Efficient, Exact Algorithms for Asian Options with Multiresolution Lattices

Tian-Shyr Dai
Dept. Computer Science &
Information Engineering
National Taiwan University
No. 1, Sec. 4, Roosevelt Rd.
Taipei, Taiwan

Yuh-Dauh Lyuu
Dept. Computer Science &
Information Engineering*
National Taiwan University
No. 1, Sec. 4, Roosevelt Rd.
Taipei, Taiwan
886-2-23625336 x429 (o)
886-2-23628167 (fax)
lyuu@csie.ntu.edu.tw
and
Department of Finance
National Taiwan University
No. 1, Sec. 4, Roosevelt Rd.
Taipei, Taiwan

Abstract

Asian options are a kind of path-dependent derivatives. How to price such derivatives efficiently and accurately has been a long-standing research and practical problem. This paper proposes a novel multiresolution (MR) trinomial lattice for pricing European- and American-style arithmetic Asian options. Extensive experimental work suggests that this new approach is both efficient and more accurate than existing methods. It also computes the numerical delta accurately. The MR algorithm is exact as no errors are introduced during backward induction. In fact, it may be the first exact discrete-time algorithm to break the exponential-time barrier. The MR algorithm is guaranteed to converge to the continuous-time value.

JEL Classification Codes: G12, G13

*The preferred corresponding address.

Keywords: Asian options, path-dependent options, trinomial model, multiresolution

Path-dependent derivatives are derivatives whose payoffs depend nontrivially on the price history of the underlying asset. Some path-dependent derivatives such as barrier options, lookback options, and geometric Asian options can be efficiently priced (see Lyuu (1998) and Lyuu (2002)). Others, however, are known to be difficult to price in terms of speed and/or accuracy. The (arithmetic) Asian option is perhaps the most prominent example in this category. The Asian option has a payoff that is determined by the (arithmetic) average price of the underlying asset. It is useful for hedging transactions whose cost is related to the average price of the asset. It is also harder to manipulate. This feature is especially important in thin markets.

Assume the asset price follows the lognormal diffusion,

$$\frac{dS}{S} = r\,dt + \sigma\,dW,$$

where W is the standard Wiener process, r is the risk-free interest rate per annum, and σ is the annual volatility. In the discrete-time setting, define $\triangle t \equiv \tau/n$, where τ is the option's time to maturity (in years) and n is the number of periods τ is divided into. The payoff of the option depends on the (arithmetic) sum of the asset prices. Specifically, define the (arithmetic) average by

$$A_{\text{avg}}(n) \equiv \frac{S_0 + S_1 + \cdots + S_n}{n+1}.$$

Here S_i denotes the underlying asset's price at (discrete) time i, which corresponds to absolute time $i\Delta t$ in the continuous-time model. The payoff for the Asian call is $\max(A_{\text{avg}}(n) - X, 0)$, where X is called the exercise price. The payoff for the Asian put is $\max(X - A_{\text{avg}}(n), 0)$. Our task is to compute the discounted expected values of the above payoffs in such a way that the values converge to the continuous-time limits $e^{-r\tau} E[\,\max(\frac{1}{\tau}\int_0^\tau S(t)\,dt - X, 0)\,]$ and $e^{-r\tau} E[\,\max(X - \frac{1}{\tau}\int_0^\tau S(t)\,dt, 0)\,]$. American-style Asian options are identical except for the early exercise feature. The algorithms should also price American-style options accurately. Although we will concentrate on the call option in this paper, the results hold for puts as well.

The major contribution of this paper is a novel trinomial lattice that is general-purpose but most useful for pricing Asian options, which has been a long-standing problem when the underlying asset's price is lognormally distributed. Pricing on the lattice is efficient and accurate. Furthermore, unlike most other schemes, interpolation is not needed in backward induction. The algorithm is therefore an exact discrete-time algorithm. This characteristic is in sharp contrast to existing discrete-time algorithms; these algorithms attempt to approximate the naive but convergent exponential-time algorithm, which simply evaluates each of the 3^n paths on an n-period trinomial model. Our algorithm can price both European- and American-style options. Convergence to the continuous-time value is guaranteed as the lattice matches the first and second moments of the continuous-time model at each node (see Duffie (1996)). Such theoretical guarantee is lacking in many other approximation schemes.

The idea is deceptively simple. It is well-known that the option value is homogeneous of degree one in the asset price (see Merton (1994)). We can thus multiply the exercise price and all the asset prices on the lattice by some number $x > 0$, price the option, and finally divide the calculated option value by x. This, together with the extra degrees of freedom afforded by the trinomial model, will be exploited to construct a trinomial lattice with *integral* asset prices. This means that the price sum of any path on the lattice will be integers as well. (Recall that the payoff is determined by the price sum.) The key insight can now be stated: The price sums of paths reaching any given node are finite in number and *enumerable*, being integers between some integral minimum and maximum price sums. Take the hypothetical 3-period integral trinomial lattice in Figure 1 for example. The underlying asset's prices are printed on the nodes. Consider the price paths that reach the shaded node with an asset price of 4. The paths with the maximum price sum and the minimum price sum are $(8, 12, 8, 4)$ and $(8, 4, 2, 4)$, respectively. The maximum and minimum price sums are thus $8+12+8+4 = 32$ and $8+4+2+4 = 18$, respectively. Now, the possible price sums at that node must be some of the 15 integers between 18 and 32, inclusively. Notice how the price sums have been enumerated without going through every possible path. The integral property will eventually allow backward induction to dispense with approximations. The algorithm is thus exact.

In practice the algorithm will not really be multiplying asset prices on the lattice as there exists a more efficient, yet equivalent, implementation. By insisting that the multiplication factor x be a power of two, multiplication amounts to shifting the prices by j bits to the left when $x = 2^j$, where $j \geq 0$. In reality, this step will be replaced by extending the asset prices' precision after the decimal point by j bits. For example, instead of multiplying 101 (base 2) by $x = 2^2$ to yield 10100, treat 101 as 101.00 and perform thereafter any arithmetic operations with it using two bits of precision after the decimal point. Thus asset prices will simply be rational numbers of finite precision. We may switch back and forth between the two equivalent interpretations—multiplication and precision extension—whichever is demanded by clarity.

It is not necessary to apply the extension of precision mentioned above to all the lattice nodes. When a node N has extended precision, those nodes reachable from it need to have extended precision as well. The reason is that a path passing through node N adds N's price, which has extended precision, to the price sum. On the other hand, nodes not reachable from node N need not have the same precision as N. This observation will be employed to reduce the complexity further and result in nodes with varying precisions, thus the term multiresolution (MR). The general notion of multiple resolution has been used explicitly in image processing since as early as 1975 (see Rosenfeld (1984)) and, in the finance literature, is implicit in the adaptive mesh of Ahn, Figlewski, and Gao (1999).

Nothing is gained with the above manipulations unless the number of states is reduced from 3^n for a trinomial lattice of depth n to a much more manageable

number. Recall that, here, the number of states is the number of price sums. It turns out that limiting the stock prices to finite-precision rational numbers does drastically reduce the possible number of price sums. For example, with 160 time periods, the total number of paths to the middle node at expiration is the astronomical

$$\sum_{i=0}^{80} \binom{160}{i} \binom{160-i}{i} \approx 8.429 \times 10^{74}.$$

No computers are expected to face down this number now, or in the future. But the total number of price sums at that node is at most 57887 under the typical scenario in Figure 2(a), which also gives a detailed account of the numbers of price sums at other nodes as well. (The number 57887 will be verified in a later section.) Thus a lot of paths produce the same price sums on the MR lattice. That is why the MR algorithm can price Asian options in the case of 160 periods. We are led to conclude that the MR algorithm has broken the 3^n barrier while remaining an exact algorithm. It may be the first exact discrete-time algorithm to break the 3^n barrier.

