expected growth rate of the stock price and the discount rate both change by exactly the same amount. However, in the risk-neutral world, investors require no compensation for risk and the expected return of any security is the risk-free interest rate.

In the case of European plain-vanilla options, the Black-Scholes-Merton differential equation can be solved analytically. The resulting closed-form solution is the famous Black-Scholes option pricing formula. The Black-Scholes formula for a plain vanilla European call is \( C = S_0 N(d_1) - X e^{-r(T-t)} N(d_2) \) where \( S_0 \) is the initial stock price, \( X \) is the strike, \( r \) is the risk-free interest rate, \( T \) is the maturity, \( N() \) is the normal distribution function, and \( d_1 = \frac{\ln(S_0/X) + (r - \sigma^2/2)(T-t)}{\sigma \sqrt{T}} \) and \( d_2 = d_1 - \sigma \sqrt{T} \).

2.2 Cox-Ross-Rubinstein Binomial Tree Pricing Model

While the Black-Scholes formula price plain vanilla European options exactly, it cannot handle other variations, such as American-style options and certain exotic options. These options have payoffs whose behavior is not conducive to finding an analytical solution to the Black-Scholes-Merton differential equation. For example, if the underlying stock of an American option pays a dividend, then the value of stock experiences a jump discontinuity in its price. It can be shown that an American option on a stock with no dividend has the same value as a European option on the same stock; however, an American option on a dividend-paying stock is worth more than its European counterpart, since it can be exercised early to capture the value of the dividend. [4]

When a differential equation cannot be solved analytically, it must be solved by numerical techniques that approximate the derivatives in the equation with finite difference. One such approximation that is commonly used in options pricing in the Cox-Rubinstein-Ross binomial tree model. [14]

The CRR binomial tree is a recombinant binary tree where the nodes of each level of tree represent possible prices achieved by the underlying asset of the option at a particular period during the life of the option. At each level \( t \) of the tree \( T \), \( T \) has \( t+1 \) nodes. If \( T[t,i] \), represents the \( i \)-th node of the tree at level \( t \), then \( T[t,0] \) and \( T[t,t] \) have one parent each and every other node \( T[t,i] \) has two parents, \( T[t-1,i] \) and \( T[t-1,i] \). Therefore, at a given level \( t \), there are \( t+1 \) nodes at that level and the total number of nodes in the tree \( T \) with depth \( n \) is \( \binom{n+1}{2} \). Every node \( T[t,i] \) has associated asset price \( s(T[t,i]) \) with children having prices \( S(T[t+1,i]) = dS(T[t,i]) \) with assigned probability of occurrence of \( 1-p \) and \( S(T[t+1,i+1]) = uS(T[t,i]) \) with probability of \( p \).

The above description of the CRR binomial tree is a discrete random walk that should approximate the continuous random walk process that underlies the Black-Scholes-Merton differential
equation. The CRR binomial model is, in fact, based on two assumptions that link it with the Black-Scholes theory.

1. A continuous random walk may be modelled by a discrete random walk with the following properties:
   
   (a) The asset price $S$ changes only at the discrete times $\delta T, 2\delta t, 3\delta t, \ldots$ up to $M\delta t = T$, the expiry date of the derivative security.
   
   (b) If the asset price is $S^m$ at time $m\delta t$, then, at time $(m + 1)\delta t$, it may take one of only two possible values, $uS^m > S^m$ or $dS^m < S^m$.

2. Options are priced in a risk-neutral world.

The parameters of the CRR binomial model are the result of using the discrete random walk process to approximate the continuous random walk model. Under the assumptions that underlie the binomial model, the parameters of the model are $u = e^{\sigma \sqrt{\delta t}}$, $d = e^{-\sigma \sqrt{\delta t}}$, and $p = \frac{e^{r \delta t} - d}{u - d}$ or $p = \frac{1 + r - d}{u + d}$. [14]

The following simple algorithm prices the plain vanilla call option for both European and American options.

**Algorithm 1 Backward Induction on the CRR Binomial Tree [9]**

1. Given the stock's price $S_0 = S_0[0, 0]$, the prices and probabilities at the nodes of the binomial tree of the asset can be determined by $s(T_S[t + 1, i + 1]) = us(T_S[t, i]) + D(T_S[t + 1])$ and $s(T_S[t, i]) = ds(T_S[t + 1, i] + D(T_S[t + 1]))$ where $u$, $d$, and $p$ are determined as above where $D(T_S[t + 1])$ is the dividend payment at time $t + 1$ if it exists.

2. Constructing a similarly structured tree to the tree of stock price $T_S$, assign terminal payoff values to the leaf nodes of the option tree $T_O$ such that $s(T_O[n, \bar{i}]) = s(T_S[n, \bar{i}]) - X$ for $0 \leq i \leq n$ where $X$ is the strike price of the option.

3. The price at each node in the option tree is given by

$$s(T_O[t, i]) = \frac{ps(T_O[t + 1, i + 1]) + (1 - p)s(T_O[t + 1, i])}{1 + r}$$

4. The price at the root node of the option tree is the price of the option.

**Theorem 1** The running time of pricing a plain vanilla European option is $\Theta(n^2)$ where $n$ is the number of levels in the binomial tree.

**Proof:** Assuming all calculations of prices at different tree nodes takes constant time, the number of calculations on both trees is equal to the number of nodes in both trees. Since the number of nodes at each level of the tree is one more than the level number, the running time of the algorithm is $\Theta(n^2)$. ■
3 Difficulty of Pricing Exotic Options

3.1 Description of exotic options

While the CRR binomial model prices plain vanilla European and American options accurately and quickly, it cannot price some exotic options with such accuracy or efficiency. Exotic options have payoff functions dependent on not only the price of one underlier over the life of the option, but also the price of other assets, economic factors, and functions of the asset price. The two options we considered from now on in the paper, Asian and basket options, belong to larger classes, respectively known as path-dependent and rainbow options.

Rainbow options are options whose payoffs are functions of more than one asset. Such options include

1. outperformance options: an option whose payoff depends on the difference of the value of two assets at expiration; \( C = \max(S_2 - S_1, 0) \).

2. best of (worst of) two options: an option whose payoff is the value of the better (worse) of two assets; \( C = \max(S_1, S_2) \)  \( (C = \min(S_1, S_2)) \).

3. basket options: an option whose underlier is the weighted sum of several different assets; \( C = \max(\sum_{i=1}^{n} \omega_i S_i, 0) \).

Basket options are the most interesting of the rainbow options, because market indices and investment portfolios can themselves be valued as asset baskets and can be hedged with basket options. Index options, in particular, are used to hedge against market risk in a variety of investment management applications. Basket options are commonly valued in practice by assuming the payoffs are themselves lognormally distributed and applying the Black-Scholes formula. Since the weighted sum of lognormal variables may not necessarily be lognormal itself, pricing basket options with the Black-Scholes formula is a fundamentally erroneous approximation. The CRR binomial method apparently does not price basket options well either, since a binomial tree must be constructed for each component of the asset basket. Since every possible combination of paths from every asset tree must be considered, the standard CRR binomial tree method generates paths that are exponential in the number of stocks in the basket. Therefore, the running time of the traditional CRR algorithm on basket options runs is insufficient. [11]

Path-dependent options only depend on one asset, but they also depend on the time to maturity and a function of the asset price history or its path. In other words, path dependent options depend on an implicit function \( I = f(S, t) \). For example, lookback options, options whose value depend on the maximum or minimum value achieved by the asset over the life of the option, have payoff functions of the form \( C = \max(S - I(S, t), 0) \) with \( I(S, t) = \min_{0 \leq \tau \leq t} S(\tau) \). The implicit function of a path-dependent option may be monitored and valued continuously or discretely in pricing theory, although, in practice, only discrete monitoring is possible. Asian options, the option we consider for the rest of this paper, are path dependent options with payoff function dependent on \( I(S, t) = \frac{1}{t} \int_{0}^{t} S(\tau) d\tau \). [7]

Path-dependent options can be visualized and priced in the following mathematical model. Given a binomial tree \( T \), a path in the tree can be described as a sequence of \( n \) independent coin tosses \( \omega = \omega_1, ..., \omega_n \) where each \( \omega_i \in H, T \). From this definition of a path, we can define \( \Omega \) to be the set of all possible coin-toss sequences and the random variable \( X_\omega \) such that for an \( \omega \in \Omega \),

\[
X_\omega = \begin{cases} 
+1 & \text{if } \omega_i = H \\
-1 & \text{otherwise}
\end{cases}
\]
This representation describes the binomial tree as a random walk model with drift $p$, the probability of getting heads or moving up one step in the tree. We can define a probability measure $P$ on $\Omega$ to be the measure for which $X_i$ are independent, identically distributed with $P[X_i = 1] = p$ and $P[X_i = -1] = 1 - p$.

