Accurate Approximations for European Asian Options *

PRASAD CHALASANI
Carnegie Mellon University
chal@cs.cmu.edu
http://www.cs.cmu.edu/∼chal

SOMESH JHA
Carnegie Mellon University
sjha@cs.cmu.edu
http://www.cs.cmu.edu/∼sjha

ASHOK VARIKOOTY
Global Quantitative Strategies
CS First Boston
New York
avarikoo@fir.fbc.com

January 5, 1998

Abstract

In the n-period binomial tree model, we provide fast algorithms to compute very accurate lower and upper bounds on the value of a European-style Asian option. These algorithms are inspired by the continuous-time analysis of Rogers and Shi [12]. Specifically we consider lower bounds that are given by grouping stock-price paths in the tree according to the value of a certain random variable Z, and treating all paths in a group as having the same arithmetic stock-price average, namely the average of the arithmetic average over the group. For a specific Z, Rogers and Shi show analytically that the error in the lower bound is small. However they are only able to estimate the lower and upper bounds numerically as integrals in continuous time. Indeed, for their choice of Z these bounds are difficult to compute exactly in the binomial tree model. We show how to choose Z so that the bounds can be computed *exactly* in time proportional to n^4 by forward induction. Moreover, our bounds are strictly better than theirs.

1 Introduction

The binomial tree model of Cox, Ross and Rubinstein [4] is commonly used as a discrete-time approximation to a continuous-time geometric Brownian motion. The pricing of simple calls and puts in this model can be done efficiently. However, *path-dependent* options such as Asian options are notoriously hard to price in this model. The Black-Scholes differential equation for the price of such options has no known closed form solution.

The payoff of an Asian option is based on the *arithmetic average* of the stock price. In particular, suppose the option expires at time T and let A_T denote the arithmetic average of the stock prices up to time T. Then the payoff of a European style Asian call with strike L is $(A_T - L)^+$ where x^+ denotes $\max\{x, 0\}$. Similarly the payoff of a European-style Asian put with strike L is $(L - A_T)^+$. Such options are of obvious appeal to a company which must buy a commodity at a fixed time each year, yet has to sell it regularly throughout the year [15]. These options allow investors to eliminate losses from movements in an underlying asset without the need for continuous rehedging. Asian

*We would like to thank Krishna Ramaswamy (Wharton) and Rui Kan (CS First Boston) for useful comments and discussions. Any errors are our own responsibility.

options are commonly used for currencies [15], interest-rates and commodities such as crude oil [6].

Several researchers have devised approximation algorithms for Asian options. Monte Carlo methods and Fourier transform techniques have been explored by Kemna and Vorst [8], and Carverhill and Clewlow [2] respectively. The second category of approaches consists of attempts to approximate the distribution of the arithmetic average (for which no simple specification is known) by a more tractable one. Examples of such work include those by Ruttiens [13], Levy [9], [10], Levy and Turnbull [11], and Turnbull and Wakeman [14]. The third approach, considered by Yor [16] and Geman and Yor [5], is to derive formulas for the Laplace transform of an Asian option. However, there have been no numerical studies of the inversion of this Laplace transform, and no simple analytical inversion has been found. Hull and White [7] consider interpolation methods for Asian options and other path-dependent options. Finally, Chalasani, Jha and Saias [3] have shown large-deviation results on the underlying additive random-walk to design approximations for these options. None of the above methods offers an approximation for the Asian option price that is both fast and very accurate.

The work in the present paper is most closely related to that of Rogers and Shi [12]. They provide a method for computing *lower bounds* on the price of an Asian option in the continuous-time geometric Brownian motion model. Their main idea is the following. We focus on the fixed-strike Asian call option throughout this paper but it should be clear how to extend our ideas to Asian puts, and to floating-strike versions of these options. For any random variable Z, using Jensen's inequality and the observation that $g(x) = x^+$ is a convex function, we can lower-bound the value of an Asian call with strike L (we ignore discount factors):

$$\begin{aligned}
\mathbf{E}\left[(A_T - L)^+\right] &= \mathbf{E}\left[\ \mathbf{E}[(A_T - L)^+|Z]\ \right] \\
&\geq \mathbf{E}\left[\ \mathbf{E}[A_T - L|Z]^+\ \right] \\
&= \mathbf{E}\left[\ [\mathbf{E}(A_T|Z) - L]^+\ \right] \qquad (1)
\end{aligned}$$

The lower-bound on the right in (1) can be used as an approximation for the option value. The quality of the approximation clearly depends on the choice of the random variable Z. In fact, Rogers and Shi give an upper bound on the error of this approximation,

$$\tfrac{1}{2}\mathbf{E}[\mathrm{var}(A_T|Z)^{\frac{1}{2}}], \qquad (2)$$

which can be combined with the above lower bound to get an upper bound on the option price. These bounds are integrals in continuous time, and Rogers and Shi *estimate* them numerically for various choices of Z. Rogers and Shi do not show how to compute these bounds efficiently in the binomial tree model. In fact, for the random variables Z that they consider, computing the bounds exactly appears to require time proportional to 2^n in the n-period binomial model. In this paper, we show how to choose Z in such a way that the lower and upper bounds can be computed *exactly* in time proportional to n^4. To see how much smaller n^4 is compared 2^n, let us take $n = 40$, as typically occurs when using the binomial model. For this n, our implementation of the algorithm (on a Sun Ultra Sparc Workstation) can compute the upper and lower bounds in about 5 seconds. By contrast the "brute force" algorithm that examines all 2^n paths would take about $2^{40}/40^4$ times longer, or at least 2×10^6 seconds, which is more than 22 days! Even for $n = 80$ our code runs in under a minute.

One way to think of the above approximation in the n-step Binomial tree model is the following. We let A_n denote the discrete-time analog A_T. The lower bound on the price of an Asian call is now

$$\mathbf{E}\left[\ [\mathbf{E}(A_n|Z) - L]^+\ \right], \tag{3}$$

and the error bound is

$$\mathbf{E}\left[\ \mathrm{var}(A_n|Z)^{\frac{1}{2}}\ \right]. \tag{4}$$

The difficulty in computing the exact[1] value $\mathbf{E}\left[(A_n - L)^+\right]$ lies in the fact that every stock price path has a different (arithmetic) average stock price A_n, and there are 2^n paths. Evaluating $\mathbf{E}\left[(A_n - L)^+\right]$ exactly therefore seems to require examining each path and checking whether or not $A_n > L$ on that path. On the other hand, to compute the lower bound (3), we do the following. For each possible value z_i of Z, we *first* compute $\mathbf{E}(A_n|Z = z_i)$, which is the average value of A_n over the group of paths with $Z = z_i$, and *then* check whether or not this average is greater than L. In other words, instead of checking whether or not $A_n > L$ for each path, we are grouping the paths according to the random variable Z, and checking whether or not the average of A_n over each group is greater than L. This is equivalent to saying that we are approximating the value of the Asian option by that of a new option where for each z_i, all paths with $Z = z_i$ are treated as having the same average stock price, equal to the *probability-weighted average* of A_n over these paths. Therefore, as can be seen from the error bound (4), if the variance of the arithmetic stock price average A_n over each group is small, then the lower bound will be accurate. If there are not too many possible values z_i of Z, and each $\mathbf{E}(A_n|Z = z_i)$ can be computed fast, then we will have an efficient approximation procedure.