Extensive computer experiments suggest that the MR algorithm is superior to the approaches in the literature sampled above in terms of both convergence and accuracy. It is also a practical algorithm. Figure 3 plots a typical convergence behavior of the algorithm. Note that it quickly converges to the continuous-time value. Sensitivity measures such as delta are also straightforward to compute with the MR algorithm. Hence hedging the Asian option presents no fundamental problems.

We now survey the literature. Approximate closed-form solutions are suggested in Levy (1992), Milevsky (1998), and Turnbull and Wakeman (1991). Geman and Yor (1993) derive an analytical expression for the Laplace transform of the Asian call. Numerical inversion of this transform is considered in Geman and Eydeland (1995) and Shaw (1998). Some inversion algorithms based on the Euler and Post-Widder methods can be found in Abate and Whitt (1995).

Because no simple closed-form solutions exist yet for the Asian option, the development of efficient numerical algorithms is critical. First, there are the popular Monte Carlo and related quasi-Monte Carlo methods; see Boyle, Broadie, and Glasserman (1997), Broadie and Glasserman (1996), Broadie, Glasserman, and Kou (1999), and Kemna and Vorst (1990). But both the Monte Carlo approach and the analytical approach suffer from the inability to handle early exercise without bias. Recently, Longstaff and Schwartz (2001) have developed a least-squares Monte Carlo approach to tackle the problem.

Tree methods, to which our algorithm also belongs, and the related discretized partial-differential-equation approach are more general than the above-mentioned schemes because they can easily incorporate early exercise. The difficulty with the tree method in the case of Asian options lies in its exponential nature. Many proposed approaches to solve this combinatorial explosion augment a state variable to each tree node, which is usually the price sum or, equivalently, the price average. A

very successful approximation paradigm by Hull and White (1993) limits the number of price sums at each node of the binomial tree to some manageable magnitude k. It then resorts to interpolation in backward induction; see Hull and White (1993), Klassen (2001), and Ritchken, Sankarasubramanian, and Vijh (1993). This approach is efficient, with a running time of $O(kn^2)$, but it is no longer exact because errors are introduced by interpolation. The convergence issue of such numerical algorithms is analyzed in Barraquand and Pudet (1996) and Forsyth, Vetzal, and Zvan (2001). Another paradigm seeks approximation algorithms that produce provable upper and lower bounds (called range bounds) that bracket the option value; see Chalasani et al. (1999) and Rogers and Shi (1995). The $O(kn^2)$-time algorithm of Aingworth, Motwani, and Oldham (2000) guarantees a theoretical error bound of $O(Xn/k)$ for pricing European-style options, where k can be varied for any desired trade-off between time and accuracy. Akcoglu, Kao, and Raghavan (2001) derive a complex trade-off by a recursive application of the above algorithm in the case of European-style Asian options. All of the algorithms resort to either interpolation (or its extreme form, rounding) or analytical approximation in pricing. They are therefore approximation algorithms. In contrast, the MR algorithm is an exact discrete-time pricing algorithm.

1 Integral Trinomial Lattice and Option Pricing

Let $N(i,j)$ denote the node on the trinomial lattice that has the jth largest asset price at time i, where $1 \leq j \leq 2i+1$. $S(N)$ will represent the asset price at node N. The trinomial lattice's topology is illustrated in Figure 4. Node N may move to node $u(N)$ by the up branch, to node $m(N)$ by the flat branch, and to node $d(N)$ by the down branch. For the flat branch, $S(m(N)) = S(N)$; hence the asset price does not change if the flat branch is taken. Use $p_u(N)$, $p_m(N)$, and $p_d(N)$ to denote the branching probabilities for the up, flat, and down nodes from node N. The trinomial lattice recombines as shown in Figure 5.

Suppose the asset prices on the lattice are all positive integers throughout this section. Then the sums of asset prices are positive integers. Denote the maximum sum from the root at time 0 to node $N(i,j)$ by $N_{\max}(i,j)$ and the minimum sum by $N_{\min}(i,j)$. Both numbers are straightforward to calculate. A price sum for a path from the root at time 0 to node $N(i,j)$ at time i must belong in the set,

$$N_\Sigma(i,j) \equiv \{k : k \text{ is an integer}, N_{\min}(i,j) \leq k \leq N_{\max}(i,j)\}.$$

The critical observation is that $N_\Sigma(i,j)$ is finitely enumerable. Without figuring out the exact number of price sums by exhaustive search, we simply take the easily derivable $N_\Sigma(i,j)$ as our state space for node $N(i,j)$, and the number of allocated states becomes

$$|\, N_\Sigma(i,j) \,| = N_{\max}(i,j) - N_{\min}(i,j) + 1.$$

The above is an upper bound on the total number of valid price sums at node $N(i,j)$. Data will demonstrate that the bound is tight in practice. The computational complexity of the pricing problem will be proportional to the total number of allocated states, $\sum_{i=0}^{n} \sum_{j=1}^{2i+1} \mid N_\Sigma(i,j) \mid$.

We now have enough information to present the formula for pricing Asian options. Let $V(i,j,k)$ denote the option value at node $N(i,j)$ given that the price sum at the node is k. Backward induction says that

$$
\begin{aligned}
(1) \qquad V(i,j,k) \;=\; & [\, p_u\, V(i+1, j, k + S(N(i+1,j))) + \\
& p_m\, V(i+1, j+1, k + S(N(i+1,j+1))) + \\
& p_d\, V(i+1, j+2, k + S(N(i+1,j+2)))\,]\, e^{-r\,\Delta t},
\end{aligned}
$$

where $0 \le i < n$, $1 \le j \le 2i+1$, and $k \in N_\Sigma(i,j)$. The dependency on parameter $N(i,j)$ is dropped from p_u, p_m, and p_d for brevity. American-style options can be handled similarly:

$$
\begin{aligned}
V(i,j,k) \;=\; & \max\{\, \frac{k}{i+1} - X, \\
& [\, p_u\, V(i+1, j, k + S(N(i+1,j))) + \\
& p_m\, V(i+1, j+1, k + S(N(i+1,j+1))) + \\
& p_d\, V(i+1, j+2, k + S(N(i+1,j+2)))\,]\, e^{-r\,\Delta t}\}.
\end{aligned}
$$

See Figure 6 for illustration. Both options have payoff

$$V(n,j,k) = \max[\, k/(n+1) - X, 0\,]$$

at maturity.

The advantage of the integral lattice over alternative approximation schemes is now crystal clear. Observe that the state space at each node is finitely enumerable because of the finitude and integrality of $N_\Sigma(i,j)$. As a consequence, backward induction can be carried out exactly as above, and no interpolations are required. The only source of error compared with the continuous-time model is therefore discretization error arising from using a discrete-time model. In contrast, suppose the asset prices are real numbers instead. Then the sum of the price sum k and the asset price to follow node $N(i,j)$ in backward induction, $S(N(i+1,m))$ where $m \in \{\, j, j+1, j+2 \,\}$, will produce real numbers. Although the price sum $k + S(N(i+1,m))$ is bounded between $N_{\min}(i+1,m)$ and $N_{\max}(i+1,m)$ at node $N(i+1,m)$ as before, the $\binom{i+1}{m}$ possible sums at node $N(i+1,m)$ cannot be identified without exhaustive search, which takes exponential time. Two alternatives are open. We can obtain the price sums by going through all possible paths. But this results in combinatorial explosion as just mentioned. For example, the number of paths ending at node $N(160,161)$ is about 8.429×10^{74}. The second alternative is to resort to approximation schemes, for which interpolation is a popular choice. But it introduces errors. Both alternatives have their disadvantages, which are unavoidable as long as asset prices are real numbers.