The stock price, given the binomial tree, is a Markov or martingale process, since the stock price, $S_k + 1$, is not determined by the sequence $\{X_i\}_{i=1}^n$, but rather by just the current stock price, $S_k$. Path-independent options are options whose payoff functions $G_n$ are martingales, since their payoffs depend solely on the current stock price. For European-style options with payoff $G_n$, the value of the option at time $k$ is

$$V_k = (1 + r)^k E[(1 + r)^{-n} G_n | X_0, ..., X_k]$$

This expectation is taken with respect to a transformation of the original probability measure $P$, known as the equivalent martingale measure. [3] For path-independent options, $E[G_n | X_0, ..., X_k] = E[G_n]$.

Therefore, just as plain-vanilla options can be priced with backward induction, an Markovian option, an option whose payoff $G_k = g(S_k)$ at time $k$, can also be priced with the backward induction algorithm given earlier, according to the backward-recursion equation given below.

$$V_n(S_n) = g(S_n)$$

$$V_k(S_k) = \max g(S_k), \frac{1}{1 + r} (pV_{k+1}(uS_k) + (1 - p)V_{k+1}(dS_k))$$

Path-dependent options, however, depend on the sequence of $X_i$, tracing the path through the tree and generally cannot be priced in the same way. The following theorems summarize the arguments presented in this section. Chalasani, Jha, and Saisias formalized the difficulty of pricing path-dependent options in their paper, "Approximate Options Pricing." [5] They conclude that pricing of such options belong to a class of problems, known as $\#P$-hard. The class $\#P$-hard, loosely speaking, includes problems of the order of difficulty of counting the number of satisfying assignments to Boolean formula in disjunctive-normal form. The problem of finding such a satisfying assignment is a NP-complete problem. Intuitively, the class $\#P$-hard consists of all counting problems associated with the decision problem in NP. [10]

**Theorem 2** The problem of pricing a European option with polynomially-specified payoff function $G_n$ is $\#P$-hard. [5]

**Proof:** The proof proceeds by reducing the problem of counting the number of perfect matching in a given graph to the pricing problem. The perfect matching counting problem is known to be $\#P$-hard.

If a path-dependent European option with expiration time $n$ has payoff $G_n$ with

$$G_n(\omega) = \begin{cases} (5/2)^n & \text{if } \{e_i; \omega_i = H\} \text{ is a perfect matching of } J \\ 0 & \text{otherwise} \end{cases},$$
where \( J \) is a graph, \( e_i \) is an edge in the graph, and \( \omega_i \) is a path in the graph given by a series of 1s and -1s, corresponding to an up-tick and a down-tick, respectively. The exponential payoff function can be computed in polynomial time.

If \( u = 2 \) and \( r = 0.25 \) such that \( p = q = \frac{1}{2} \) and \( 1 + r = 5/4 \), then every path \( \omega \) has probability \( \Pr(\omega) = \left(\frac{1}{2}\right)^n \). The value of the option, therefore, is

\[
V = (1 + r)^{-n} \sum_{\omega \in \{1,-1\}^n} \Pr(\omega)G_n(\omega) = (4/5)^n \left(\frac{1}{2}\right)^n \sum_{\omega \in \{1,-1\}^n} G_n(\omega) = (2/5)^n (5/2)^n M(J) = M(J)
\]

where \( M(J) \) is the perfect matching in the graph. Therefore, if we can compute \( V \) exactly in polynomial time, it also determines the number of perfect matchings in a graph.

**Theorem 3** If an option has payoff process \( \{G_k\}_{k=0}^n \) where \( G_k = g(C_k) \) such that \( C_k \) is an adapted Markov process such that for each \( k \), the set of different possible values of \( C_k \) is computable in time polynomial in \( n \), then the value \( V \) of the option can be computed in polynomial-time in \( n \) through dynamic programming. \([5]\)

**Proof:** A plain European-style option with payoff \( G_n \) has value at time \( k \), given by the binomial tree model, of \( V_k = (1 + r)^k E[(1 + r)^{-n} G_n|\omega], \) \( k = 0, 1, ..., n \) where \( \omega \) is a price path. Therefore, \( V_n = G_n \) and \( V = V_0 = (1 + r)^{-r} E[G_n] \).

If the payoff function \( G_k \) is dependent on an adapted Markov process and can be computed in polynomial time, then calculating the option’s value is the same as above, since the process’s value only depends on the current price. Therefore, if the payoff can be computed in polynomial time, then the entire pricing tree can be computed in polynomial time.

Unlike path-dependent options with polynomially computable payoff functions, certain exotic options, such as Asian and basket options, have payoffs that must be evaluated over an exponential number of paths. Therefore, backwards induction and dynamic programming cannot be used to price such options efficiently.

### 3.2 Alternative Pricing Methods for Exotic Options

Before we consider specific deterministic approximation algorithms to price exotic options, we briefly describe the application of Monte Carlo algorithms to pricing such options. Monte Carlo methods were first applied to pricing assets by Phelim Boyle, based on their application to option pricing. \([3]\)

Monte Carlo algorithms work well in options pricing, because they “break the curse of dimensionality.” As stated earlier, options pricing can be viewed as a dynamic programming problem. When options are Markovian, they can be solved directly by linear programming or approximated by the Monte Carlo technique. This technique reduces the time complexity of the problem, while sacrificing accuracy. The real advantage of the Monte Carlo method, however, arises when applied to non-Markovian options. In such options, the problem again can be phrased as a dynamic programming problem, but the price of the option at a particular time now depends on the path taken to reach the current price, a non-Markovian problem. Any problem of this form, where the payoff function cannot be computed in polynomial time, suffers from the curse of dimensionality. In other words, the time complexity of the pricing problem is exponential in the number of days over which the option is priced. Although path-dependent options do not belong to the class of Markovian
dynamic programming problems, their structure is very similar and Monte Carlo methods are very successful in solving such problems, although without the performance that is found in Markovian option pricing problems. [13]

The Monte Carlo approach generally follows these steps:

**Algorithm 2 Monte Carlo Option Pricing**

1. Simulate the sample paths within the binomial pricing tree, according to the equivalent martingale measure.
2. Evaluate the discounted expected value of the option on each sample path.
3. Average the discounted expected value over the sample paths.

Monte Carlo algorithms, therefore, price exotic options by sampling paths in the binomial pricing trees and taking an average over the sample. This method applies to all options, whether they are path-dependent, basket, or plain vanilla. While Monte Carlo techniques are practical, the error bounds on the approximation are unknown and the algorithm may not run quickly. Obviously, a trade-off exists between accuracy of the approximation and the time required to price the option.

Rust, in his paper *Using Randomization to Break the Curse of Dimensionality*, suggests an alternative method for pricing non-Markovian, path-dependent options. In any arbitrage-free environment, any path-dependent option can be expressed as a dynamic programming problem, similar to the pricing of a Markovian option, under a particular equivalent martingale measure, as stated previously. [8] For example, if we adopt a discrete-time framework with a Markov transition density \( p(S_{t+1}|S_t) \), then an Asian option can be described by the average stock price as a state variable, \( I(t) \), determined by \( I(t) = \frac{(t-1)I((t-1)+S(t))}{t} \). The value of the Asian option, therefore, is given by the dynamic program

\[
V(S_t, I_t) = \max \left[ \max(I_t - X, 0), \frac{1}{1+r} EV_{t+1}(S_t, I_t) \right]
\]

where

\[
EV_{t+1}(S_t, I_t) = \int V_{t+1} \frac{tI_t + S'}{t + 1} p(S'|S_t) dS'.
\]

Using numerical quadratures to approximate the integral giving the conditional expectation, we obtain a close approximation to the price of an Asian option. Evaluating the average becomes the only issue, since the average is not Markovian. Rust, however, suggests sampling possible average points at expiration and fitting a regression surface, giving the stock price and the average stock price at different points in time. This regression surface then provides a polynomial-time approximation for the path-dependent state variable, the average, and allows us to solve the dynamic programming problem in polynomial time. The time required for this algorithm is of order \( O(n^2) \) with error inversely proportional to the number of points evaluated to obtain the regression surface for the average. [13] This method can be extended generally to all path-dependent options, Markovian or not. [8]

Many other methods, such as the ones described above, exist to approximate the pricing of exotic options where exact solutions do not exist. Some solutions, such as Hull and White and the Levy-Turnbull-Wakeman approximation for Asian options, approximate the distribution of the exotic payoff function or the state variable, \( I(t) \), with other distributions that are easily computable. Although such approximations are widely used, we focus on algorithms on the binomial tree, because they are simpler to understand, easier to derive error bounds, and simple to implement.
due to discrete structured nature of the tree. Furthermore, the binomial tree model is conducive to analytical techniques commonly found in computer science.