Rogers and Shi find that a particularly accurate lower bound is obtained by taking

$$Z = \int_0^T B_u\, du, \tag{5}$$

which is the integral of the underlying Brownian motion from time 0 to T. In the n-step binomial tree model this corresponds to the *position-sum* of the underlying random walk, defined as follows. The steps of this random walk are given by the random variables $X_i = \pm 1$, $i = 1, 2, \ldots, n$. The **position** of this random walk at time k is given by $\mathbb{X}_0 = 0$, and

$$\mathbb{X}_k = \sum_{i=1}^{k} X_i, \quad k = 1, 2, \ldots, n. \tag{6}$$

The **position-sum** of the random walk at time k is

$$\mathcal{X}_k = \sum_{i=0}^{k} \mathbb{X}_i, \quad k = 0, 2, \ldots, n. \tag{7}$$

[1] Henceforth, for brevity, we will refer to $\mathbf{E}\left[(A_n - L)^+\right]$ as the "exact" option value, and our goal is to approximate this value. Note, however, that this "exact" value is itself an approximation to the option value in continuous time, and converges to that value as n tends to infinity.

Then the discrete-time analog of Z in (5) is $Z = \mathcal{X}_n$. Fig. 1 shows an example of how $\mathcal{X}_n$ is calculated. The problem with choosing $Z = \mathcal{X}_n$, however, is that one cannot compute the conditional expectation $\mathbf{E}(A_n|\mathcal{X}_n)$ in the lower bound (3) efficiently in the binomial model. This is because different tree paths with the same value of $\mathcal{X}_n$ may have different probabilities. For example, the top path in Fig. 1 has a probability different from that of the other three paths, even though they all have the same position-sum $\mathcal{X}_6 = 1$. There is no simple way to compute the expected value of A_n over these paths. However, if we take $Z = (\mathcal{X}_n, S_n)$ (as we show in Section 3) where S_n is the stock price at time n, then we get two useful properties. Firstly, all paths with the same value of Z have the same final stock price, and so have the same number of up-ticks, which implies they have the same probability. Secondly, for $k = 1, 2, \ldots, n$, the number of possible values of the position-sum $\mathcal{X}_k$ is proportional to k^3 (as opposed to 2^k possible values of A_k). Using these two facts we can compute the conditional expectations $\mathbf{E}(A_k|\mathcal{X}_k, S_k)$ by forward induction in time proportional to n^4. In addition, an improved version of the Rogers-Shi error bound (which we describe later) can also be computed in the same order of time along with the lower bound. A further advantage of our choice of Z is that the lower and upper bounds will be strictly better than with $Z = \mathcal{X}_n$, since the grouping of paths is now finer. In fact we show by means of numerical experiments (Section 4) that our choice of Z produces lower and upper bounds on the option price that are very close to each other.

Note that the *geometric* stock price average is a simple function of $\mathcal{X}_n$ (See Section 3 for the precise expression). This helps explain intuitively why conditioning on $(\mathcal{X}_n, S_n)$ yields a good approximation: paths with the same geometric stock-price average that end at the same stock price will likely have arithmetic stock-price averages that do not vary too much. Figure 3 illustrates this point. It is important to point out that simply replacing the arithmetic average of stock prices by the geometric average does not yield a good approximation to the option value, as Turnbull and Wakeman [14] have shown. They derive a closed form expression for the geometric-average-based option value and show that it can differ considerably from the usual Asian option value, particularly for long maturities and high volatilities. This can also be seen in Fig. 3. For $n = 30$, on all paths with $h = 28$ up-ticks and position-sum $\mathcal{X}_n = 403$, the geometric stock-price average is about 160, whereas the arithmetic stock-price average ranges from 166 to 169.

The paper is organized as follows. In Section 2 we describe the binomial model and some basic notation. Section 3 develops our algorithm. After showing some important properties of the position-sums $\mathcal{X}_k$ in subsection 3.1, we present the forward-induction algorithm in subsection 3.2. We analyze the time and space complexity of our algorithm in subsection 3.3. The computation of our improved error bound is described in subsection 3.4. We show numerical comparisons with other algorithms from the literature in Section 4. Section 5 contains the conclusion and a discussion of future work.

2 The binomial model and basic notation

In this paper we consider an Asian option written on an underlying risky asset, which we refer to as the "stock". Following Black and Scholes [1], we make the standard assumption that the price of the stock $S(t)$ at time t satisfies the stochastic differential equation for geometric Brownian motion:

$$dS(t) = \mu S(t)\, dt + \sigma S(t)\, dB(t),$$

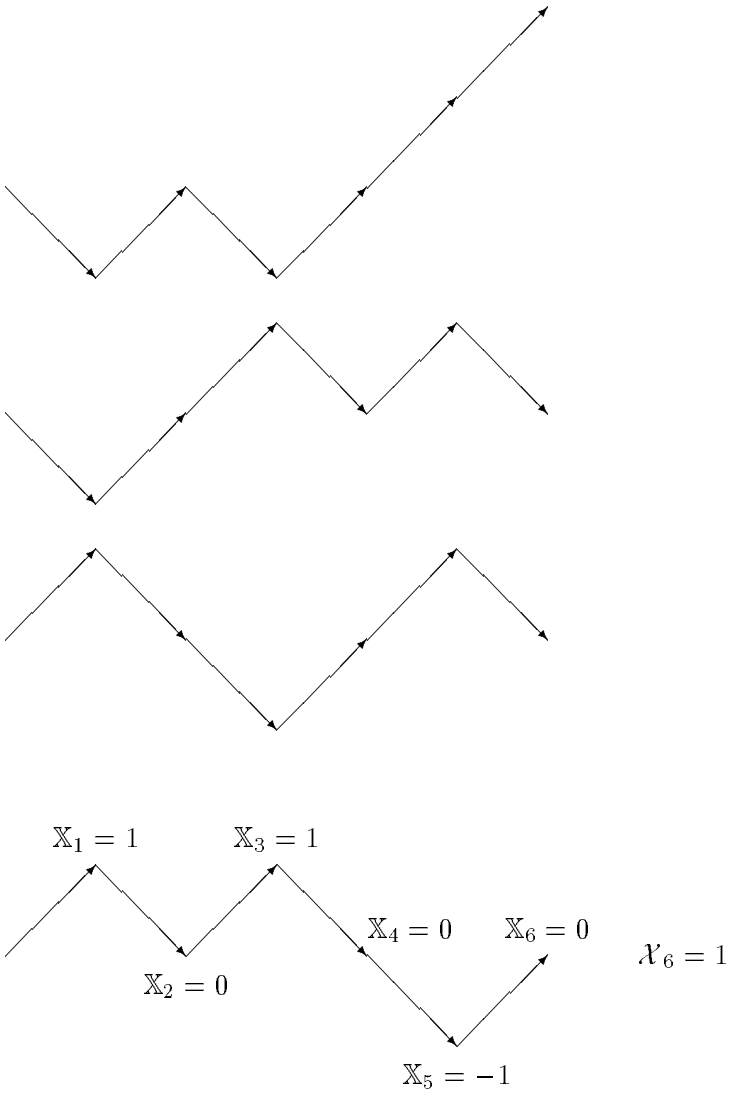

Figure 1: Showing some paths in the binomial tree that have position-sum $\mathcal{X}_6 = 1$. Note that the top path has $h = 4$ up-ticks and so reaches lattice node $(k, h) = (6, 4)$, whereas the other three paths reach lattice node $(k, h) = (6, 3)$ since they have 3 up-ticks. The values of the position $\mathbb{X}_i$, $i = 1, 2, \ldots, k$ are shown for one of the paths.

where the drift rate μ and volatility σ are constant, and $B(t)$ is a Brownian motion process. We will also assume that the risk-free interest rate is r, a constant. The derivative security expires at time T.

The continuous-time function $S(t)$ can be approximated in the form of a Cox-Ross-Rubinstein [4] binomial tree, as follows. The life of the option is divided into n time steps of length $\Delta t = T/n$. In time Δt the stock price moves up by a factor u with probability p, or down by a factor $d = 1/u$ with probability $q = 1 - p$, where

$$u = e^{\sigma\sqrt{\Delta t}},$$
$$p = \frac{e^{\mu\Delta t} - d}{u - d}.$$

Time k in this discrete model corresponds to continuous-time $k\Delta t$. Also we assume that a unit of currency lent or borrowed at time k will be worth $1 + r$ units at time $k + 1$. For brevity we write R to denote $1 + r$. This binomial tree model will be assumed throughout the paper.