2 Lattice Construction

We now turn to the issue of finding integral asset prices for the lattice. Let

$$\begin{aligned} \mu(N) &= S(N)\, e^{r\,\Delta t}, \\ \mathrm{Var}(N) &= S(N)^2 e^{2r\Delta t}(e^{\sigma^2\Delta t} - 1) \end{aligned} \tag{2}$$

denote the mean and variance of the asset value one period from node N, respectively. To guarantee that every asset price is an integer, we impose

$$\begin{aligned} S(u(N)) &= S(N) + u_N, \\ S(d(N)) &= S(N) - d_N \end{aligned}$$

for some positive integers u_N and d_N for node N. Courtesy of the flat branches, the asset prices in the interior nodes such as nodes C, F, G, and H in Figure 5 are uniquely determined by the asset prices in the boundary nodes such as nodes B, A, and D in the same figure.

The positive integers u_N and d_N will be found by the following considerations. To guarantee convergence to the continuous-time option value, the lattice is calibrated to the first and second moments of the underlying asset price at each node N:

$$\mu(N) = p_u(N)(S(N) + u_N) + p_m(N)\, S(N) + p_d(N)(S(N) - d_N) \tag{3}$$

$$\begin{aligned} \mathrm{Var}(N) &= p_u(N)[\, S(N) + u_N - \mu(N)\,]^2 + p_m(N)[\, S(N) - \mu(N)\,]^2 + \\ &\quad p_d(N)[\, S(N) - d_N - \mu(N)\,]^2 \end{aligned} \tag{4}$$

$$1 = p_u(N) + p_m(N) + p_d(N) \tag{5}$$

$$0 < p_u(N), p_m(N), p_d(N) < 1 \tag{6}$$

But two issues have been glossed over. First, the trinomial lattice as stated is underdetermined because there are more variables (five) than equations (three) at each node. Second, what happens if integral, positive displacements u_N and d_N satisfying conditions (3)–(6) cannot be found? We proceed now to show how the extra freedom of the lattice—the first issue—can actually be utilized to address the second issue.

The algorithmic idea will be expounded with reference to the lattice in Figure 5. We only need to work on boundary nodes E, B, A, D, and I as the interior nodes' asset prices follow automatically. Because the root node A has asset price S, the interior nodes C and G at the same level must also have the same asset price S. Now choose

$$\left\lceil \sqrt{\mathrm{Var}(N)} \right\rceil \tag{7}$$

for the up displacement $u \equiv u_N$ and the down displacement $d \equiv d_N$ from the root node $N \equiv$ A. This determines the asset prices at B and D as well as the interior nodes at their levels: nodes F and H. From node D, the integral down displacement f can be

determined by formula (7), which then implies the asset price at I. The computation from node B is symmetric to that from node D in determining node E's asset price.

In general, at any given time, only the top boundary node (like nodes B and E) and the bottom boundary node (like nodes D and I) need to have their displacements calculated. The displacements at the interior nodes (like C, F, G, and H) are given by the leftmost boundary node at the same level. The same holds for the branching probabilities. In fact, only the up displacements at the top boundary nodes need to be calculated, and only the down displacements at the bottom boundary nodes need to be calculated. These properties follow from the lattice topology.

Consider the lattice in Figure 5 with $S = 50$, $r = 10\%$, $\sigma = 30\%$, $\tau = 0.5$, and $n = 30$. The asset prices at nodes A, C, and G are automatically 50. The variance of the asset price at the end of the first period is 3.76534 by formula (2). The up and down displacements u and d are then determined by formula (7) to be 2. The branching probabilities can be solved via (3)–(5) to be $p_u(\mathsf{A}) = 0.492$, $p_m(\mathsf{A}) = 0.057$, and $p_d(\mathsf{A}) = 0.451$. They are indeed valid probabilities. The asset prices at nodes B and D are hence 52 and 48, respectively. The asset price at node F is then 52 and that at node H is 48. Continuing with the calculations, we will find that the asset price at node E is 55 and that at node I is 46.

One problem remains. It may happen that the integral displacement given by formula (7) does not satisfy all the conditions (3)–(6). This can only occur along the bottom path of decreasing asset prices like (A, D, I) in Figure 5. The obvious reason is that, compared to a larger asset price, a smaller asset price entails a smaller $\sqrt{\mathrm{Var}(N)}$, which, when rounded, induces a bigger percentage error.

The key to solving the problem is that the option value is homogeneous of degree one in the asset price. Hence, we may construct the lattice for m times the asset prices for some integer $m > 0$ and divide the computed option value by m later. Since m is an integer, the asset prices remain integers. Specifically, when formula (7) fails to give branching probabilities satisfying (3)–(6) at a node N along the bottom path on the lattice, we search for an $m > 1$ so that, after all the current asset prices are multiplied by it, an integral down displacement that satisfies all the said conditions can be found among $\{ 1, 2, \ldots, m-1 \}$. For ease of use on the binary computer, integers $2, 4, 8, 16, \ldots$ are tried for m, in that order. When an m which results in a valid choice for the positive down displacement is found, all asset prices on the lattice are multiplied by it. In general, if $m_1, m_2, \ldots, m_\ell$ are used for m during the construction, the lattice will be one for $\prod_{i=1}^{\ell} m_i$ times the asset prices, and the resulting option value must be divided by the same number.

In practice, the multiplication of asset prices is not actually carried out for efficiency's reasons. As the value of m is some power of two, the multiplication step is replaced by extending the precision bits. More precisely, if m is found to be 2^j, multiplication by m is equivalent to allocating j additional precision bits after the decimal point to the asset price. As a bonus, the computed option value need no longer be divided by m at the end.

To give a more precise explanation for the above ideas, define

$$A_j \equiv \left\{ \frac{i}{2^j} \;\middle|\; i = 1, 2, \ldots \right\}, \quad j \text{ is a positive integer,}$$

the set of numbers with j precision bits after the decimal point. Note that A_0 is the set of positive integers and $A_0 \subset A_1 \subset A_2 \subset \cdots$. The crucial property to note is that A_j, like positive integers, is closed under addition: A sum of numbers from A_j remains in A_j. Define $\lceil x \rceil_k$ as the number which results from rounding *up* x to k precision bits after the decimal point. For example, $\lceil 1.0101 \rceil_3 = 1.011$ (base 2) and $\lceil 1.0100 \rceil_3 = 1.010$ (base 2).

We are now able to give a fuller account of the transition to extra precision bits. Suppose we are working under A_k when either $d_N = \lceil \sqrt{\mathrm{Var}(N)} \rceil_k > 0$ fails to give valid probabilities or $\sqrt{\mathrm{Var}(N)} < 2^{-(k+1)}$. When this happens, we will proceed to extend the precision. Let

$$2^{-(\ell+1)} \leq \sqrt{\mathrm{Var}(N)} < 2^{-\ell}$$

for some $\ell > k$. Then for each $j = \ell, \ell+1, \ldots$ we locally search the following 2^{j-k-1} numbers in A_j for a number d_N that leads to valid probabilities:

$$\begin{array}{rcl}
2^{-j} + 0 \times 2^{-j+1} & = & 0.\overbrace{0\cdots0\,0\cdots0001}^{j} \text{ (base 2)} \\
2^{-j} + 1 \times 2^{-j+1} & = & 0.0\cdots0\,0\cdots0011 \text{ (base 2)} \\
2^{-j} + 2 \times 2^{-j+1} & = & 0.0\cdots0\,0\cdots0101 \text{ (base 2)} \\
2^{-j} + 3 \times 2^{-j+1} & = & 0.0\cdots0\,0\cdots0111 \text{ (base 2)} \\
2^{-j} + 4 \times 2^{-j+1} & = & 0.0\cdots0\,0\cdots1001 \text{ (base 2)} \\
\vdots & \vdots & \vdots \\
2^{-k} - 2^{-j} & = & 0.\underbrace{0\cdots0}_{k}1\cdots1111 \text{ (base 2)}
\end{array}$$

Call this set A_j^*. We stop the moment such a j is found. The construction can now continue, working under A_j.

In practice, the need to add precision happens only occasionally. In the 160-period lattice of Figure 2, for example, it happens four times. Furthermore, when it does happen, usually only one bit of precision is added as testified by the lack of bigger-than-one jumps in Figure 2(b). This implies that, usually, $d_N = 2^{-(k+1)}$. Both work to our advantage.