4 Asian Options

4.1 Basics of Asian Options Pricing

As discussed earlier, Asian options are options whose value is based on the average of the underlier’s price throughout the life of the option. This condition on the value, of course, make Asian options path-dependent.

Asian options are popular options, particularly among commodity traders. For example, when a commodity is purchased all at once, and used consistently throughout the year, an Asian option on the commodity is commonly used to hedge against price fluctuations during the course of the year. Asian options have similar applications for investment managers when hedging equity portfolios. [11] A pension may purchase an average price put option on the stock market through an Asian index put to guarantee an average value of the pension to match the market.

Asian options also occur in different variations, some of which are difficult to price. Average strike options have a strike price equal to the average of the stock price throughout the life of the option, while average rate options have the same payoff as plain vanilla options where the asset price is replaced by the average of the stock price. Averaging can be performed arithmetically or geometrically, and price sampling can occur continuously or discretely. In practice, only discrete price monitoring is possible, but we shall only consider continuous monitoring.

Geometric averaging Asian options can be priced used the CRR binomial method, because the geometric average, resulting from the product of lagged underlier price, is itself lognormal. Since the payoff function is lognormal, it can be represented accurately on the binomial tree or can even be priced with a closed-form solution to the Black-Scholes-Merton differential equation. [14]

The value of an Asian option depends on the stock price $S$, the time to maturity $t$, and the average of the stock price, $I$. When the price of the Asian option must be calculated according to all three dimensions, determining the price becomes a computationally hard problem. Some Asian options, however, can have one of the dimensions reduced into one of the other two. Efforts to study the valuation of average rate options generally fall into three categories.

1. Approximations of the arithmetic average by the geometric average, which is more tractable, due to the preservation of lognormal properties of the underlier. Similarly, some methods approximate the true distribution of the price average by some approximate distribution, such as the lognormal distribution.

2. Approximations of the underlying motion of the average of the stock price to rederive a closed-form solution, similar to the Black-Scholes formula.

3. Numerical methods that estimate the true value of the expected value of the payoff function. We will focus on such numerical techniques, because they are used in practice more than the previous two approaches. [1]

The time complexity of pricing Asian options is not known, according to Chalasani, but we have some idea that the problem should be in NP-hard. [5] This intuition relies on the idea that an Asian option does not have polynomially decidable payoff functions and must be evaluated along every path in the tree.

10
Conjecture 4 The pricing of arithmetic-averaging Asian options is NP-hard.

Proof: We do not develop an explicit proof. Instead, we observe that the only way to determine the exact price of an Asian arithmetic-average option would be to follow all possible price paths. In the simplest binomial model, such an algorithm would imply following $2^n$ paths. Furthermore, unlike the product of lognormal variables, the sum of lognormal variables does not itself have a lognormal distribution. Therefore, the methods that apply in determining the closed-form solution of the geometric averaging option does not work for arithmetic averaging options.

Throughout the rest of this paper, we examine algorithms developed by Akcoglu, et al, in their paper, Fast Pricing of European Asian Options with Provable Accuracy: Single-stock and Basket Options, including implementations that I developed to illustrate some of the theoretical results in their paper as part of their research.

4.2 A Monte Carlo Algorithm with Known Error Bounds

The problem of pricing an Asian average rate option reduces to finding $E[\max(A_n(\omega) - X, 0)] = \frac{1}{n+1}E[\max(T_n(\omega) - (n+1)X, 0)]$, where $A_n(\omega)$ is the running average and $T_n(\omega)$ is the running total of stock prices over path $\omega$. As discussed earlier, Monte Carlo techniques can be used to price path-dependent options, such as Asian options, although they are not Markovian. Such methods involve randomly sampling paths $\omega \in \Omega$ according to some distribution $\Pi$ and computing the average of the payoff functions, $A_n(\omega) - X)^+ = \max(A_n(\omega) - X, 0)$. If $N$ paths $\omega^1, ..., \omega^n$ are sampled from $\Omega$, then the Monte Carlo price estimate of the call is

$$\mu = \frac{1}{N} \sum_{i=1}^{N} (A_n(\omega^i) - X)^+$$

The accuracy of the estimate depends on the number of simulations $N$ and the variance $\tau^2$ of the payoff; the error bound typically guaranteed by Monte Carlo methods is $O(\tau / \sqrt{N})$. The variance is usually unknown and must be estimated to determine the error bound of the price estimate. [2]

In their paper, Akcoglu, et al, develop a Monte Carlo method whose error bound is analytical and not dependent on an estimated variance. From the properties of martingales and Chernoff bounds, they present an estimate of $E[\max(T_n(\omega) - (n+1)X, 0)]$ where $A_n(\omega)$ with Monte Carlo sampling.

Theorem 5 Let $0 < \delta < 1$ be given. With $N = \Theta(\log \frac{1}{\delta} + \frac{1}{\epsilon^2} e^{A_0 \lambda_0 (1 + 2\epsilon)} \frac{1}{\lambda_0 - 2\epsilon})$ trials, the following statements hold.

1. If $\frac{X}{N} \leq 2\epsilon$, then with probability $1 - \delta$, $E(T_n - (n+1)X)$ estimates $E((T_n - (n+1)X)^+)$ with error at most $4\epsilon(n+1)X$. Correspondingly, the price of the call is estimated with error at most $4\epsilon X$.

2. If $\frac{X}{N} > 2\epsilon$, then, with probability $1 - \delta$, $\frac{1}{N} \sum_{i=1}^{N} (T_n(\omega^i) - (n+1)X)^+$ estimates $E((T_n - (n+1)X)^+)$ with standard deviation at most $\epsilon(n+1)X$. Correspondingly, the price of the call is estimated with standard deviation at most $\epsilon X$.

With the results above, Akcoglu et. al, present the following algorithm.
Algorithm 3 \textit{BoundedMC}(\delta, \epsilon)

1. Generate \( N = \Theta\left(\frac{1}{\delta} + \frac{1}{\epsilon^2} e^{4\sigma^2 \lambda_0 \frac{1+2\sigma^2}{\lambda_0 - 2\sigma^2}}\right) \) paths.

2. Let \( Z \) be the number of paths \( \omega^i \) such that \( T_n(\omega^i) \leq (n + 1)X \).

3. If \( Z/N \leq 2\epsilon \), return \( \frac{1}{n+1} E(T_n - (n+1)X) = \frac{1}{n+1} (E(T_n) - (n+1)X) \) else return \( \frac{1}{n} \sum_{i=1}^{N} (A_n(\omega^i) - X)^+ \)

The expected value of the running sum can be computed as follows:

Theorem 6 \textit{If} \( S_0 \) \textit{is the initial stock price and if} \( r \) \textit{is the risk-free interest rate, then}

\[
E(T_n) = \begin{cases} 
(n+1)S_0 & \text{if } r = 0, \\
\frac{(1+r)^{n+1}-1}{r}S_0 & \text{if } r > 0
\end{cases}
\]

4.3 Bucketed Tree Traversal

Chalasani, et al., developed the first numerical algorithm to approximate the price of an Asian option. Until their algorithm, most Asian option approximations concentrated on the other two aspects of pricing Asian options, namely approximating the arithmetic average payoff by a geometric average payoff function or approximating the distribution of the average payoff to obtain a closed-form solution. Their algorithm for European Asian call options estimates the expected value of the payoff function \( E(A_n - X) \) with

\[
\sum_{k=0}^{n} p^k q^{n-k} C(n, k) [E(A_n | Y_n = k) - X]^+
\]

where \( Y_n \) is the node position equal to the sum of \( X_t \) over a path. This so-called path clustering algorithm runs in time \( O(n^3) \). [5]