More formally, we have a sample space Ω of all possible sequences of n independent coin-tosses (i.e. H's and T's), where an H corresponds to an up-tick of the stock and a T corresponds to a down-tick. A typical sequence (or "path") in Ω is written $\omega = \omega_1\omega_2 \ldots \omega_n$, where $\omega_i \in \{H, T\}$ represents the ith coin-toss. We will use the term "path" loosely to refer to any prefix of a path in Ω. The **length** of a path is the number of branches it contains. A k**-prefix** of a path $\omega \in \Omega$ is the length-k prefix of ω. For $k = 1, 2, \ldots, n$, we define the random variable X_k to be 1 if $\omega_k = 1$ and -1 otherwise. We can thus treat the process X_k as defining a symmetric random walk. Then we can define, for $k = 0, 1, \ldots, n$, the random variables $\mathbb{X}_k, \mathcal{X}_k$ as in the Introduction (see (7)).

We now introduce some additional notation that will be useful later. The random variable $H_k(\omega)$ is defined as the number of heads (or up-ticks) in the k-prefix of ω. We define $H_0(\omega) = 0$ for all ω. It is easy to see that the position $\mathbb{X}_k$ of the random walk at time k is the number of up-ticks minus the number of down-ticks up to time k, so

$$\mathbb{X}_k = H_k - (k - H_k) = 2H_k - k, \quad k \geq 1. \tag{8}$$

In this paper it will be useful to consider the **binomial lattice** whose **nodes** correspond to the possible positions of the underlying random walk at different times. Specifically, for $k = 0, 1, \ldots, n$ and $h = 0, 1, \ldots, k$, we say that a tree path ω **passes through**, or **reaches**, the node (k, h) if $H_k(\omega) = h$. From (8), all paths that pass through a node (k, h) have the same random-walk position $\mathbb{X}_k$ at time k, namely $2h - k$. We say that the value of a random variable (e.g., stock price S_k) at the node (k, h) is x if the value of the random variable at time k is x on a tree path that passes through (k, h). See Fig. 2 for an example of a lattice.

The stock price at time k is denoted S_k, $k = 0, 1, \ldots, n$, and is given by

$$S_k = S_0 u^{X_1 + X_2 + \ldots + X_k} = S_0 u^{\mathbb{X}_k} = S_0 u^{2H_k - k}.$$

Thus the stock price at point (k, h) is $S_0 u^{2h-k}$. The average stock price at time k is defined as

$$A_k = (S_0 + S_1 + \ldots + S_k)/(k + 1), \quad k \geq 0.$$

The payoff of an **Asian call** with strike L at time n is $(A_n - L)^+$ and the payoff of an **Asian put** is $(L - A_n)^+$. The payoff of a **floating-strike Asian call** at time n is $(A_n - S_n)^+$, and for a **floating-strike Asian put**, the payoff is $(S_n - A_n)^+$. In general we write W_n^+ to denote the payoff of any

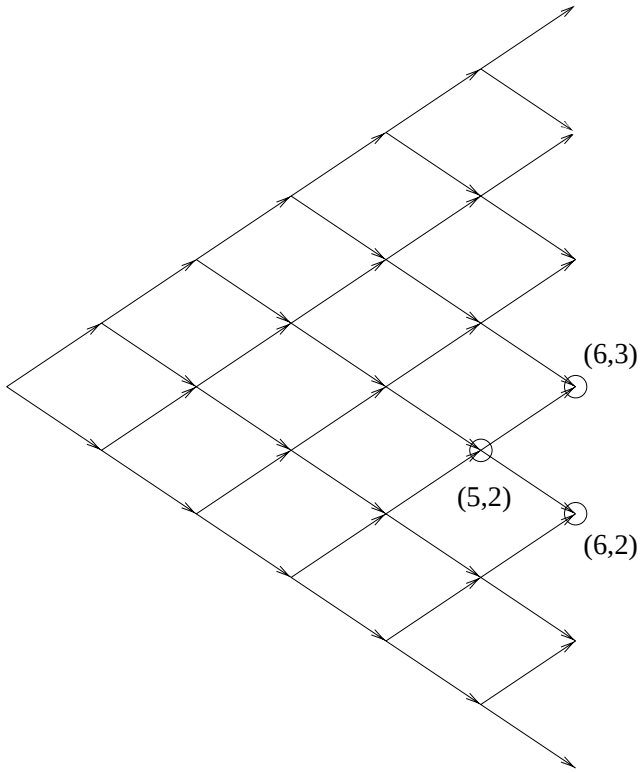

Figure 2: An example of a binomial lattice. The nodelets inside the circled nodes are shown in Fig. 4

of the above options at time n. The **value** of such an option (at time 0) is defined as

$$V = \mathbf{E}(W_n^+)/R^n.$$

In this paper we are concerned with estimating $\mathbf{E}(W_n^+)$, and we will ignore the discounting factor R^n.

Let $\mathcal{F}$ denote the σ-algebra consisting of all subsets of Ω, and $\mathcal{F}_k$ denote the σ-algebra generated by the first k coin-tosses, i.e., by $X_1, \ldots, X_k$. We will always assume the probability measure $\mathbf{P}$ on $(\Omega, \mathcal{F})$ given by:

$$\mathbf{P}[X_k = 1] = p, \quad k = 1, 2, \ldots, n.$$

3 The algorithm

We now show how to compute the lower bound (3), with $Z = (\mathcal{X}_n, S_n)$. Note that since the stock price S_n only depends on the number of up-ticks H_n by time n, this choice of Z is equivalent to using $Z = (\mathcal{X}_n, H_n)$. Thus the lower bound we want to compute is

$$\mathbf{E}\left[\ [\mathbf{E}(A_n|\mathcal{X}_n, H_n) - L]^+\ \right], \tag{9}$$

and the error bound is

$$\tfrac{1}{2}\mathbf{E}\left[\ \mathrm{var}(A_n|\mathcal{X}_n, H_n)^{\frac{1}{2}}\ \right], \tag{10}$$

Note, incidentally, that the *geometric average* of the stock prices from time 0 to time n is a simple function of $\mathcal{X}_n$:

$$(S_0 S_1 \ldots S_n)^{1/(n+1)} = S_0 \left(u^{\sum_{i=0}^n \mathbb{X}_i}\right)^{1/(n+1)} = S_0 \left(u^{\mathcal{X}_n}\right)^{1/(n+1)}.$$

In [12], Rogers and Shi consider an approximation based on conditioning on $\mathcal{X}_n$ alone rather than both $\mathcal{X}_n$ and H_n. As we show below, conditioning on both random variables has the advantage that it allows efficient computation in the binomial tree model. Moreover, the resulting approximation is strictly better than the one obtained by conditioning on $\mathcal{X}_n$ alone.