3 Additional Optimization: Multiresolution

When an asset price is multiplied by m (equivalently, has $\log_2 m$ extra precision bits after the decimal point), it increases the state space, and thus the running time, m-fold because $|\, N_\Omega \,|$ becomes m times as large. Fortunately, such scaling does not have

to be applied to the whole lattice. Refer to the MR lattice in Figure 7(a). Let node I be the first node along the bottom path from the root to have a fractional asset price, say from set A_1. Clearly, the price sums at nodes not reachable from I remain in A_0. These nodes lie to the "north" and "west" of the path (I, N). Only those nodes reachable from node I need to have at least 1 bit of precision after the decimal point for their price sums. This results in an MR lattice.

The above idea can be generalized naturally. In Figure 8, let node A be the first node from the root along the bottom path L_3 to have an asset price in A_a ($a > 0$), node B be the first node from the root along the bottom path to have an asset price in A_b ($b > a$), etc. Every node to the north and west of line L_1 has price sums in A_0; every node to the north and west of line L_2 but below L_1 (inclusive) has price sums in A_a; every node below L_2 (inclusive) has price sums in A_b; and so on. The savings are substantial as shown in Table 1 based on the parameters in Figure 2. The reason is also evident from looking into Figure 2(b): Although the asset prices need to be conceptually multiplied by up to $2^4 = 16$ to achieve integrality, the majority are multiplied by the much smaller 1, 2, 4, and 8 if the idea of multiresolution is incorporated.

As an example to illustrate the above ideas, we now verify that the number 57887 stated in the introductory section is the number of states at the middle terminal node $N(160, 161)$. According to our computer program, the maximum price sum is 36414 and the minimum price sum is 7471. Since Figure 2(b) reveals that this node's price sums belong in A_1, the maximum and minimum price sums are to be multiplied by two to yield $N_{\max}(160, 161) = 72828$ and $N_{\min}(160, 161) = 14942$, respectively. A price sum at node $N(160, 161)$—after the multiplication of all asset prices by two—must be an integer between $N_{\max}(160, 161)$ and $N_{\min}(160, 161)$, inclusively. The number of allocated states at node $N(160, 161)$ thus equals

$$| N_{\Sigma}(160, 161) | = N_{\max}(160, 161) - N_{\min}(160, 161) + 1 = 57887,$$

as claimed.

4 Example: A 3-Period Integral Trinomial Lattice

We use the 3-period lattice in Figure 7 to illustrate our ideas. At the root node A and working under A_0, the standard deviation one period forward can be found to be $\sqrt{0.2641} = 0.514 > 2^{-1}$ by formula (2). So the candidate up and down displacements equal 1 by formula (7). As they produce the valid probabilities in the A-C-G row of Figure 7(b), we move down to node D. The standard deviation one period forward from node D is $\sqrt{0.1690} = 0.4112$ by formula (2). Since $2^{-2} \leq 0.4112 < 2^{-1}$, we look first for a solution in $A_1^* = \{ 2^{-1} \}$. The only candidate for down displacement d_{D} is 2^{-1}, and it results in valid probabilities in the D–H row of Figure 7(b), satisfying conditions (3)–(6). Node I's asset price is hence $4 - 2^{-1} = 3.5$, which is 11.1 (base 2).

Now go further down the bottom path and work under A_1. The standard deviation one period forward from node I equals $\sqrt{0.1294} = 0.3598$. Since $2^{-2} \leq 0.3598 < 2^{-1}$, again 2^{-1} is selected as the candidate down displacement from node I. As it leads to valid probabilities in the I row of Figure 7(b), the asset price at node P equals $3.5 - 2^{-1} = 3.0$, which equals 11.0 (base 2). The calculations for top boundary nodes B, E, and J are similar.

After the lattice is constructed, we allocate the state space $N_\Sigma(i, j)$ for each node $N(i, j)$ and proceed to carry out backward induction. Consider node G in Figure 9, extracted from Figure 7. The maximum price sum at node G equals $101+110+101 =$ 10000 (base 2) or 16 (base 10), and the minimum price sum equals $101 + 100 +$ $101 = 1110$ (base 2) or 14 (base 10). All price sums at node G are in A_0 because the underlying asset's prices on any path leading to G belong in A_0. As a result, the possible price sums at node G must be integers between 14 and 16, i.e., in set $N_\Sigma(\mathsf{G}) = \{\, 14, 15, 16 \,\}$. The total number of allocated states is therefore 3. Take node N that belongs in A_1 as another example. The maximum price sum equals $101 + 110 + 101 + 100.0 = 10100.0$ (base 2) or 20 (base 10). The minimum price sum equals $101 + 100 + 11.1 + 100.0 = 10000.1$ (base 2) or 16.5 (base 10). Because node N belongs in A_1, price sums there must be numbers between 16.5 and 20.0 with an increment of 2^{-1}, i.e., in set

$$N_\Sigma(\mathsf{N}) = \{\, 16.5, 17.0, 17.5, 18.0, 18.5, 19.0, 19.5, 20.0 \,\}.$$

The number of allocated states is 8. Similar computations can be carried out for nodes L and M. The states at nodes G, L, M, and N are shown in Figure 9.

Without the use of multiple resolution, many states would be wasted, leading to inefficiency. Consider the MR lattice in Figure 7(a) again. Because an extra precision bit is eventually needed at node I, every asset price should add an extra precision bit, potentially doubling the state space, if the notion of multiple resolution were not in place. The states at node G, for example, would then be rational numbers between 14.0 and 16.0 with an increment of 2^{-1}, or in $\{\, 14.0, 14.5, 15.0, 15.5, 16.0 \,\}$ instead of the earlier and smaller $\{\, 14, 15, 16 \,\}$.

Once the lattice is in place, backward induction can start. Assume $n = 3$ and the exercise price is $X = 4.8$. From node G with price sum 16, the asset price can move upward to terminal node L with price sum $16 + 6 = 22$, move flatly to terminal node M with price sum $16 + 5 = 21$, or move downward to terminal node N with price sum $16 + 4 = 20$. The option value for the state corresponding to the price sum 22 at terminal node L equals $(22/4) - 4.8 = 0.7$. Similarly, the option values for the states corresponding to the price sum 21 at terminal node M and the price sum 20 at terminal node N are $(21/4) - 4.8 = 0.45$ and $(20/4) - 4.8 = 0.2$, respectively. The option value for the state corresponding to price sum 16 at node G can be computed by applying backward-induction formula (1) as follows:

$$(0.203 \times 0.7 + 0.720 \times 0.45 + 0.077 \times 0.2) \times e^{-0.1 \times 0.75/3} = 0.470.$$

Option values corresponding to price sums 15 and 14 at node G can be computed similarly as follows:

$$\begin{aligned}(0.203 \times 0.45 + 0.720 \times 0.2 + 0.077 \times 0.0) \times e^{-0.1 \times 0.75/3} &= 0.230, \\ (0.203 \times 0.2 + 0.720 \times 0.0 + 0.077 \times 0.0) \times e^{-0.1 \times 0.75/3} &= 0.040.\end{aligned}$$

After similar calculations at other nodes, the final option value at node A is 0.415.

Now is a good time to see how the MR algorithm differs radically from other lattice-based algorithms. Suppose the asset price for node L in Figure 9 is a real number 6.43 instead of the integer 6. From node G with price sum 16, the asset price can move upward to node L with price sum $16 + 6.43 = 22.43$. But there is no state corresponding to price sum 22.43 at node L. Algorithms such as Hull and White (1993) hence must resort to interpolation or even rounding, an extreme form of interpolation, to obtain an *approximate* option value corresponding to 22.43 from, say, option values corresponding to price sums 22 and 23.