Aingworth, et al., and Akcoglu, et al., develop similar algorithms that run theoretically faster than Chalasani’s algorithm with error bounds that are analytical, as well. [1] The basic idea behind Kao’s BTT algorithm is that we do not need to calculate all the path sums or averages in the binomial tree, a process that makes the CRR pricing of Asian options exponentially time consuming. Instead, the intuition behind their algorithm is to maintain a set of “buckets” that stores the probability that the running average \( A_n \) exceeds the strike price \( X \) or that the running total \( T_n \) exceeds the barrier \( (n + 1)X \). To estimate the value of the option along all paths, we need only know the probability that the path has a sum that exceeds the barrier \( (n + 1)X \). If so, the average exceeds the strike. The BTT algorithm calculates such probabilities by traversing the binomial tree only once, instead of calculating all possible path sums. As the algorithm traverses the tree, it also makes a local estimate of the amount by which a sum along a path exceeds the barrier. After the “buckets” are populated with probabilities and the approximate excess sums are calculated, the price of the Asian call option is simply the sum of the excess sums weighted by the probability that the path over which the sum is obtained exceeds the barrier. On cursory inspection, this algorithm runs in \( O(n^2) \), an obvious improvement over Chalasani’s running time.

Given a binomial tree \( T \) of depth \( n \), \( T_m[t,i] \) is a subtree of depth \( m \) rooted at node \( T[t,i] \). If \( \omega|_t \) is a prefix of the path \( \omega \) up to level \( t \) of \( T \), then, given \( \psi, \omega \in \Omega, \psi \) is an extension of \( \omega|_m \) if, for
0 \leq t \leq m$, $\phi_t = \omega$. If another binomial tree $U$ has depth $n$ and $\phi \in \Omega(U)$, then $\phi$ is isomorphic to $\omega \in \Omega(T)$, if, for all $0 \leq t \leq n$, $\phi_t = \omega_t$.

The BTT algorithm is based on the following observation about path extensions. If the running total $T_m(\omega) = T_m(\omega|_m)$ of stock prices on path $\omega \in \Omega$ exceeds the total barrier $B = \omega(1 + n)X$, then, for any extension $\phi$ of $\omega|_m$, $T_n(\phi)$ also exceeds $B$ and the call will be exercised. Since the call will be exercised for all extensions of $\omega|_m$, the payoff of the call is easily computed. Therefore, as we travel a path $\omega$, once the running total $T_m(\omega)$ exceeds $B$, the running total on extensions $\phi$ of $\omega|_m$, weighted by $\Pi(\phi)$ determines the value of the option. The option price, therefore, depends on the number of path prefixes $\omega|_m$ that have running totals less than the barrier $B$.

Although there exist an exponential number of possible path prefixes $\omega|_m$ with running totals less than the barrier, the running totals $T_m(\omega|_m)$ are in the bounded range $[0, B)$. The running totals, therefore, that terminate at each node can be grouped into boxes that divide the bounding interval. If the left endpoint of each subdivided interval is used to represent the summed value in the bucket, then an error of at most $\frac{B}{\kappa}$ is introduced. At the end leaf nodes of the tree, the total sum $T_n(\omega)$ of a path $\omega$ is underestimated by at most $\frac{B^2}{\kappa}$ and the average $A_n(\omega)$ is underestimation by at most $\frac{B}{\kappa}$.

In the bucketed tree traversal (BTT) algorithm developed by Akcoglu, et al, a set of $k + 1$ buckets is created at each tree node $v = T[t,i]$ of $T$ that store that partial sums of path prefixes that terminate at that node. For $0 \leq j < k$, core bucket $b_j(v)$ stores the probability mass $b_j(v).mass$ of the path prefixes that terminate at node $v$ and have running totals in its range $[j \frac{B}{\kappa}, (j+1) \frac{B}{\kappa})$. The representative value in the core bucket $j$ is $b_j(v).value = j \frac{B}{\kappa}$. The extra bucket, the overflow bucket $b_k(v)$, stores the probability-weighted running total estimates of path prefixes that have estimated running totals greater than the barrier $B$. [2]

**Algorithm 4 BTT(T, k, B)**

for each node $v \in T$ and each bucket $b_j(v)$

set $b_j(v).mass \leftarrow 0$;

take $j$ such that initial price $s(T[0,0]) \in range(j)$; $b_j(T[0,0]).mass \leftarrow 1$; % assume $s(T[0,0]) < B$

for $t = 0, \ldots, (n-1)$ % iterate through each level

for $i = 0, \ldots, t$ % iterate through each node at level $t$

let $v = T[t,i]$; % shorthand notation for node $T[t,i]$

for $w \in \{T[t+1,i], T[t+1,i+1]\}$ % for each child of node $v$

let $p' \in \{p, q\}$ be the probability of going from node $v$ to $w$;

for $b_j(v) \in \{b_0(v), \ldots, b_k(v)\}$ % for each bucket at node $v$

let $V \leftarrow b_j(v).value + s(w)$;

let $M \leftarrow b_j(v).mass \times p'$;

if $V < B$

take $\ell$ such that $V \in range(\ell)$;

$b_\ell(w).mass \leftarrow b_\ell(w).mass + M$;

else % in overflow bucket

$b_k(w).mass \leftarrow \frac{b_k(w).mass \times b_k(w).value + M \times V}{b_k(w).mass + M}$;

$b_k(w).mass \leftarrow b_k(w).mass + M$;

return $\sum_{i=0}^{n} b_k(n,i).mass \times (b_k(n,i).value - B)$.
% return option price estimated from overflow buckets at leaves

**Theorem 7** The algorithm BTT has total running time of $\Theta(kn^2)$.
Proof: Assuming that updating the bucket values take a constant amount of time, the BTT algorithm propagates the value and probability masses of the buckets at each node of the tree. Since it performs \( k + 1 \) operations at each node, the running time is \( \Theta(kn^2) \).

### 4.4 Recursive Bucketed Tree Traversal Algorithm

Akcoglu, et al, propose a recursive extension to the BTT algorithm that can be generalized to other pricing algorithms on a binomial tree. First, we describe the algorithm with respect to BTT as the base pricing algorithm. Then, we discuss how the recursion used in RecBTT can be applied with other pricing algorithms as the base case.

The recursive BTT algorithm generates the same probability distribution in buckets that BTT generates, but it obtains the probability distribution quickly by not traversing the entire tree. It achieves this by estimating certain buckets for certain nodes that comprise a subtree within the tree. For a set of nodes with buckets in the tree, the set of nodes that includes one node higher than the given set has essentially similar buckets with a similar probability distribution. Therefore, once we determine the buckets and the probability distribution for the lower set of nodes, the higher set of nodes have a distribution that is similar within a constant factor. The recursive BTT uses this fact to generate bucket weights for the tree in time faster than BTT.

Recursive BTT performs the BTT algorithm on smaller subtrees within the given option pricing tree and estimates the probability distribution of the average at certain nodes by the probability distribution from nodes determined by BTT. For example, for any node \( T[t, i] \) in the tree, the price at the node directly above it is \( s(T[t, i + 1]) = u^2s(T[t, i]) \). Furthermore, if we considered the subtrees rooted at \( T[t, i] \) and \( T[t, i + 1] \), the corresponding nodes in both trees differ by a factor of \( u^2 \). The probability distributions of the two subtrees, considered without respect to the original pricing tree, are the same. The only difference between the subtree nodes considered independently and considered within the tree is the probability of reaching either \( T[t, i] \) or \( T[t, i + 1] \). These observations motivate the algorithm. The algorithm first runs BTT to some level in the tree. It then alternately runs BTT or estimates the probability distributions at different nodes in the tree, merging the results with the original tree.

BTT uses the weights of all path prefixes in some buckets at some terminating node at level \( T \) to compute the bucket weights for nodes at level \( t + 1 \). RecBTT, instead, recursively solves the problem for subtrees \( T_m^{[t,i]} \), \( 0 \leq i \leq t \) of some depth \( m < n - t \) rooted at node \( T[t, i] \). As each recursive call is complete, RecBTT merges the bucket weights at the leaves of \( T_m^{[t,i]} \) into the corresponding nodes at level \( t + m \) of \( T \). This recursive method provides two distinct advantages over the BTT algorithm.