In order to compute the lower bound (9), we need to know, for each value h of H_n (i.e. at each terminal node (n, h) in the lattice), the set of possible values of $\mathcal{X}_n$. For $k = 0, 1, \ldots, n$ and $h = 0, 1, \ldots, k$, we let $N(k, h)$ denote the number of possible values of $\mathcal{X}_k$ at node (k, h), and write the elements of this set in increasing order as

$$\xi(k, h, 0), \xi(k, h, 1), \ldots, \xi(k, h, N(k, h) - 1). \tag{11}$$

Recall that a path is said to pass through a node (k, h) of the lattice if $H_k = h$ on this path. It will be useful to visualize each node (k, h) as being partitioned into $N(k, h)$ **nodelets**. For $k = 0, 1, \ldots, n$, $h = 0, 1, \ldots, k$, and $i = 0, 1, \ldots, N(k, h) - 1$, we say that a tree path ω **passes through**, or **reaches**, the nodelet (k, h, i) if $H_k(\omega) = h$ and $\mathcal{X}_k(\omega) = \xi(k, h, i)$. We let $M(k, h, i)$ be the number of paths in the tree that pass through the nodelet (k, h, i). Fig. 4 shows the nodelets contained inside the nodes $(5, 2), (6, 3)$ and $(6, 2)$ of Fig. 2. We show how to compute these quantities later, but for now let us assume that they are known.

n	H_n	$\mathcal{X}_n$	Paths	G_n	Min. A_n	$\mathbf{E}(A_n\|\mathcal{X}_n, H_n)$	Max. A_n	$\mathrm{var}(A_n\|\mathcal{X}_n, H_n)^{\frac{1}{2}}$
8	6	18	4	115.19	115.57	115.83	116.35	0.31
10	8	33	5	120.89	121.55	121.91	122.61	0.40
16	14	102	8	134.99	136.82	137.38	138.50	0.58
20	18	168	10	143.01	145.83	146.48	147.81	0.66
30	28	403	15	160.75	166.49	167.34	169.06	0.82
30	20	207	279260	127.61	128.05	128.56	130.64	0.33

Figure 3: Illustrating the small variance of the arithmetic stock-price average A_n over the set of paths that have the same geometric stock-price average G_n and up-ticks H_n. The parameters assumed are $S_0 = 100,\ L = 90,\ \sigma = .2$, and $\mu = .05$. "Paths" indicates the number of paths with the indicated value of H_n and $\mathcal{X}_n$. "Min. A_n," "Max. A_n," and $\mathbf{E}(A_n|\mathcal{X}_n, H_n)$ respectively indicate the smallest, largest and average value of A_n over this set of paths; all these paths have the same value of G_n. Finally, $\mathrm{var}(A_n|\mathcal{X}_n, H_n)$ is the variance of A_n over this set of paths. Note the significant difference between the arithmetic and geometric stock-price averages.

Using the nodelet terminology, it is now easy to see how to compute the expectation (9). The inner conditional expectation, for a specific combination of values of H_n and $\mathcal{X}_n$, is of the form $\mathbf{E}[A_n|H_n = h, \mathcal{X}_n = \xi(n,h,i)]$. For brevity we write

$$\widetilde{A}(k,h,i) = \mathbf{E}[A_k|H_k = h, \mathcal{X}_k = \xi(k,h,i)]. \tag{12}$$

Thus there is one such conditional expectation term for each nodelet (n,h,i) at level n in the lattice. For a given nodelet (n,h,i), $\widetilde{A}(n,h,i)$ is the expectation of A_n on a tree path given that the path passes through this nodelet. Since all paths through this nodelet have the same probability $p^h q^{n-h}$, this is simply the (unweighted) average of A_n over these paths. We will describe later how this can be computed by forward induction. Suppose these averages $\widetilde{A}(n,h,i)$ have been computed at the level n nodelets. Then the expression (9) is simply the weighted sum of all the $\widetilde{A}(n,h,i)$ values, where the weight of $\widetilde{A}(n,h,i)$ is $\mathbf{P}(H_n = h, \mathcal{X}_n = \xi(n,h,i))$, which is $p^h q^{n-h} M(n,h,i)$. In the expressions that follow we write, for brevity,

$$P(n,h,i) = \mathbf{P}(H_n = h, \mathcal{X}_n = \xi(n,h,i)).$$

Thus we can write the above lower bound as

$$\begin{aligned}
&\mathbf{E}\left[\ [\mathbf{E}(A_n|\mathcal{X}_n, H_n) - L]^+\ \right] \\
&= \sum_{h=0}^{n} \sum_{i=0}^{N(n,h)-1} P(n,h,i) \left[\widetilde{A}(n,h,i) - L\right]^+ \\
&= \sum_{h=0}^{n} \sum_{i=0}^{N(n,h)-1} p^h q^{n-h} M(n,h,i) \left[\widetilde{A}(n,h,i) - L\right]^+
\end{aligned} \tag{13}$$

3.1 Properties of the position-sum $\mathcal{X}_k$

For future reference we collect here some properties of the position-sums $\mathcal{X}_k$ that we will need later. First we show an important property of the possible values $\xi(k,h,i)$ of the position-sum $\mathcal{X}_k$ at node (k,h).

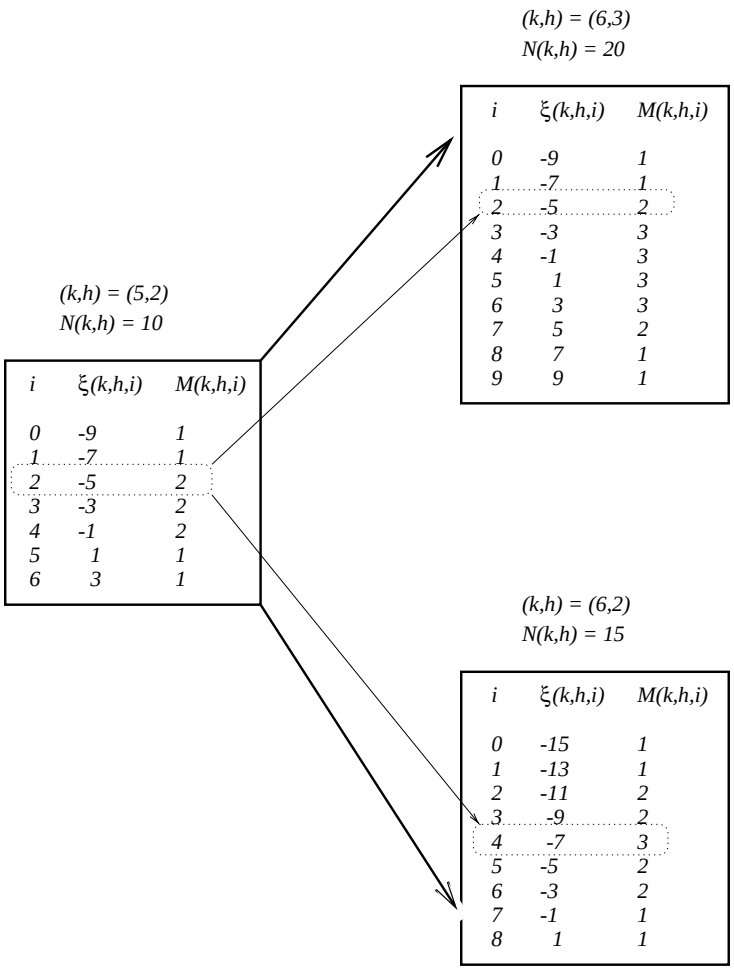

Figure 4: Nodelets inside the circled nodes $(5,2), (6,3), (6,2)$ of Fig. 2. Each box represents a node (k,h), and row i in a box represents nodelet (k,h,i). We have shown the number of paths $M(k,h,i)$ reaching each nodelet (k,h,i). An example of the forward induction update is shown: nodelet $(5,2,2)$ updates the values at nodelet $(6,3,2)$ and $(6,2,4)$.