Recall that $|N_\Sigma(i,j)|$, the number of allocated states at node $N(i,j)$, is an upper bound on the number of valid price sums. Their difference, a measure of wasted states, tends to be a small proportion of the total number of states in practice. In Figure 10, for example, all nodes with a large number of states have high portions of the states being valid price sums. This should make the overall usage ratio high. Indeed, 94.4258% of the total states at the 61 terminal nodes correspond to valid price sums; less than 6% of the states are wasted. The MR algorithm is therefore highly efficient in its usage of memory and computing resources.

5 Numerical Results

There are at least two problems with most existing approaches. One is that they may not be applicable to American-style options. The other is that most approaches fail to get acceptable results for some cases as pointed out in Fu, Dilip, and Wang (1998/9). Take the prominent Hull-White algorithm as an example. One version of the algorithm is based on linear interpolation. In Table 2 the values under Hull-White/linear increase monotonically with n and do not seem to converge. The version with exponential interpolation also does not seem robust when n is large (see Table 2 under Hull-White/expo). Both are consistent with the analysis in Forsyth, Vetzal, and Zvan (2001). Since exponential interpolation performs slightly better, by Hull-White algorithm we will refer exclusively to this version unless stated otherwise. Other algorithms are also compared in the same figure such as the algorithm of Aingworth, Motwani, and Oldham (2000) (called AMO for brevity). The MR algorithm converges well and does not overprice the options as do the Hull-White algorithms when n increases. It also converges better than the AMO algorithm for the same n.

The running time of the MR algorithm is reasonable and much less than 3^n. For example, the algorithm finishes in 145 seconds for $n = 141$, whereas $3^{141} \approx 1.88 \times 10^{67}$,

making the naive $O(3^n)$-time algorithm hopeless.

Delta is key to hedging and replication. It is therefore important that the MR algorithm compute the option delta accurately. The computing of delta is a straightforward by-product of option pricing in the MR algorithm. Refer to Figure 5, with root node A. Now note that each of nodes B and D has only one state (price sum). This means that node B has only one option value f_{B}, corresponding to asset price $S+u$, and node D has only one option value f_{D}, corresponding to asset price $S-d$. Both option values are intermediate results in the process of pricing the option. The numerical delta is therefore given by $(f_{\mathsf{B}}-f_{\mathsf{D}})/(u+d)$; see Pelsser and Vorst (1994). Table 2 shows that the MR algorithm does a good job in the calculation of delta. Figure 11 shows further that the numerical delta as determined by the MR algorithm varies smoothly with asset price S. We therefore do not expect problems in constructing hedge portfolios.

Additional experiments are tabulated in Table 3. With $n=30$, the running time is about 2 seconds on an Intel Pentium II 233MHz computer. Most of the values computed by the MR algorithm are close to the value computed by Monte Carlo simulation. Another set of experiments are focused on extreme cases mentioned in Fu, Dilip, and Wang (1998/9). In that paper, the authors compare many proposed algorithms and conclude that some algorithms may fail in extreme cases. We test their extreme cases, and the results are shown in Table 4. The MR algorithm performs well in each and every one of them.

All the experimental results up to now are for European-style Asian options. Table 5 tabulates American-style Asian option values generated by various algorithms: the Hull-White algorithm, the upper- and lower-bound algorithms of Chalasani et al. (1999), and the MR algorithm. Both the Hull-White and MR algorithms generate results that exceed the upper bounds of Chalasani et al. (1999). Since the bounds of Chalasani et al. (1999) are valid for the CRR binomial model only, that the Hull-White and MR algorithms' results lie outside the bounds does not prevent them from being closer to the continuous-time limits. In fact, judging from the MR algorithm's excellent convergence in the European-style case, we suggest that the bounds of Chalasani et al. (1999) may actually underestimate the continuous-time limits for any finite n. Our claim is consistent with the Hull-White algorithm's similar tendency to exceed their upper bounds. Lack of a benchmark like Monte Carlo simulation in the European-style case necessarily means that our assessments are preliminary.

6 Conclusions

This paper develops a new trinomial lattice particularly with the Asian option in mind. The lattice uses the notion of integrality of asset prices and multiple resolution to make an exact pricing algorithm realizable and practical. The first property is made possible by the well-known fact that the option value is homogeneous of degree

one in the asset price, while the second property is made possible by simple causal considerations. No errors are introduced by the algorithm in backward induction. The algorithm can handle both European- and American-style options easily. Extensive experiments show that the algorithm compares favorably with existing approaches. Its practical running time means that it may be the first exact algorithm to break the 3^n barrier. The source of the tremendous reduction in running time is the dramatic decrease in the possible number of price sums.

The MR lattice can be easily modified to price Asian options when averaging is applied to daily closing prices. It can also be used to price other path-dependent options whose payoff depends on the average price such as average-strike options and Asian barrier options in Zvan and Vetzal (1999).

Acknowledgements

We thank Prof. Ming-Yang Kao of Northwestern University and Dr. Guang-Shieng Huang and Ming-Chen Sun of National Taiwan University for discussions.

References

[1] Abate, J., and W. Whitt. (1995). "Numerical Inversion of Laplace Transforms of Probability Distributions," *ORSA Journal of Computing* 7, 36–43.

[2] Ahn, D.-H., S. Figlewski, and B. Gao. (1999). "Pricing Discrete Barrier Options with an Adaptive Mesh Model," *Journal of Derivatives* 6, 33–43.

[3] Aingworth, D., R. Motwani, and J.D. Oldham. (2000). "Accurate Approximations for Asian Options," In *Proc. 11th Annual ACM-SIAM Symposium on Discrete Algorithms.*

[4] Akcoglu, K., M.-Y. Kao, and S.V. Raghavan. (2001). "Fast Pricing of European Asian Options with Provable Accuracy: Single-Stock and Basket Options." *Lecture Notes in Computer Science*, 2161. Berlin: Springer-Verlag, 2001, pp. 404–415.

[5] Barraquand, J., and T. Pudet. (1996). "Pricing of American Path-Dependent Contingent Claims," *Mathematical Finance* 6(1), 17–51.

[6] Boyle, P.P., M. Broadie, and P. Glasserman. (1997). "Monte Carlo Methods for Security Pricing," *Journal of Economic Dynamics & Control* 21, 1267–1321.

[7] Broadie, M., and P. Glasserman. (1996). "Estimating Security Price Derivatives Using Simulation," *Management Science* 42, 269–285.

[8] Broadie, M., P. Glasserman, and S. Kou. (1999). "Connecting Discrete and Continuous Path-Dependent Options," *Finance and Stochastics* 3, 55–82.

[9] Chalasani, P., S. Jha, F. Egriboyun, and A. Varikooty. (1999). "A Refined Binomial Lattice for Pricing American Asian Options," *Review of Derivatives Research* 3, 85–105.

[10] Choa, H.Y., and H.Y. Lee. (1997). "A Lattice Model for Pricing Geometric and Arithmetic Average Options," *Journal of Financial Engineering* 6(3), 179–191.

[11] Duffie, D. (1996). *Dynamic Asset Pricing Theory.* 2nd ed. Princeton: Princeton University Press.

[12] Forsyth, P.A., K.R. Vetzal, and R. Zvan. (2001). "The Use of Interpolation in Pricing Path-Dependent Options: A Detailed Examination of Convergence for the Case of Asian Options," Working Paper, University of Waterloo.

[13] Fu, M.C., D.B. Dilip, and T. Wang. (1998/9). "Pricing Continuous Asian Options: a Comparison of Monte Carlo and Laplace Transform Inversion Methods," *Journal of Computational Finance* 2(2), 49–74.

[14] Geman, H., and A. Eydeland. (1995). "Domino Effect," *Risk* 8(4), 65–67.

[15] Geman, H., and M. Yor. (1993). "Bessel Processes, Asian Options, and Perpetuities," *Mathematical Finance* 3, 349–375.

[16] Hull, J., and A. White. (1993). "Efficient Procedures for Valuing European and American Path-Dependent Options," *Journal of Derivatives* 1, 21–31.