1. The resulting tree uses finer bucket granularity, resulting in improved accuracy.

2. The results of a single recursive call on a particular subtree \( T_m^{[t,i]} \) are used to estimate the results of other recursive calls to other subtrees \( T_m^{[j,i]} \) where \( j > i \) as long as the node prices in \( T_m^{[h,i]} \) are “sufficiently close” to the corresponding node prices in \( T_m^{[t,i]} \). This estimation improves the runtime, since not all the \( t + 1 \) of the recursive calls to the pricing algorithm are made at the level \( t \). Some parts of the tree \( T \) are not traversed directly. [2]

#### 4.4.1 The Merge Procedure

If BTT is applied on the subtree \( T_1 = T_n^{[i_1, i_0]} \) of depth \( n_1 \) rooted at \( v_0 = T[i_0, i_0] \), then a leaf \( v_1 = T_1[n_1, i_1] \) \( (0 \leq i_1 \leq n_1) \) of \( T_1 \) corresponds to the node \( v_2 = T[i_0 + n_1, i_0 + i_1] \) of the original
tree $T$. The Merge procedure combines the bucket weights developed for the leaf node of the smaller subtree into the corresponding node of the larger tree. Since we wish to achieve greater accuracy with the smaller subtree, we must have greater bucket granularity when applying BTT to the subtree than when we apply BTT to the large tree. If the algorithm uses $k_1 = h_0 k_0$ core buckets for the subtree, then each group of $h_1$ buckets at $v_1$ must be combined into a single bucket at $v_2$ such that $v_2$ only has $k_0$ buckets.

If $0 \leq j_0, j_1, j_2 < k_0$ are core bucket indices where $k_0$ is the total number of core buckets, then bucket $b_{j_0}(v_0)$ stores the mass of path prefixes in $T$ terminating at $v_0$ whose running total estimates fall into the interval $[j_0 \frac{B}{k_0}, (j_0 + 1) \frac{B}{k_0})$. The mass in buckets of $v_1$ in a particular interval corresponding to the same mass in the same interval for $v_2$. Merging $T_1$ into $T$ involves combining each $v_1$ of $T_1$ into the corresponding node $v_2$ of $T$. After merging, the mass of the node buckets are updated to contain the weights of path prefixes in $T$ passing through $v_0$ and terminating at $v_2$. The overflow buckets are handled in an analogous manner.

\textbf{Algorithm 5} \texttt{Merge}($T$, $T_1 = T_{h_1}^{[j_0, j_0]}$)

\begin{enumerate}
\item let $v_0 \leftarrow T[t_0, i_0]$;
\item for $i_1 = 0, \ldots, n_1$ \% for each leaf of $T_1$
\begin{enumerate}
\item let $v_1 \leftarrow T_1[n_1, i_1]$,
\item $v_2 \leftarrow T[t_0 + n_1, i_0 + i_1]$;
\end{enumerate}
\item for $j_0 = 0, \ldots, k_0$ \% buckets in $v_0$
\begin{enumerate}
\item for $j_1 = 0, \ldots, k_0$ \% buckets in $v_1$
\begin{enumerate}
\item let $V \leftarrow b_{j_0}(v_0).value + b_{j_1}(v_1).value$;
\item let $M \leftarrow b_{j_0}(v_0).mass \times b_{j_1}(v_1).mass$;
\item if $V < B$
\begin{enumerate}
\item take $j_2$ such that $V \in \text{range}(j_2)$;
\item $b_{j_2}(v_2).mass \leftarrow b_{j_2}(v_2).mass + M$;
\end{enumerate}
\item else
\begin{enumerate}
\item $b_{k}(v_2).value \leftarrow \frac{b_{k}(v_2).mass \times b_{k}(v_2).value + M \times V}{b_{k}(v_2).mass + M}$;
\item $b_{k}(v_2).mass \leftarrow b_{k}(v_2).mass + M$.
\end{enumerate}
\end{enumerate}
\end{enumerate}
\end{enumerate}

The price at $V_0$ is counted, once during the path prefix from the root and once in the partial path from $v_0$ to $v_2$. When the problem is solved recursively on subtree $T_1$, the price at the root of the subtree should be set to 0 to ensure a given price is counted once.

\textbf{Theorem 8} If, for an arbitrary node $v$, $E(v)$ is the maximum amount by which running totals terminating at node $v$ are underestimated by the bucket values, then, using the Merge algorithm, $E(v_2) \leq E(v_0) + E(v_1) + \frac{B}{k_0}$.

\textbf{Theorem 9} Merge can be made to run in $O(n_1 k \log k)$ time.

\textbf{Proof:}

This result arises from performing the Merge algorithm with the Fast Fourier Transform, instead of the algorithm described above. The Fast Fourier Transform achieves all possible products of all possible bucket masses in time $\Theta(k \log k)$. Since the Merge procedure is performed for $n_1$ nodes in each subtree, the resulting running time is $O(n_1 k \log k)$.

\[\square\]
4.4.2 The Estimate Procedure

If \( T_1 = T_{[t_0,i_1]} \) and \( T_2 = T_{[t_0,i_2]} \) are two subtrees of at the same level \( t_0 \) of the binomial tree \( T \) where \( i_2 > i_1 \), then the ESTIMATE procedure estimates the weights in the leaf buckets of \( T_2 \) from the weights in the leaf buckets of \( T_1 \). The procedure ESTIMATE exploits the fact that, given any \( v_1 = T_1[t,i] \), it has corresponding node \( v_2 = T_2[t,i] \) in tree \( T_2 \). For some constant \( \alpha > 1 \), \( s(v_2) = \alpha s(v_1) \) for all such pairs \((v_1,v_2)\). Therefore, for any path \( \psi \in \Omega(T_2) \), \( T_m(\psi) = \alpha T_m(\omega) \) where \( \omega \in \Omega(T_1) \) is isomorphic to \( \phi \).

Algorithm 6 \( \text{ESTIMATE}(T_2 = T_{[t_0,i_2]}, T_1 = T_{[t_0,i_1]}) \)

for \( i = 0, \ldots, m \) go through the leaf buckets of \( T_1 \) and \( T_2 \)

let \( v_1 \leftarrow T_1[m,i] \), \( v_2 \leftarrow T_2[m,i] \);

for \( j_1 = 0, \ldots, k \) go through each bucket at \( v_1 \)

let \( V \leftarrow \alpha b_{j_1}(v_1).\text{value} \);

let \( M \leftarrow b_{j_1}(v_1).\text{mass} \);

if \( V < B \)
take \( j_2 \) such that \( V \in \text{range}(j_2) \);

\( b_{j_2}(v_1).\text{mass} \leftarrow b_{j_2}(v_2).\text{mass} + M \);

else

\( b_k(v_2).\text{value} \leftarrow \frac{b_k(v_2).\text{mass} \times b_k(v_2).\text{value} + M \times V}{b_k(v_2).\text{mass} + M} \);

\( b_k(v_2).\text{mass} \leftarrow b_k(v_2).\text{mass} + M \);


Theorem 10 If \( \alpha \leq 2 \) and the total path sums in \( T_1 \) are underestimated by bucketing by at most \( E \), then \( \text{ESTIMATE} \) underestimates that total path sum in \( T_2 \) by at most \( 2E + 2\frac{B}{k} \).

Theorem 11 Suppose that we would like to determine the leaf bucket weights of the subtrees \( T_{[t,i]} \) where \( 0 \leq i \leq t \). We need only call \( \text{RecBTT} \) at most once for every \( \Theta(\sqrt{n}) \) subtrees, and use the \( \text{ESTIMATE} \) procedure to estimate the leaf bucket weights of the other subtrees with bounded error.

4.4.3 Error and Runtime Analysis

We now derive the analytical error bound and running time of the algorithm \( \text{RecBTT} \). If we simulate \( n_0 \) trading periods with a tree of \( k_0 \) buckets per node, then we can estimate the error, which is the amount by which \( \text{RecBTT} \) underestimates the running totals. We recursively call \( \text{RecBTT} \) on binomial trees of decreasing depth (decreasing \( n_i \)) with an increasing number of buckets (increasing \( k_i \)), thereby determining the error \( E_i = E(n_i,k_i) \) and the running time \( T_i = T(n_i,k_i) \) of the \( i \)-th subproblem. [2]

Theorem 12 When \( i \) recursive calls are made, the error made by \( \text{RecBTT} \) is

\[
E_0 = 5Bn_0 \sum_{j=0}^{i} \frac{2^j}{k_jn_{j+1}} + \frac{2^{i+1}n_0}{n_{i+1}} \bar{E}_{i+1},
\]

where \( \bar{E}_{i+1} \) is the error in solving the \((i+1)\)-st subproblem.