Lemma 1 *The possible values of $\mathcal{X}_k$ at node (k,h) range in increments of 2 between the minimum and the maximum, i.e.,*

$$\xi(k,h,i+1) = \xi(k,h,i) + 2, \quad i = 0, 1, \ldots, N(k,h) - 2.$$

Proof: To show this, it will suffice to show two facts: (a) For a given k, the possible position-sums $\mathcal{X}_k$ are either all even or all odd, and (b) If $\xi(k,h,i)$ is a possible (but not maximum) value of $\mathcal{X}_k$ at (k,h), then there is a path through (k,h) with $\mathcal{X}_k = \xi(k,h,i) + 2$. To show (a), we write down the expression for $\mathcal{X}_k$, for $k > 0$:

$$\begin{aligned}
\mathcal{X}_k &= \sum_{i=1}^{k} \mathbb{X}_i \quad \text{(From definition (7))} \\
&= \sum_{i=1}^{k} (2H_i - i) \quad \text{(From (8))} \\
&= 2\sum_{i=1}^{k} H_i - \sum_{i=1}^{k} i.
\end{aligned}$$

The first term in the last sum takes different *even* values depending on the path, and the second term has a fixed value for a given k. Thus if the second term is even, then all possible values of $\mathcal{X}_k$ are even, otherwise all possible values will be odd. To show (b), consider a path passing through the node (k,h). Examine the sequence of values of X_i on this path from $i = 1$ to $i = k$. Since by assumption this path does not have the maximum possible value of $\mathcal{X}_k$, there must be some $j < k$ such that $X_j = -1$ and $X_{j+1} = 1$. Now modify this path so that all X_i remain the same except that $X_j = 1$ and $X_{j+1} = -1$. This new path also passes through (k,h). Clearly this change does not affect any of the positions $\mathbb{X}_i$, except that $\mathbb{X}_j$ is now 2 more than before. Thus $\mathcal{X}_k$ on this new path is 2 more than on the old path. This completes the proof. ■

We use the above property to show

Lemma 2 *(a) The i'th possible value of $\mathcal{X}_k$ at node (k,h) is given by*

$$\xi(k,h,i) = h^2 + h - (k^2 + k)/2 + 2i. \tag{14}$$

(b) The number of possible values of $\mathcal{X}_k$ on a path through (k,h) is

$$N(k,h) = 1 + kh - h^2. \tag{15}$$

(c) Consider a tree path that passes through nodelet (k,h,i) and let $v = 1$ if there is an uptick on the path, and $v = 0$ otherwise. Then the path passes through nodelet $(k+1, h+v, j)$ where

$$j = i + (1 - v)h. \tag{16}$$

Proof: First we establish the value of $\xi(k,h,0)$, which is the minimum possible value of $\mathcal{X}_k$ at the node (k,h). It is easy to see that this minimum occurs on a path consisting of $(k-h)$ down-ticks followed by h up-ticks, so

$$\begin{aligned}
\xi(k,h,0) &= -\sum_{i=1}^{k-h} i - \sum_{i=1}^{h} (k - h - i) \\
&= h^2 + h - (k^2 + k)/2
\end{aligned} \tag{17}$$

The statement (a) then follows by Lemma 1. Note that the maximum possible position-sum $\mathcal{X}_k$ at the node (k,h) occurs on a path with h up-ticks followed by $k-h$ down-ticks, so this maximum equals $-\xi(k,k-h,0)$. Therefore the number of possible values of $\mathcal{X}_k$ at node (k,h) is, using Lemma 1,

$$N(k,h) = 1 \ + \ \tfrac{1}{2}\Big[-\xi(k,k-h,0) - \xi(k,h,0)\Big],$$

which works out to the expression required in (b).

Consider a path passing through the nodelet (k,h,i). If there is an up-tick at time $k+1$ on this path, the nodelet reached at the next level is of the form $(k+1,h+1,j)$. In the case of a down-tick, the nodelet reached at level $k+1$ is of the form $(k+1,h,j')$. In either case, the nodelet reached at level $k+1$ is of the form $(k+1,h+v,j)$, where v is as defined in the Lemma statement. We need to determine the index j. Note that the position-sum $\mathcal{X}_k$ on all paths passing through the nodelet (k,h,i) is $\xi(k,h,i)$, and the position-sum $\mathcal{X}_{k+1}$ is, from (8),

$$\xi(k,h,i) + \mathbb{X}_{k+1} = \xi(k,h,i) + 2(h+v) - (k+1),$$

and the index j must be chosen such that this equals $\xi(k+1,h+v,j)$. Therefore by Lemma 1, j must equal

$$\frac{1}{2}\Big(\xi(k,h,i) + 2(h+v) - (k+1) - \xi(k+1,h+v,0)\Big),$$

which simplifies to $i + (1-v)h$. ■

3.2 The forward induction

We now show how to compute the quantities $M(n,h,i)$ and $\widetilde{A}(n,h,i)$ in the lower bound (13). To see how to compute $\widetilde{A}(n,h,i)$ we introduce the random variable $\mathbb{S}_k$ defined as the sum of the stock prices from time 0 to k:

$$\mathbb{S}_k = \sum_{i=0}^{k} S_i, \quad k = 0,1,\ldots,n. \tag{18}$$

Note that $A_k = \mathbb{S}_k/(k+1)$ for $k = 0,1,\ldots,n$. We also define $\mathcal{S}(k,h,i)$ as the sum of $\mathbb{S}_k$ over all paths passing through the nodelet (k,h,i). Since $\widetilde{A}(n,h,i)$ is simply the unweighted average of A_n over all paths reaching the nodelet (n,h,i), it follows that

$$\widetilde{A}(n,h,i) = \frac{\mathcal{S}(n,h,i)}{(n+1)M(n,h,i)}. \tag{19}$$

Our algorithm computes the quantities $M(k,h,i)$ and $\mathcal{S}(k,h,i)$ by forward induction on k as follows. First, for $k = 1,2,\ldots,n$, $h = 0,1,\ldots,k$ and $i = 0,1,\ldots,N(k,h)-1$, we initialize $M(k,h,i)$ and $\mathcal{S}(k,h,i)$ to 0. To start the induction we set, in accordance with their definitions, $M(0,0,0) = 1$ and $\mathcal{S}(0,0,0) = S_0$. Let us assume inductively that we have computed all the $M(m,h,i)$ and $\mathcal{S}(m,h,i)$ for $m = 0,1,\ldots,k$, and we want to compute these quantities at level $k+1$. To do this we consider each nodelet (k,h,i) at level k and increment M and $\mathcal{S}$ at the appropriate nodelets at level $k+1$ by an amount equal to the contribution made by paths reaching nodelet (k,h,i).

Specifically, we proceed as follows. Suppose we are considering the contribution of nodelet (k,h,i) at level k to nodelets at level $k+1$. For $v=1$ (up-tick at time $k+1$) and $v=0$ (down-tick at time $k+1$) we do the following. Recall that $M(k,h,i)$ is the number of paths passing through the nodelet (k,h,i), and $\mathcal{S}(k,h,i)$ is the sum of $\mathbb{S}_k$ over these paths. From Lemma 2, we know that all these paths must at time $k+1$ pass through the nodelet $(k+1,h+v,j)$ where j is given by (16). Thus the paths reaching nodelet $(k+1,h+v,j)$ via nodelet (k,h,i) contribute an amount $M(k,h,i)$ to $M(k+1,h+v,j)$, so we increment the latter by this amount. Also, the paths reaching $(k+1,h+v,j)$ via (k,h,i) contribute

$$\mathcal{S}(k,h,i) + M(k,h,i)S_k = \mathcal{S}(k,h,i) + M(k,h,i)S_0 u^{2(h+v)-(k+1)}$$

to $\mathcal{S}(k+1,h+v,j)$, so we increment the latter by this amount.

Example. We illustrate the computations involved in the forward induction by means of the nodelets shown in Fig. 4. Suppose we have computed all the $M(k,h,i)$ and $\mathcal{S}(k,h,i)$ at level $k=5$, and we are considering the contribution of nodelet $(k,h,i)=(5,2,2)$ to the nodelets at level 6. To determine which nodelet $(6,3,j)$ is reached from $(5,2,2)$ by means of an up-tick, we use Lemma 2 (expression (16)) with $v=1$, and obtain

$$j = i + (1-v)h = i = 2.$$

We therefore increment $M(6,3,2)$ by the amount $M(5,2,2)$. Similarly, to determine which nodelet $(6,2,j)$ is reached from $(5,2,2)$ by means of an down-tick, we use expression (16) with $v=0$, and obtain

$$j = i + (1-v)h = i + h = 4.$$

We therefore increment $M(6,2,4)$ by the amount $M(5,2,2)$. Note that some other nodelet in node $(5,2)$ will contribute 1 to the M value at nodelet $(6,2,4)$, so eventually $M(6,2,4)$ will reach the correct value 3 as shown in the figure. The updates of the $\mathcal{S}$ values are done in a similar manner.
■

To summarize, the forward induction algorithm in pseudocode is given below. We use the standard notation $x+=y$ to stand for the assignment $x := x+y$. Note that we are using the expressions (15) and (16) from Lemma 2.