[17] Klassen, T.R. (2001). "Simple, Fast and Flexible Pricing of Asian Options," *Journal of Computational Finance*, 4(3), 89–124.

[18] Kemna, A.G.Z., and A.C.F. Vorst. (1990). "A Pricing Method Based on Average Asset Values," *Journal of Banking and Finance* 14, 113–129.

[19] Levy, E. (1992). "Pricing European Average Rate Currency Options," *Journal of International Money and Finance* 11, 474–491.

[20] Longstaff, F.A., and E.S. Schwartz. (2001). "Valuing American Options by Simulation: a Simple Least-Squares Approach," *Review of Financial Studies* 14(1), 113–147.

[21] Lyuu, Y.-D. (1998). "Very Fast Algorithms for Barrier Option Pricing and the Ballot Problem," *Journal of Derivatives* 5, 68–79.

[22] Lyuu, Y.-D. (2002). *Financial Engineering and Computation: Principles, Mathematics, and Algorithms*. Cambridge, U.K.: Cambridge University Press.

[23] Merton, R.C. (1994). *Continuous-Time Finance*. Revised ed. Cambridge, MA: Blackwell.

[24] Milevsky, M.A., and S.E. Posner. (1998). "Asian Options, the Sum of Lognormals, and the Reciprocal Gamma Distribution," *Journal of Financial and Quantitative Analysis* 33, 409–422.

[25] Pelsser, A., and T. Vorst. (1994). "The Binomial Model and the Greeks," *The Journal of Derivatives* 1(3), 45–49.

[26] Ritchken, P., L. Sankarasubramanian, and A.M. Vijh. (1993). "The Valuation of Path Dependent Contracts on the Average," *Management Science* 39(10), 1202–1213.

[27] Rogers, L.C.G., and Z. Shi. (1995). "The Value of an Asian Option," *Journal of Applied Probability* 32(4), 1077–1088.

[28] Rosenfeld, A. (ed.) (1984). *Multiresolution Image Processing and Analysis.* Berlin: Springer-Verlag.

[29] Shaw, W.T. (1998). *Modeling Financial Derivatives with* Mathematica. Cambridge, U.K.: Cambridge University Press.

[30] Turnbull, S.M., and L.M. Wakeman. (1991). "A Quick Algorithm for Pricing European Average Options," *Financial and Quantitative Analysis* 26(3), 377–389.

[31] Zvan, R., and K. Vetzal. (1999). "Discrete Asian Barrier Options," *Journal of Computational Finance* 3, 41–68.

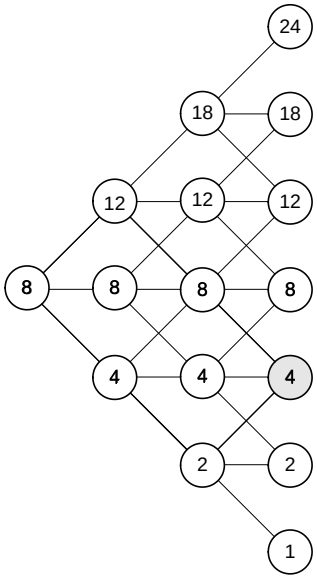
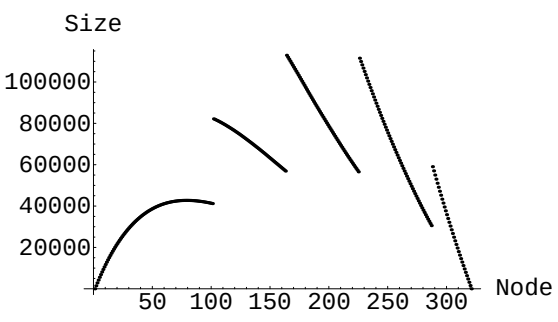
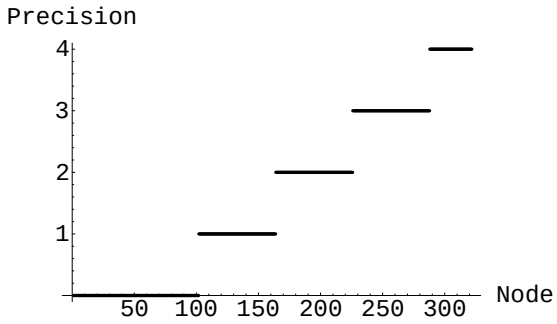


Figure 1: **A 3-Period Trinomial Lattice with Integral Asset Prices.** All paths reaching the shaded node have integral price sums. The maximum price sum at the shaded node is achieved by the upper path in thickened lines, whereas the minimum price sum at the shaded node is achieved by the lower path in thickened lines.



(a)



(b)

Figure 2: **Number of Price Sums under Multiple Resolution.** The upper bounds on the number of price sums at the terminal nodes are plotted in (a), starting from the node with the highest asset price. The sum of these upper bounds represents the total number of states allocated by the algorithm. In this particular case, the parameters are $S_0 = 100$, $X = 100$, $\sigma = 20\%$, $r = 10\%$, $\tau = 1$, and $n = 160$. There are $2n+1 = 321$ terminal nodes. The number of precision bits after the decimal point at the 321 terminal nodes are plotted in (b), starting from the node with the highest asset price. The maximum number of precision bits after the decimal point is 4.

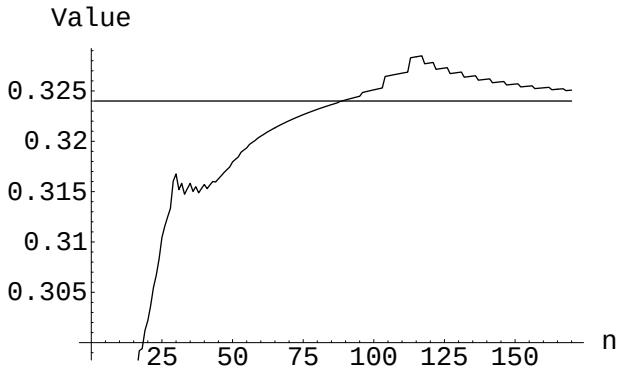


Figure 3: **Convergence Behavior of the MR Lattice.** The Monte Carlo simulation value from Choa and Lee (1997) is plotted for reference. The parameters are $S_0 = 50$, $X = 60$, $r = 10\%$, $\sigma = 30\%$, and $\tau = 0.5$.

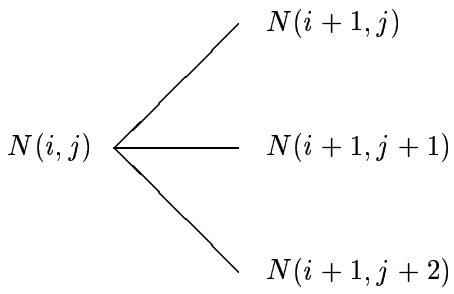


Figure 4: **The Trinomial Model.** Node $N(i,j)$ has the jth largest asset price at time i, where $1 \leq j \leq 2i+1$.

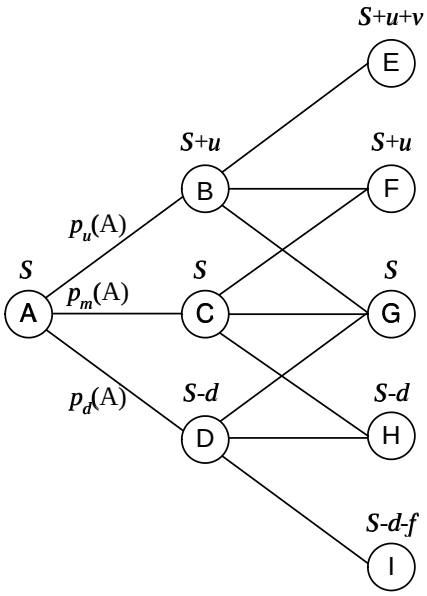


Figure 5: **The Trinomial Lattice.** The interior nodes are C, F, G, and H, whereas the boundary nodes are E, B, A, D, and I. Numbers $p_u(N)$, $p_m(N)$, and $p_d(N)$ denote the branching probabilities for the up, flat, and down nodes from node N. The up displacements are u and v, whereas the down displacements are d and f.