Proof: Consider the error in the \( i \)-th subproblem, \( E_i = E(n_i,k_i) \). Each recursive call has an error of \( E_{i+1} \). When the \( k_{i+1} \) smaller granularity buckets in the \((i+1)\)-st problem are grouped into the \( k_i < k_{i+1} \) buckets of the \( i \)-th problem, an error of one bucket size, \( \frac{B}{k_i} \), is introduced. A call
to ESTIMATE underestimates the price sums by at most \(2(E_{i+1} + \frac{B}{k_i}) + 2\frac{B}{k_i} = 4\frac{B}{k_i} + 2E_{i+1}\), by estimation accuracy result described above. This error is made at most \(\frac{n_i}{k_i} \) times, once at each of levels \(n_{i+1}, 2n_{i+1}, \ldots, n_{i+1}\). At each of these levels, an additional error of \(\frac{B}{k_i}\) is introduced by the Merge procedure. The total error in the \(i\)-th subproblem is

\[
E_i = \frac{5Bn_i}{k_i n_{i+1}} + \frac{2n_i}{n_{i+1}} E_{i+1}.
\]

Unraveling \(i\) levels of the recursion on the original problem yields the result.

To determine the total analytic runtime of the algorithm, we must also determine the number of times the \((i+1)\)-st subproblem needs to be solved when solving the \(i\)-th subproblem. At each level in the binomial tree of the \(i\)-th subproblem, suppose the \((i+1)\)-st subproblem is called at most \(C_i\) times.

**Theorem 13** When \(i\) recursive calls are made, the runtime of RecBTT is

\[
T_0 = O\left(\sum_{j=0}^{i} n_0 n_j k_j \log(k_j) \prod_{k=0}^{j-1} \frac{c_k}{n_{k+1}} + \frac{n_0 \prod_{k=0}^{i} c_k}{n_{i+1}} \tilde{T}_{i+1}\right),
\]

where \(\tilde{T}_{i+1}\) is the time it takes to solve the \((i+1)\)-st subproblem.

With the running times and recursion depth developed above for the subproblems, we develop these for the running time and accuracy of the algorithm.

**Theorem 14** Given integer \(R > 2\), let \(\gamma = \frac{1}{R}\), and for \(i > 0\), let \(n_i = \left(\frac{n_i}{n_0}\right)^{1/2 - i\gamma} \) and \(k_i = 4^i k_0 \left(\frac{n_i}{n_0}\right)^{i\gamma}\), where \(\sigma\) is the volatility of the stock. recBTT underestimates \(E\left((T_n - (n + 1)X)^{+}\right)\) by at most \(O\left(\frac{Bn_0^{1/2 + \gamma} \sigma^{1 - 2\gamma}}{k_0}\right)\) and takes time \(O\left(2^{1/\gamma} n_0^2 k_0 \left(\frac{1}{\gamma} + \log \frac{k_0}{\sigma}\right)\right)\).

**Proof:** After \(K = \frac{1}{2\gamma}\) levels of recursion, we have \(n_K = \left(\frac{n_0}{n_0}\right)^0 = 1\). The problem consists of a single node and can be solved in constant time \(\tilde{T}_K\) with an error of \(\tilde{E}_K = \frac{B}{k_K} = \frac{\sigma B}{4^{K_0} k_0 \sigma^{1/2}}\). From Theorem 12, the error is

\[
E_0 = 5Bn_0 \sum_{i=1}^{K} 2^{i-1} \frac{n_i}{k_i n_{i+1}} = 5Bn_0 \sum_{i=1}^{K} \frac{1}{2^{i-1} k_0 \left(\frac{n_i}{n_0}\right)^{1/2 - i\gamma}} + 2^K n_0 \sigma B \frac{2^K n_0 \sigma B}{4^{K_0} k_0 n_0^{1/2}} = \left(\sigma^{1 - 2\gamma} Bn_0^{1/2 + \gamma}\right) \frac{1}{k_0}.
\]

For the runtime analysis, by Theorem 11, note that \(c_0 = \frac{n_0}{\sqrt{n_0}} = \sigma \sqrt{n_0}\). For \(i > 0\), since \(n_i < \frac{\sigma}{\sqrt{\sigma}}\), \(c_i = 1\). Combining this fact with Theorem 13 yields the runtime

\[
T_0 = n_0^2 k_0 \log k_0 + O\left(\sum_{i=1}^{K} n_i c_i k_i \log k_i\right) + O(n_0 c_0) = n_0^2 k_0 \log k_0 + O\left(\sum_{i=1}^{K} n_0^{2i} k_0 \left(\frac{1}{\gamma} + \log \frac{k_0}{\sigma}\right)\right) = O\left(2^{1/\gamma} n_0^2 k_0 \left(\frac{1}{\gamma} + \log \frac{k_0}{\sigma}\right)\right).
\]
Theorem 15 Corollary
Given integer $R > 2$, let $\gamma = \frac{1}{R}$ and choose $n_i$ and $k_i$ as in Theorem 14. In time $O\left(2^{1/\gamma}n_0^{2k_0}(\frac{1}{\gamma} + \log \frac{k_0}{\varepsilon_0})\right)$, recBTT returns an option price in the range $[P - \frac{Xn_0^{1/2+\gamma}\sigma^{1-2\gamma}}{k_0}, P]$, where $P$ is the exact price of the option.

Proof: This follows from Theorem 14 and the fact that

$$E\left((A_n - (n + 1)X)^+\right) = \frac{1}{n + 1}E\left((T_n - (n + 1)X)^+\right).$$

\[\square\]

4.5 Empirical Testing of BTT and RecBTT

In this section, we present results from our implementation of the BTT and recBTT algorithms and compare them with the exponential tree traversal algorithm (exhaustive pricing) that computes the price of an Asian option exactly. We also compare our algorithms with the path-clustering algorithm developed by Chalasani et al.

The implementation of the BTT algorithm was written in C on an Intel Pentium II machine running Redhat Linux. The BTT and recBTT algorithms, as well as the exhaustive pricing algorithm, are presented in the Appendix. Although the algorithms themselves are intuitive and easily seen on a binomial tree, they cannot be implemented on a tree, due to time and space constraints. Instead, the implementations both use an array of arrays that represents the nodes at a particular level in the tree with their respective “buckets”, containing the probability distribution of the average at that particular node. This method of implementation greatly reduces the time and space necessary to run the algorithm, although it increases the complexity, as well.

In our first set of experiments, whose results are shown in Table 1, we compare the accuracy and running time of the BTT algorithm and recBTT with that of exhaustive pricing. We used $S_0 = 1.0$, $X = 2.0$, $u = 1.2$, and $p = 0.727$ as the input pricing parameters. As shown in figure 1, exhaustive pricing, as expected, takes an exponential amount of time, not even pricing a tree of depth 50, due to time constraints. The asterisks indicate that the pricing algorithm cannot be run within a practical amount of time. The BTT algorithm prices the option accurately in under a minute. Furthermore, the algorithm appears to be scalable, pricing a tree of depth 50, within four seconds.

The recursive BTT algorithm, however, does not fare as well. Although the algorithm prices the option within five percent error, well within the suggested error bounds, recBTT does not run as quickly as BTT in some cases. This relative poor performance may be due to the overhead required in setting up the data structures for use in the algorithm and not the algorithm itself. This situation, however, seems to improve as the size of the tree grows larger. The recursive BTT implementation, may in fact, price longer-term options, such as LEAPS, and trees with finer granularity better, thereby approaching the continuous-time price solution. Another implementation may be more efficient, obtaining the same accuracy, but with greater speed.

As shown in Table 2, both algorithms price options better than Chalasani’s path-clustering paper. The inputs from Table 2 come from [5]. Both the BTT and recBTT algorithms price options with greater accuracy and within a few minutes, time that is tolerable in practice. The performance of both algorithms does not change with changes in the depth of the tree, the volatility of the option, or the other parameters. Furthermore, the recBTT algorithm’s results are, by and large, well within the error bounds predicted earlier in the paper. Despite the accuracy and
efficiency of recBTT, it still does not match or exceed the performance of BTT. Again, this may be due to the implementation method described earlier.