Initialization:
$M(0,0,0) = 1;\ \mathcal{S}(0,0,0) = S_0;$
for $k := 1$ **to** n **do**
  for $h := 0$ **to** k **do**
    for $i := 0$ **to** $kh - h^2$ **do**
      $M(k,h,i) := 0;$
      $\mathcal{S}(k,h,i) := 0;$
    end
  end
end
Forward Induction:
for $k := 0$ **to** $n-1$ **do**
  for $h := 0$ **to** k **do**
    for $i := 0$ **to** $kh - h^2$ **do**
      for $v := 0$ **to** 1 **do**
        $j := i + (1-v)h;$
        $M(k+1, h+v, j) += M(k,h,i);$
        $\mathcal{S}(k+1, h+v, j) += \mathcal{S}(k,h,i) + M(k,h,i) S_0 u^{2(h+v)-(k+1)};$
      end
    end
  end
end

3.3 Time and space complexity

We now analyze the time and space required by our algorithm. The quantities $M(k,h,i)$ and $\mathcal{S}(k,h,i)$ are stored in a 3-dimensional array, where the index k ranges from 0 to n, the index h ranges from 0 to k, and the index i ranges from 0 to $N(k,h)$. Any element of these arrays can be accessed in essentially a constant amount of time. It is clear that the innermost initialization loop above is executed once for each nodelet (k,h,i) in the lattice, whereas the innermost loop in the forward induction is executed twice for each nodelet up to level $n-1$ in the lattice. For a given nodelet, the body of the induction loop consists only of simple additions, multiplication, exponentiation and array-access operations. For our asymptotic time analysis we will consider these operations as taking a "constant" amount of time. Thus the total time of the algorithm is proportional to the number of nodelets in the lattice from level 0 to level n.

The number of nodelets at a given node (k,h) is $N(k,h)$, since this is the number of possible values of $\mathcal{X}_k$ at node (k,h). From the expression (15), the total number of nodelets at level k in the lattice is

$$\begin{aligned}\sum_{h=0}^{k} N(k,h) &= \sum_{h=0}^{k} (1 + kh - h^2) \\ &= 1 + k.k(k+1)/2 - k(k+1)(2k+1)/6 \\ &= 1 + (k^3 - k)/6,\end{aligned}$$

and the total number of nodelets from level 0 to level n is

$$\begin{aligned}\sum_{k=0}^{n}\Big[1+(k^3-k)/6\Big] &= (n+1)+\frac{1}{6}\sum_{k=1}^{n}k^3-\frac{1}{6}\sum_{k=1}^{n}k\\ &= n+n^2(n+1)^2/24-n(n+1)/12\\ &= (n^4+2n^3-n^2)/24+11n/12+1.\end{aligned}$$

Thus the total time required by the algorithm is proportional to n^4. Since there is a constant amount of storage corresponding to each nodelet, the total space requirement of the algorithm is also proportional to n^4.

3.4 Error bound

We mentioned in the introduction that Rogers and Shi bound the error in the lower bound (9) by the expression (10). We now show a modified bound that is significantly better. For brevity let us denote $A_n - L$ by W_n. Rogers and Shi's error bound is based on the observation that, for any random variable Z,

$$\mathbf{E}(W_n^+|Z) - \mathbf{E}(W_n|Z)^+ \le \tfrac{1}{2}\operatorname{var}(W_n|Z)^{\frac{1}{2}}. \tag{20}$$

(The error in the approximation (3) is then upper bounded by the expectation of this quantity.) In other words, for a fixed value z_i of Z, the difference between $\mathbf{E}(W_n^+|Z=z_i)$ and $\mathbf{E}(W_n|Z=z_i)^+$ is at most $\frac{1}{2}\operatorname{var}(W_n|Z=z_i)^{\frac{1}{2}}$. However for some values z_i of Z, this difference is 0. To make this precise, we introduce the function

$$W_n^{min}(z) = \min_{\omega\in\Omega}\{W_n(\omega) : Z(\omega) = z\},$$

which is the minimum possible value of W_n over paths with $Z = z$. Similarly,

$$W_n^{max}(z) = \max_{\omega\in\Omega}\{W_n(\omega) : Z(\omega) = z\}.$$

Now for a given value z_i of Z_i, if $W_n^{max}(z_i) \le 0$, then for all paths ω with $Z(\omega) = z_i$, we have $W_n^+(\omega) = 0$, which implies $\mathbf{E}(W_n^+|Z=z_i) = 0$, and also $\mathbf{E}(W_n|Z=z_i) \le 0$, which implies $\mathbf{E}(W_n|Z=z_i)^+ = 0$. Therefore,

$$\mathbf{E}(W_n^+|Z=z_i) - \mathbf{E}(W_n|Z=z_i)^+ = 0 - 0 = 0.$$

On the other hand, if $W_n^{min}(z_i) \ge 0$, then for all paths ω with $Z(\omega) = z_i$, we have $W_n^+(\omega) = W_n(\omega)$, which implies $\mathbf{E}(W_n^+|Z=z_i) = \mathbf{E}(W_n|Z=z_i)$, and also $\mathbf{E}(W_n|Z=z_i) \ge 0$, which implies $\mathbf{E}(W_n|Z=z_i)^+ = \mathbf{E}(W_n|Z=z_i)$. Thus again we have

$$\mathbf{E}(W_n^+|Z=z_i) - \mathbf{E}(W_n|Z=z_i)^+ = 0.$$

Therefore we can refine the bound in (20) by writing

$$\mathbf{E}(W_n^+|Z) - \mathbf{E}(W_n|Z)^+ \le \tfrac{1}{2}\operatorname{var}(W_n|Z)^{\frac{1}{2}}\mathbf{1}_{\{W_n^{min}(Z)<0,W_n^{max}(Z)>0\}}, \tag{21}$$

where for any set of paths A, $\mathbf{1}_A$ denotes a random variable whose value is 1 on paths in A and zero elsewhere. (The notation $\{W_n^{min}(Z) < 0, W_n^{max}(Z) > 0\}$ denotes the set of paths ω for which $W_n^{min}(Z(\omega)) < 0$ and $W_n^{max}(Z(\omega)) > 0$.