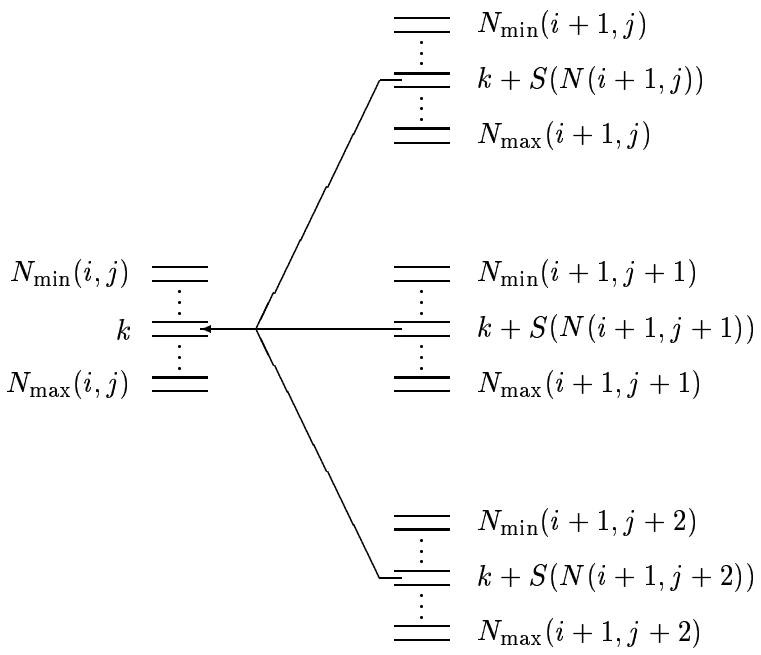
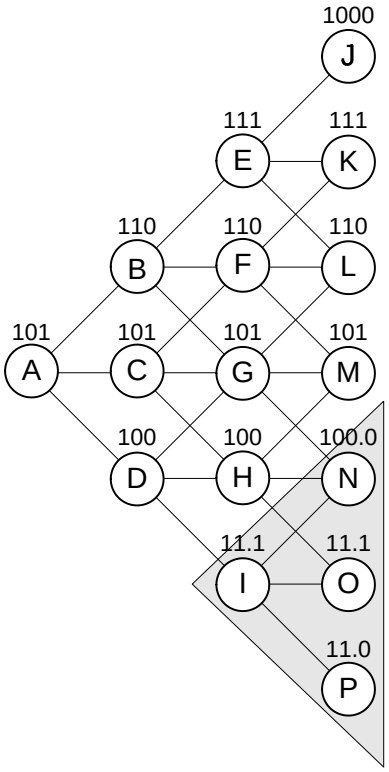


Figure 6: **Backward Induction for Asian Options.** The possible option values at $N(i,j)$ are stored in an array indexed by states which denote the possible price sums. Each option value depends on option values corresponding to specific states at the three nodes that follow: $N(i+1,j)$, $N(i+1,j+1)$, and $N(i+1,j+2)$.



(a)

	p_u	p_m	p_d
E	0.363	0.451	0.186
B F	0.278	0.597	0.126
A C G	0.203	0.720	0.077
D H	0.153	0.743	0.104
I	0.363	0.451	0.186

(b)

Figure 7: **A Sample MR Lattice.** (a) The shaded area covers nodes with an extra bit of precision after the decimal point (i.e., in A_1) for their price sums. All asset prices are binary numbers. The parameters are $S_0 = 5$, $r = 10\%$, $\sigma = 20\%$, $\tau = 0.75$, and $n = 3$. (b) In the table, p_u, p_m and p_d denote the up, flat, and down branching probabilities for each node, respectively.

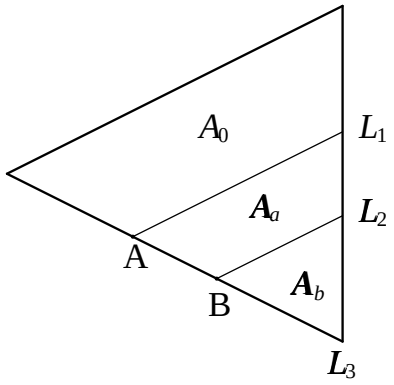


Figure 8: **Partition of the Lattice Based on Precisions.** All nodes reachable from the root but not reachable from node A have integral asset prices. All nodes reachable from node A but not reachable from node B have asset prices with a precision bits after the decimal point. All nodes reachable from node B have asset prices with b precision bits after the decimal point.

	Before optimization		After optimization			
n	100	160	100	135	150	160
Size	16,106,074	118,524,029	2,969,062	9,065,895	14,030,903	18,280,584

Table 1: **Optimization of the State Space.** The table records the total number of states allocated by the MR lattice. The input parameters are identical to those in Figure 2: $S_0 = 100$, $X = 100$, $\sigma = 20\%$, $r = 10\%$, $\tau = 1$, and $n = 160$. The number 18,280,584 under "After optimization/160," for example, is the sum of the numbers plotted in Figure 2(a).

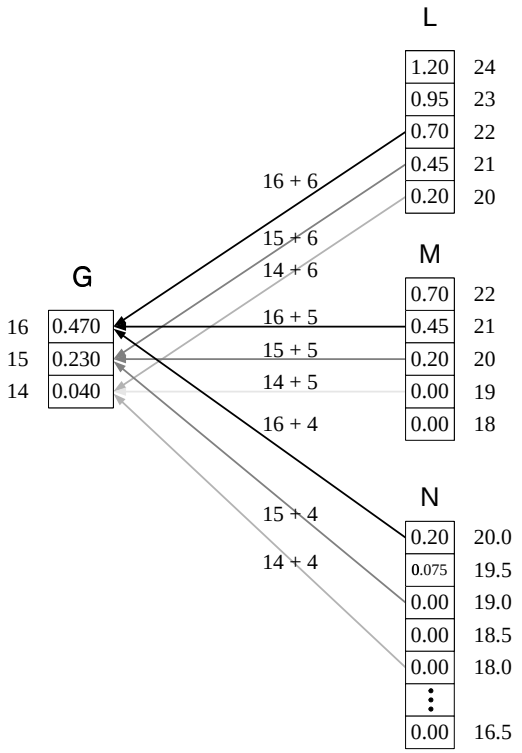


Figure 9: **A Backward-Induction Step.** Nodes G, L, M, and N are from Figure 7. The number next to a cell denotes the state (the price sum) that the cell corresponds to. A number inside the cell denotes the option value corresponding to the cell's price sum. The underlying asset's value for terminal nodes L, M, and N are 6, 5, and 4, respectively. Recall that the parameters are $S_0 = 5$, $r = 10\%$, $\sigma = 20\%$, $\tau = 0.75$, and $n = 3$.