We can extend the algorithms, as well as their implementations, to the pricing of basket options and European Asian basket options. This will be discussed in the next section.

5 Basket Options

Basket options belong to a larger class of options, known as rainbow or multiasset options. Unlike Asian options, rainbow options are not path-dependent; their complexity arises from the correlation between the multiple assets that comprise their underlying asset. Multi-asset options include such option types as outperformance options, options with values based on the difference of two assets, extreme options, options based on the highest or lowest valued asset, or options based on the product of asset prices. Of all the different types of rainbow options, basket options have the most general application, particularly index options, or those that have a particular stock index as an underlier.

Because the level of a stock index is the sum or the market capitalization weighted sum of the individual stocks in the index, the index level, or its value, does not have a lognormal distribution. Yet, for the sake of simplicity, finance practitioners simply assume that indices have a lognormal distribution and price index options with the Black-Scholes formula. This assumption is obviously wrong depending on the correlation of the stocks in the index. If there is perfect correlation, then the stocks move as one and the index level has the same distribution as any one stock in the index. If the stocks are independent, then the Central Limit Theorem of probability theory predicts that the index level should, in fact, have a normal distribution. In practice, indices lie somewhere between the two extreme cases. Stocks that comprise the popular indices, such as the Dow Jones Industrial Average, the Nasdaq 100, and the Standard and Poor's 500 Index, generally are correlated, because certain features of the underlying companies have some similarities, such as similar market capitalization. These indices, however, are designed to reflect the market as a whole and represent different sectors of the stock market. Therefore, the correlations between index components is not necessarily perfect, either.

In this section, we shall present two results of interest in the pricing of basket options. First, we compare the traditional Black-Scholes pricing of basket options with a polynomial-time binomial tree algorithm. Next, we develop techniques to price basket Asian options based on the algorithms developed earlier in the paper for pricing Asian options.

5.1 Fast Fourier Transform Algorithm

We adapt a similar technique to BTT to price basket options. Basket options are similar to Asian options in that the payoffs of both depend on the sum of lognormal variables, which itself is not lognormal. Just as with Asian options, if we can estimate all possible sums of the prices of the components of the asset basket and the probabilities of the components jointly achieving those sums, then we can find the expected value of \( \sum_{k=1}^{m} S_k \) where each \( S_k \) is the price of a basket component at the option's expiration.

The algorithm generally works in three stages. In the first stage, we generate the binomial tree of stock prices in a method similar to the first part of the backward induction algorithm for each stock in the basket. The binomial tree calculations results in the possible prices of expiration for each stock, as well as the probability distribution that the stock achieves that price. We next generate a set of buckets, representing a set of intervals between 0 and \( X \), the strike price of the
basket option plus one overflow bucket that will contain the probability that the price exceeds the strike. For each price-probability distribution that generated for each stock with the backward induction algorithm, we populate a set of buckets as described earlier with the probabilities of the stock achieving a particular price. Given a set of buckets for each stock, we now multiply the buckets together to generate a bucket set for the entire basket, as well as estimate by how much the basket could exceed the strike price. The estimation of the overflow value in each bucket is similar to the calculation performed for the overflow bucket in BTT.

The product of the probability masses for each bucket set from each stock is done quickly by using the Fast Fourier Transform. The Fast Fourier Transform performs the multiplication of two polynomials quickly by performing the convolution of the polynomial in the frequency domain. [6] The probability distribution produced in the first part of the algorithm is just a set of discrete probabilities, one probability mass for each price interval between 0 and the strike, with some probability for a value greater than the strike. Treating each probability distribution in the buckets as the coefficients of a polynomial, we can easily obtain their products through the Fast Fourier Transform. The implementation of the algorithm is presented in the appendix as EuroBasket.c.

5.2 Pricing Basket Asian Options

Akcoglu, et al., combine the FFT algorithm for pricing European basket options with their bucketing scheme developed for Asian option to price European Asian basket options. The pricing of Asian options is further complicated in the case of basket options, because the pricing algorithm must consider an exponential number of paths for the \( m \) stocks in the basket, as well as the exponential paths in each stock’s tree.

Our European Asian basket call pricing algorithm, BasketBTT is described in Algorithm 7. Let \( B = (n+1)X \), where \( X \) is the strike price of the basket option. For each stock \( z_i \), \( 1 \leq i \leq m \), we use \( \text{reeBTT} \) to construct the bucketed binomial tree structure \( T^i \) described earlier, this time using \( B \) as the barrier; should the running total of any \( z_i \) exceed \( B \), the basket option will always be exercised, regardless of what the other stocks do. For each stock \( z_i \), we construct \( k+1 \) superbuckets \( \beta^i_j \), \( 0 \leq j \leq k \), where \( \beta^i_0 \) is the combination of buckets \( b_j(v) \) for all leaves \( v \in T^i \). For the core buckets \( \beta^i_j \), \( 0 \leq j < k \), let \( \beta^i_j.value = \frac{jB}{k} \) and \( \beta^i_j.mass = \sum_{\ell=0}^n b_j(T^i[n, \ell]).mass \), where this summation ranges over all leaves \( T^i[n, \ell] \) of \( T^i \). For the overflow bucket \( \beta^i_k \), let \( \beta^i_k.mass = \sum_{\ell=0}^n b_j(T^i[n, \ell]).mass \) and

\[
\beta^i_k.value = \frac{\sum_{\ell=0}^n b_k(T^i[n, \ell]).value \times b_j(T^i[n, \ell]).mass}{\beta^i_k.mass}.
\]

Handling overflow superbuckets If the running total of a stock \( z_i \) reaches the overflow superbucket \( \beta^i_k \), the option will be exercised regardless of what the other stocks do. Given this, we can determine the value of the option exactly, since

\[
E((T_n - (n+1)X)_+) = E(T_n - (n+1)X) = E(T_n) - (n+1)X = \beta^i_k.value + \sum_{i \neq i} E(T^i_n) - (n+1)X,
\]

where \( T^i_n \) is the random variable denoting the running total of stock \( z_{i'} \) up to day \( n \). \( E(T^i_n) \) can be computed exactly.

Handling core superbuckets Consider now the core superbuckets \( \beta^i_j \), \( 0 \leq j < k \). Let \( f_i(x) = \sum_{j=0}^{k-1} \beta^i_j.mass \cdot x^j \) be the polynomial representation of the core bucket masses of stock \( z_i \) and let
$f(x) = \prod_{i=1}^{m} f_i(x)$. This product can be computed efficiently, as described in Lemma 16. Notice that $f(x)$ has the form

$$f(x) = b_0x^0 + b_1x^1 + \cdots + b_{m(k-1)}x^{m(k-1)}.$$  

From the definition of $f(x)$, observe that $b_j$ is just the probability that the sum (over all stocks $z_i$) of running totals $T^i_n$ from the core buckets falls in the range $[j\frac{B}{K}, (j+1)\frac{B}{K})$. That is,

$$b_j = \Pr \left( \sum_{i=1}^{n} T^i_n \in \left[ j\frac{B}{K}, (j+1)\frac{B}{K} \right) \mid T^i_n < B \text{ for all } i \right).$$

Hence, the contribution to the option price from the core buckets can be estimated by

$$\sum_{j=k}^{m(k-1)} b_j \left( j\frac{B}{K} - (n+1)X \right).$$

**Pricing the option** Combining the above results for the overflow and the core superbuckets, we see that $E\left( (T_n - (n+1)X)^+ \right)$ can be estimated by

$$\sum_{i=1}^{m} \beta_i^j \cdot \text{mass} \left( \beta_i^j \cdot \text{value} + \sum_{i' \neq i} E(T^i_n') - (n+1)X \right) + \sum_{j=k}^{m(k-1)} b_j \left( j\frac{B}{K} - (n+1)X \right).$$