We are now ready to compute an upper-bound on the error of the lower bound (9). Let $A^{min}(k,h,i)$ denote the minimum value of A_k over paths that pass through the nodelet (k,h,i). Let $A^{max}(k,h,i)$ be the maximum value of A_k over paths reaching (k,h,i). The error of the lower bound (9) is then

$$\begin{aligned}\mathbf{E}\left[\mathbf{E}[\,(A_n - L)^+|H_n, \mathcal{X}_n\,]\right] - \mathbf{E}\left[\mathbf{E}[(A_n - L)|H_n, \mathcal{X}_n\,]^+\right] \\ = \mathbf{E}\left[\mathbf{E}[\,(A_n - L)^+|H_n, \mathcal{X}_n\,] - \mathbf{E}[\,(A_n - L)|H_n, \mathcal{X}_n\,]^+\right]. \quad (22)\end{aligned}$$

Now using (21), noting that $\mathrm{var}(A_n - L) = \mathrm{var}\ A_n = \mathbf{E}A_n^2 - (\mathbf{E}A_n)^2$, and denoting

$$\widetilde{A^2}(n,h,i) = \mathbf{E}\left[A_n^2 | H_n = h, \mathcal{X}_n = \xi(n,h,i)\right],$$

this equals

$$\begin{aligned}\sum_{h=0}^{n} \sum_{\substack{0\le i\le N(n,h)-1 \\ A^{min}(n,h,i)<L, A^{max}(n,h,i)>L}} P(n,h,i)\left(\widetilde{A^2}(n,h,i) - \widetilde{A}(n,h,i)^2\right)^{\frac{1}{2}} \\ = \sum_{h=0}^{n} \sum_{\substack{0\le i\le N(n,h)-1 \\ A^{min}(n,h,i)<L, A^{max}(n,h,i)>L}} M(n,h,i)p^h q^{n-h}\left(\widetilde{A^2}(n,h,i) - \widetilde{A}(n,h,i)^2\right)^{\frac{1}{2}} \quad (23)\end{aligned}$$

We already know how to compute $\widetilde{A}(n,h,i)$ (see (19)). It only remains to show how to compute the values $A^{min}(n,h,i)$, $A^{max}(n,h,i)$ and the conditional expectations $\widetilde{A^2}(n,h,i)$. To compute the A^{min} and A^{max} values, we define $\mathcal{S}^{max}(k,h,i)$ as the maximum value of $\mathbb{S}_k$ over the paths that reach (k,h,i), and define $\mathcal{S}^{min}(k,h,i)$ as the minimum value of $\mathbb{S}_k$ over these paths. Clearly then

$$A^{min}(k,h,i) = \mathcal{S}^{min}(k,h,i)/(k+1),$$

and an analogous relation holds for A^{max}. To compute $\widetilde{A^2}(n,h,i)$ we define the random variable

$$\mathbb{T}_k = \sum_{i=0}^{k} S_i^2,$$

and

$$\mathbb{R}_k = \sum_{0\le i<j\le k} S_i S_j,$$

the sum of pairwise products of distinct stock prices from time 0 to time k. Now we can write

$$\begin{aligned}(n+1)^2 A_n^2 &= \left(\sum_{i=0}^{n} S_i\right)^2 \\ &= \mathbb{S}_n^2 \quad \text{(From the definition (18))} \\ &= \sum_{i=0}^{n} S_i^2 \ + \ 2 \sum_{0\le i<j\le n} S_i S_j \\ &= \mathbb{T}_n + 2\mathbb{R}_n. \quad (24)\end{aligned}$$

Note that since all paths reaching (n,h,i) have the same probability, $\widetilde{A^2}(n,h,i)$ is the unweighted average of $A_n^2 = (\mathbb{T}_n + 2\mathbb{R}_n)/(n+1)^2$ over these paths. Let us denote by $\mathcal{T}(k,h,i)$ the sum of $\mathbb{T}_k$ over all paths reaching (k,h,i), and let $\mathcal{R}(k,h,i)$ be the sum of $\mathbb{R}_k$ over all paths reaching (k,h,i). Then we can write

$$\widetilde{A^2}(n,h,i) = \frac{\mathcal{T}(n,h,i) + 2\mathcal{R}(n,h,i)}{(n+1)^2 M(n,h,i)}. \tag{25}$$

Thus it suffices to show how to compute the quantities $\mathcal{S}^{min}(k,h,i)$, $\mathcal{S}^{max}(k,h,i)$, $\mathcal{T}(k,h,i)$ and $\mathcal{R}(k,h,i)$ by forward induction. This is done by simply adding some steps to the previously described algorithm for computing $M(k,h,i)$ and $\mathcal{S}(k,h,i)$. We add the initialization steps

$$\mathcal{T}(0,0,0) := S_0^2; \quad \mathcal{R}(0,0,0) := 0; \quad \mathcal{S}^{max}(0,0,0) := \mathcal{S}^{min}(0,0,0) := S_0.$$

and in the body of the innermost initialization loop we add

$$\mathcal{T}(k,h,i) := 0; \quad \mathcal{R}(k,h,i) := 0;$$

and

$$\mathcal{S}^{min}(k,h,i) := \infty; \; \mathcal{S}^{max}(k,h,i) := 0;$$

where ∞ denotes some suitably value (such as nS_0u^n) that is larger than the maximum possible value of $\mathbb{S}_k$.

The updates in the forward induction part of the algorithm are done using a reasoning similar to that used in updating the $\mathcal{S}(k,h,i)$ values. It is easy to see that the updates required are

$$\mathcal{S}^{min}(k+1,h+v,j) := \min\Big\{\mathcal{S}^{min}(k+1,h+v,j), \mathcal{S}^{min}(k,h,i) + S_0u^{2(h+v)-(k+1)}\Big\};$$
$$\mathcal{S}^{max}(k+1,h+v,j) := \max\Big\{\mathcal{S}^{max}(k+1,h+v,j), \mathcal{S}^{max}(k,h,i) + S_0u^{2(h+v)-(k+1)}\Big\};$$
$$\mathcal{T}(k+1,h+v,j) \mathrel{+}= \mathcal{T}(k,h,i) + M(k,h,i)S_0^2u^{4(h+v)-2(k+1)};$$
$$\mathcal{R}(k+1,h+v,j) \mathrel{+}= \mathcal{R}(k,h,i) + \mathcal{S}(k,h,i)S_0u^{2(h+v)-(k+1)};$$

Thus both the lower bound and the error bound can be computed in time proportional to n^4.

4 Numerical experiments

We show three sets of numerical experiments. First, in Fig. 5, we run a series of experiments analogous to those in Rogers and Shi [12]. For these experiments, we used $n = 30$ steps in the binomial tree, and $S_0 = 100$. We varied the strike L, volatility σ and drift μ exactly as in [12]. For our lower bound we use (9), and for the upper bound we add our refined error bound (22) to this lower bound. The yardstock for our comparison with the Rogers-Shi bounds is their very accurate PDE estimate of the option price (They mention that this estimate agrees with the Kemna and Vorst [8] Monte Carlo results.) The PDE estimate, while very accurate, is very slow, often taking on the order of 100 seconds on a SUN SPARC Classic workstation. It will be noticed from the table that (a) the gap between our lower and upper bounds is much smaller than the gap between their bounds,

and (b) the average of our lower and upper bounds is in most cases closer to the PDE estimate than the average of their bounds. The difference between our bounds and those of Rogers-Shi can be explained by the fact that their bounds are continuous-time integrals that are *estimated* numerically. By contrast, our bounds are based on a discrete-time binomial-tree (which only approximates the continuous-time Brownian-driven process), but we compute our bounds *exactly* in discrete time. For $n = 30$ our algorithm computes the lower and upper bounds in about 5 seconds on a Sun UltraSparc workstation.

In Fig. 6 we again run a series of experiments analogous to those in Rogers-Shi. This time we include Monte Carlo option-value estimates from Levy and Turnbull [11]. Our bounds are considerably closer to the Monte Carlo results than the bounds of Rogers-Shi.

Finally, in Fig. 7, we compare our bounds with the approximations of Hull-White [7] (HW) and Levy-Turnbull-Wakeman (LTW) [11, 14]. The HW approximation computes, in the binomial tree model (with $n = 40$) the option value by backward recursion for only certain designated values of the arithmetic average A_k at each level in the lattice, using linear interpolation to estimate the option price at the other A_k values. The LTW method uses an analytic approximation. We also show the Monte Carlo estimates along with the standard error in parentheses. As can be seen, the *average* of our lower and upper bounds is much closer to the Monte Carlo estimate than the LTW approximation. Our approximation is also at least as good as the Hull-White approximation. The HW method overestimates the option value, whereas our method has the advantage of guaranteeing that the option value (in the binomial model) lies between our lower and upper bounds.