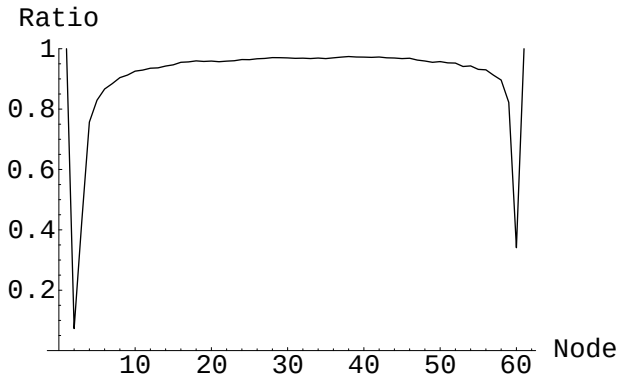


Figure 10: **Ratios of the Number of Price Sums to the Number of States.** The parameters are $S_0 = 100$, $r = 10\%$, $\sigma = 30\%$, $\tau = 1$, and $n = 30$. The above graph plots the ratio for each of the 61 terminal nodes, starting from the node with the highest asset price.

n	Monte Carlo		Hull-White		AMO	MR		
	Lower	Upper	linear	expo	Value	Value	Time	Delta
42	0.321	0.325	0.318*	0.318*	0.315*	0.316*	1	0.1013
53	0.322	0.326	0.322*	0.321*	0.318*	0.319*	2	0.1040
64	0.323	0.327	0.326	0.324	0.320*	0.321*	3	0.1057
75	0.323	0.326	0.329*	0.325	0.321*	0.323	7	0.1068
86	0.324	0.328	0.332*	0.327	0.322*	0.324	13	0.1077
97	0.325	0.329	0.335*	0.328	0.323*	0.325	23	0.1083
108	0.324	0.328	0.337*	0.329*	0.323*	0.327	39	0.1085
119	0.326	0.330	0.341*	0.330	0.323*	0.328	61	0.1081
130	0.324	0.328	0.346*	0.330*	0.325	0.327	96	0.1081
141	0.325	0.329	0.353*	0.331*	0.324*	0.326	145	0.1083

Table 2: **Monte Carlo Simulation, Hull-White, AMO, and the MR Algorithm.** The parameters are $S_0 = 50$, $X = 60$, $r = 10\%$, $\sigma = 30\%$, and $\tau = 0.5$. The Hull-White algorithms use $h = 0.005$ where linear denotes "linear" interpolation and "expo" denotes exponential interpolation (see Hull and White (1993)). Monte Carlo simulations are based on 2,000,000 trials. The AMO values are the results of an optimized code with $k = 50000$ (see Aingworth, Motwani, and Oldham (2000)). "Lower" and "Upper" bracket the 95% confidence interval. MR denotes the multiresolution method. The pricing values, the computational times, and the deltas computed by the MR algorithm are all listed in this table. Asterisks mark the cases where the answers are out of 95% confidential interval. The computational times are in seconds and are based on an Intel Pentium II 233MHz computer.

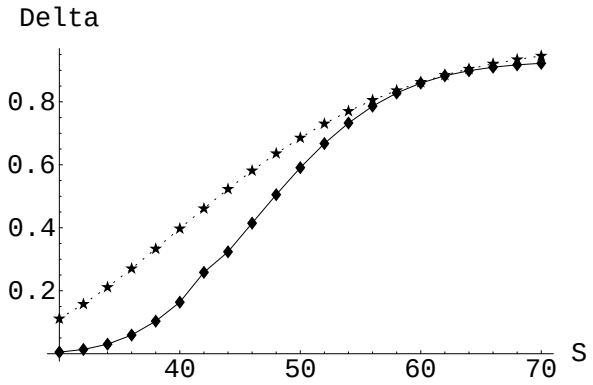


Figure 11: **Numerical delta.** The numerical delta of the Asian call with respect to the underlying asset's price as determined by the MR algorithm is plotted above. The parameters are $30 \leq S_0 \leq 70$, $X = 50$, $r = 10\%$, $\sigma = 30\%$, and $\tau = 1$. For comparison, the higher dotted curve plots the delta of the vanilla call as computed by the Black-Scholes formula with the same parameters.

Maturity (years)	Algorithm	Exercise price				
		40	45	50	55	60
0.5	HW	10.755	6.363	3.012	1.108	0.317
	MC	10.759	6.359	2.998	1.112	0.324
		(0.003)	(0.005)	(0.007)	(0.005)	(0.003)
	MR	10.754	6.356	2.997	1.104	0.317
	L	10.765	6.386	3.024	1.105	0.313
1.0	HW	11.545	7.616	4.522	2.420	1.176
	MC	11.544	7.606	4.515	2.401	1.185
		(0.006)	(0.008)	(0.010)	(0.009)	(0.007)
	MR	11.547	7.616	4.517	2.412	1.170
	L	11.576	7.662	4.557	2.431	1.172
1.5	HW	12.285	8.670	5.743	3.585	2.124
	MC	12.289	8.671	5.734	3.577	2.135
		(0.008)	(0.010)	(0.012)	(0.012)	(0.010)
	MR	12.284	8.674	5.750	3.585	2.118
	L	12.337	8.738	5.801	3.619	2.133
2.0	HW	12.953	9.582	6.792	4.633	3.057
	MC	12.943	9.569	6.786	4.639	3.055
		(0.010)	(0.013)	(0.014)	(0.015)	(0.013)
	MR	12.944	9.577	6.786	4.625	3.045
	L	13.024	9.671	6.874	4.691	3.087

Table 3: **Comparing Various Asian Option Pricing Algorithms.** The parameters are $S_0 = 50$, $r = 10\%$, and $\sigma = 30\%$. HW denotes the Hull-White algorithm based on $n = 40$ and $h = 0.005$. MC denotes Monte Carlo simulation based on $n = 40$ and 100,000 trials (the standard errors are in parenthesis). The MR algorithm uses $n = 30$. L denotes the analytic approach described in Levy (1992). The data for the Hull-White algorithm and Monte Carlo simulation are from Choa and Lee (1997).

r	σ	T	S_0	GE	Shaw	Euler	PW	TW	MC10	MC100	SE	MR
5.0%	50%	1	1.9	.195	.193	.194	.194	.195	.192	.196	.004	.193
5.0%	50%	1	2.0	.248	.246	.247	.247	.250	.245	.249	.004	.246
5.0%	50%	1	2.1	.308	.306	.307	.307	.311	.305	.309	.005	.306
2.0%	10%	1	2.0	.058	.520	.056	.0624	.0568	.0559	.0565	.0008	.0558
18.0%	30%	1	2.0	.227	.217	.219	.219	.220	.219	.220	.003	.219
12.5%	25%	2	2.0	.172	.172	.172	.172	.173	.173	.172	.003	.172
5.0%	50%	2	2.0	.351	.350	.352	.352	.359	.351	.348	.007	.351

Table 4: **Stress Tests.** The exercise price X is 2.0, and the MR algorithm uses $n = 30$. The approximation methods for comparison are: Geman-Eydeland (GE), Shaw, Euler, Post-Widder (PW), and Turnbull-Wakeman (TW). The benchmark values (MC10 and MC100) and the approximation values are from Fu, Dilip, and Wang (1998/9). MC10 uses 10 periods per day, whereas MC100 uses 100. Both are based on 100,000 trials. SE stands for standard error, also from Fu, Dilip, and Wang (1998/9).

τ	X	HW	LB	UB	MR
0.5	40	12.115	12.111	12.112	12.132
	45	7.261	7.255	7.255	7.275
	50	3.275	3.269	3.269	3.272
	55	1.152	1.148	1.148	1.147
	60	0.322	0.320	0.320	0.322
1.0	40	13.153	13.150	13.151	13.194
	45	8.551	8.546	8.547	8.576
	50	4.892	4.888	4.889	4.901
	55	2.536	2.532	2.534	2.541
	60	1.208	1.204	1.206	1.210
1.5	40	13.988	13.984	13.985	14.013
	45	9.652	9.648	9.650	9.669
	50	6.199	6.195	6.197	6.206
	55	3.771	3.767	3.770	3.786
	60	2.194	2.190	2.193	2.209
2.0	40	14.713	14.709	14.712	14.756
	45	10.623	10.620	10.623	10.659
	50	7.326	7.322	7.325	7.358
	55	4.886	4.882	4.885	4.912
	60	3.171	3.167	3.170	3.195

Table 5: **American-Style Asian Options.** HW denotes the Hull-White algorithm, while UB and LB denote the upper and lower bounds on the option values given by Chalasani et al. (1999) for the CRR model. All algorithms use $n = 40$. The other parameters are $S_0 = 50$, $r = 10\%$, and $\sigma = 30\%$.