**Algorithm 7 BasketBTT($z_1, \ldots, z_m, B = (n+1)X$)**

for $i = 1, \ldots, n$, % for each stock $z_i$
compute $E(T^i_n)$;
run recBTT on stock $z_i$ with barrier $B$ and $k$ buckets;
for $j = 0, \ldots, k - 1$ % construct core superbuckets
$\beta_j^i \cdot \text{value} \leftarrow j\frac{B}{K}$;
$\beta_j^i \cdot \text{mass} \leftarrow \sum_{\ell=0}^{n} b_j(T^i[n, \ell]) \cdot \text{mass}$;
$\beta_k^i \cdot \text{mass} \leftarrow \sum_{\ell=0}^{n} b_j(T^i[n, \ell]) \cdot \text{mass}$ % construct overflow superbucket
$\beta_k^i \cdot \text{value} \leftarrow \frac{1}{\beta_k^i \cdot \text{mass}} \sum_{\ell=0}^{n} b_k(T^i[n, \ell]) \cdot \text{value} \times b_j(T^i[n, \ell]) \cdot \text{mass}$;
let $f_i(x) \leftarrow \sum_{j=0}^{k-1} \beta_j^i \cdot \text{mass} x^j$;
compute $f(x) \leftarrow \prod_{i=1}^{m} f_i(x)$ as described in Theorem 16;
let $f(x) \leftarrow b_0x^0 + b_1x^1 + \cdots + b_{m(k-1)}x^{m(k-1)}$; % for some $b_0, \ldots, b_{m(k-1)}$
return $\sum_{i=1}^{m} \beta_i^j \cdot \text{mass} \left( \beta_i^j \cdot \text{value} + \sum_{i' \neq i} E(T^i_n') - (n+1)X \right) + \sum_{j=k}^{m(k-1)} b_j \left( j\frac{B}{K} - (n+1)X \right)$;

**Theorem 16** Let $\sigma_{\min}$ be the minimum volatility among the stocks in the basket. The runtime of BasketBTT is

$$O\left( mT(\sigma_{\min} n, k) + mk \log m \log k + mk \log^2 m \right),$$

where $T(\sigma_{\min} n, k)$ is the runtime of recBTT on a binomial tree of size $n$ when $k$ buckets are used and the volatility of the underlying stock is $\sigma_{\min}$.

**Proof:** Computing the leaf bucket weights for each stock $z_i$ (using recBTT) takes $O(m \cdot T(\sigma_{\min} n, k))$ time. The runtime of the rest of the computation is dominated by the time to compute $\prod_{i=1}^{m} f_i(x)$. We now describe an efficient way to do this. Assume that $m = 2^r$, for some $r$. The general case is handled similarly. We conduct the multiplication by successively
multiplying consecutive polynomials together \( \log m = r \) times until we are left with a single polynomial. For \( 1 \leq i \leq m \), let \( f_i^0(x) = f_i(x) \). For \( 1 \leq j \leq r \) and \( 1 \leq i \leq \frac{m}{2^j} \), let \( f_i^j(x) = f_{2i-1}^{j-1}(x)f_{2i}^{j-1}(x) \). The answer that we are looking for is just \( \prod_{i=1}^{m} f_i(x) = f_1(x) \). At stage \( j \), we multiply together \( \frac{m}{2^j} \) pairs of polynomials, each of degree \( 2^{j-1}k \). Using FFT, this takes \( O\left( \frac{m}{2^j}2^{j-1}k \log(2^{j-1}k) \right) = O(mk \log(2^{j-1}k)) \) time. The total runtime to compute the product, over all stages, is \( O(mk(r \log k + \sum_{j=1}^{r} \log 2^{j-1})) = (mk(\log m \log k + \log^2 m)) \), from which the claimed result follows.

Our definition of the error made by BasketBTT is symmetric to the definition of error made by recBTT; ie, the maximum amount by which BasketBTT can underestimate \( \sum_{i=1}^{m} \sum_{t=0}^{n} S_t^i(\omega^i) \), for paths \( \omega^i \in T^i \).

**Theorem 17** Let \( \sigma_{\max} \) be the maximum volatility among the stocks in the basket. The error made by BasketBTT is at most \( mE(\sigma_{\max}, n, k) \), where \( E(\sigma_{\max}, n, k) \) is the error made by recBTT on a single stock with volatility \( \sigma_{\max} \).

**Proof:** For \( 1 \leq i \leq m \), let \( \omega_i \) be any path down the binomial tree corresponding to stock \( z_i \) and let \( T^i_n(\omega_i) \) be the total price down \( \omega_i \). When we run recBTT on \( z_i \), each \( T^i_n(\omega_i) \) is underestimated by at most \( E(\sigma_{\max}, n, k) \). Hence for any \( \omega_1, \ldots, \omega_m \), \( \sum_{i=1}^{m} T^i_n(\omega_i) \) is underestimated by at most \( mE(\sigma_{\max}, n, k) \), as claimed.

**Theorem 18** Given \( n, m, k, R > 2, \gamma = \frac{1}{\gamma}, \sigma_{\min} \) and \( \sigma_{\max} \), if we apply recBTT as described in Theorem 14 to construct the bucketed binomial tree for each stock, BasketBTT has an error of \( O\left( m \frac{B_n^{1/2+\gamma} \sigma_{\max}^{1-2\gamma}}{\gamma \sigma_{\min}} \right) \) and runs in time

\[
O(2^{1/\gamma}n^2mk(1/\gamma + \log \frac{k}{\sigma_{\min}} + mk \log m \log k + mk \log^2 m)).
\]

**Proof:** This follows directly from Theorems 16 and 17 and Theorem 14.

**Theorem 19** Given \( n, m, k, R > 2, \gamma = \frac{1}{\gamma}, \sigma_{\min} \) and \( \sigma_{\max} \), BasketBTT underestimates the price of a European Asian basket call by at most \( O\left( m \frac{B_n^{1/2+\gamma} \sigma_{\max}^{1-2\gamma}}{k \sigma_{\min}} \right) \) and runs in time

\[
O(2^{1/\gamma}n^2mk(1/\gamma + \log \frac{k}{\sigma_{\min}} + mk \log m \log k + mk \log^2 m)).
\]

### 6 Conclusion

#### 6.1 Concluding Remarks

This paper demonstrates the application of fast approximation algorithms in the pricing of exotic options on a binomial tree. First, we demonstrate that pricing Asian options and basket options are hard in finance, due to the complexity of their payoff functions. We then state several algorithms developed by Akcoglu, et al. that take advantage of both the binomial tree model and the payoff structure of Asian and basket options to price them accurately and quickly. Finally, we demonstrated from empirical comparisons that the BTT and recBTT algorithms are superior in terms of accuracy, running time, and precise analytical error bounds to exhaustive pricing and the Chalasani path-clustering algorithm. The recursive BTT algorithm, however, does not match the performance of the BTT algorithm, due to the relative complexity of the recBTT algorithm and the resulting overhead in its implementation. Although recBTT did not perform as well as expected in theory, the algorithm still prices options accurately within pre-determined error bounds in time that is tolerable in practice.
6.2 Further Research and Open Problems

Many open questions still remain in theory and practice of pricing Asian and basket options. The outstanding theoretical question is the complexity of pricing Asian options on a binomial tree. Although we have suggested that pricing Asian options is NP-hard, a proof is still unknown. Finding a related NP-hard problem would not only prove the complexity of pricing Asian options, but also possibly suggest other approximation methods.

Although we motivated reasons for extending BTT and recBTT to pricing basket options, we did not implement such variations here. The bucketed-traversal technique maybe easily extended to pricing basket options, and also to other option types, as well.

In this paper, we only compared BTT and recBTT with the exact CRR pricing algorithm and Chalasani’s path-clustering algorithms. To determine the true performance of both algorithms, we hope to compare them with other Asian option pricing approximations, such as the Hull-White and Levy-Turnbull-Wakeman geometric approximations, and other numerical algorithms, such as those suggested by [13]. The BTT and recBTT algorithms appear to be superior to the geometric approximations, because they assume that arithmetic-average Asian options behave like geometric-average Asian options, an erroneous assumption. We feel that bucketed-tree traversals algorithms are also superior to Monte Carlo and other numerical techniques, because they have analytical error bounds and are intuitively and simply related to the CRR binomial tree.

Although American options were not considered here, these algorithms can be easily applied to the American version of Asian and basket options, since the binomial tree is conducive to American pricing. In contrast, no analytical closed-form solution or approximation could handle the pricing of the American style of any option. Therefore, the algorithms could naturally be extended to the American style of the options considered here.

An interesting area of research would be to apply the algorithms developed here to real options, economic and legal situations that can be modeled as options. For example, companies that use Asian options often use them as a hedge against long-term average pricing contracts. Naturally, a method to price Asian options would be easily applicable in determining the value of such contracts. As better methods for pricing exotic options are developed, other real options can be priced by similar techniques.
References