5 Conclusion and Future Work

In this paper we introduced an elegant and fast forward induction algorithm for computing very accurate lower and upper bounds on the value of a European Asian option in the n-step binomial model. The payoff of an Asian option depends on the arithmetic average of the stock prices, A_n. The difficulty in computing the option value exactly in the binomial model is that each path has a different value of A_n, and there are 2^n paths. Our lower bounding method is based on grouping paths into groups, where two paths belong to the same group if and only if they have the same value of (a) the final stock price S_n and (b) the *geometric* stock price average G_n (or equivalently, the position-sum of the underlying additive random walk). We treat all paths in a group as having the same arithmetic stock-price average, namely the *average* of A_n over the paths in the group. This is equivalent to computing the expectation of the conditional expectation given S_n and G_n.

Rogers and Shi consider a similar lower bound in continuous time, conditioning only on the integral of the underlying Brownian (which is analogous to the position-sum in discrete time). However, they only estimate their integrals by numerical methods and do not consider whether the corresponding approximation can be computed fast in the binomial model. We have argued in this paper that with their grouping of paths it is difficult to compute the lower bound exactly in the binomial model. We overcome this difficulty by grouping paths according to both G_n and S_n. This ensures that all paths in a group have the same probability. We use this fact, combined with the fact that the number of possible geometric averages G_k is only proportional to k^3, to devise a simple forward induction algorithm for both the lower bound and upper bound.

We have compared our algorithm experimentally with others that have appeared in the literature.

Strike L	Vol σ	Drift μ	RS-PDE	LB	UB	RS-LB	RS-UB
95	0.05	0.05	7.157	7.178	7.178	7.178	7.183
100			2.621	2.708	2.708	2.716	2.722
105			0.439	0.309	0.309	0.337	0.343
95		0.09	8.823	8.811	8.811	8.809	8.821
100			4.185	4.301	4.301	4.308	4.318
105			1.011	0.892	0.892	0.958	0.968
95		0.15	11.090	11.100	11.100	11.094	11.114
100			6.777	6.798	6.798	6.794	6.810
105			2.639	2.667	2.667	2.744	2.761
90	0.10	0.05	11.942	11.949	11.949	11.951	11.973
100			3.624	3.632	3.632	3.641	3.663
110			0.359	0.306	0.306	0.331	0.353
90		0.09	13.382	13.386	13.386	13.385	13.410
100			4.887	4.902	4.902	4.915	4.942
110			0.659	0.582	0.583	0.630	0.657
90		0.15	15.398	15.404	15.404	15.399	15.445
100			7.000	7.015	7.015	7.028	7.066
110			1.430	1.316	1.317	1.413	1.451
90	0.30	0.05	13.951	13.929	13.937	13.952	14.161
100			7.944	7.924	7.931	7.944	8.153
110			4.074	4.040	4.049	4.070	4.279
90		0.09	14.981	14.964	14.971	14.983	15.194
100			8.827	8.807	8.814	8.827	9.039
110			4.698	4.661	4.669	4.695	4.906
90		0.15	16.510	16.499	16.504	16.512	16.732
100			10.208	10.187	10.193	10.208	10.429
110			5.731	5.685	5.694	5.728	5.948

Figure 5: Comparison of our binomial-tree lower and upper bounds with the continuous-time bounds of Rogers-Shi [12]. The columns LB, UB indicate our lower and upper bounds respectively, while columns RS-LB and RS-UB indicate the bounds of Rogers-Shi. The column RS-PDE shows the Rogers-Shi PDE-based estimate of the option price.

Strike L	Vol σ	Drift μ	Monte Carlo	LB	UB	RS-LB	RS-UB
95	0.10	0.09	8.91	8.91	8.91	8.91	8.95
100			4.91	4.90	4.90	4.92	5.10
105			2.06	2.03	2.03	2.07	2.34
90	0.30		14.96	14.96	14.97	14.98	15.23
100			8.81	8.81	8.81	8.83	9.39
110			4.68	4.66	4.67	4.70	5.37
90	0.50		18.14	18.15	18.18	18.18	18.52
100			12.98	12.99	13.02	13.02	13.69
110			9.10	9.08	9.11	9.18	9.97

Figure 6: Comparison with Monte Carlo (MC) results from Levy-Turnbull [11] and with the Rogers-Shi [12] bounds. As in the previous figure, LB and UB indicate our lower and upper bounds respectively, and RS-LB and RS-UB indicate the bounds of Rogers-Shi.

Strike L	HW	LTW	MC (std. err.)	LB	UB	Avg
40	11.545	11.573	11.544 (0.006)	11.543	11.546	11.544
45	7.616	7.652	7.606 (0.008)	7.613	7.617	7.615
50	4.522	4.542	4.515 (0.010)	4.519	4.524	4.521
55	2.420	2.415	2.401 (0.009)	2.416	2.422	2.419
60	1.176	1.159	1.185 (0.007)	1.173	1.180	1.176

Figure 7: Comparison with Hull-White (HW) and Levy-Turnbull-Wakeman (LTW) results, with $n = 40$, $\sigma = 0.3$, $\mu = 0.1$, $S_0 = 50$. Our bounds are indicated by LB and UB. The column "Avg" contains the average of LB and UB.

In most cases the gap between our lower and upper bounds is much smaller, even for $n = 40$. Moreover, our results are much closer to the Monte Carlo estimates. Our algorithm is very fast as well: For $n = 40$, the upper and lower bounds can be computed in about 20 seconds on a Sun Ultra SPARC workstation.

Among the algorithms for Asian options that have previously appeared in the literature, the interpolation method of Hull and White appears to be the most accurate one. We have developed a hybrid algorithm that combines our approach with interpolation to obtain extremely accurate lower and upper bounds not just for European Asian options but also for American Asians. We will describe this work in a future paper. We also plan to estimate the error of the algorithm analytically.

References

[1] F. Black and M. Scholes. The pricing of options and corporate liabilities. *J. Political Economy*, 81:637–654, 1973.

[2] A. Carverhill and L. Clewlow. Flexible convolution. *RISK*, pages 25–29, April 1990.

[3] P. Chalasani, S. Jha, and I. Saias. Approximate option pricing. In *Proc. IEEE Symp. Foundations of Computer Science*, 1994. To appear in Algorithmica.

[4] J. Cox, S. Ross, and M. Rubinstein. Option pricing: A simplified approach. *J. Financial Economics*, 7:229–264, 1979.

[5] H. Geman and M. Yor. Bessel processes, asian options and perpetuities. In *FORC Conference, Warwick, UK*, 1992.

[6] B. Heenk, A. Kemna, and A. Vorst. Asian options on oil spreads. *Review of Futures Markets*, 9(3):510–528, 1990.

[7] J. Hull and A. White. Efficient procedures for valuing european and american path-dependent options. *Journal of Derivatives*, 1:21–31, 1993.

[8] A. Kemna and A. Vorst. A pricing method for options based upon average asset values. *J. Banking and Finance*, pages 113–129, March 1990.

[9] E. Levy. Asian arithmetic. *RISK*, pages 7–8, May 1990.

[10] E. Levy. Pricing european average rate currency options. *J. Internat. Money Finan.*, 11:474–491, 1992.

[11] E. Levy and S. Turnbull. Average intelligence. *Risk Magazine*, Feburary 1992.

[12] L. Rogers and Z. Shi. The value of an asian option. *J. Appl. Prob.*, 32:1077–1088, 1995.

[13] A. Ruttiens. Classical replica. *RISK*, pages 5–9, Feb 1992.

[14] S. Turnbull and L. Wakeman. A quick algorithm for pricing european average options. *J. Financial and Quantitative Analysis*, 26(3):377–390, 1991.

[15] P. Wilmott, J. DeWynne, and S. Howison. *Option pricing: mathematical models and computation*. Oxford Financial Press, 1993.

[16] M. Yor. On some exponential functionals of brownian motion. *Adv. Appl. Prob.*, 1992.