Linear-time Option Pricing Algorithms by Combinatorics

Tian-Shyr Dai* Li-Min Liu† Yuh-Dauh Lyuu‡

Abstract

Options are popular financial derivatives that play essential roles in financial markets. How to price them efficiently and accurately is very important both in theory and practice. Options are often priced by the lattice model. Although the prices computed by the lattice converge to the theoretical option value under the continuous-time model, they may converge slowly. Worse, for some options like barrier options, the prices can even oscillate wildly. For such options, huge amounts of computational time are required to achieve acceptable accuracy. Combinatorial techniques have been used to improve the performance in pricing a wide variety of options. This paper uses vanilla options, power options, single-barrier options, double-barrier options, and lookback options as examples to show how combinatorics can help to derive linear-time pricing algorithms. These algorithms compare favorably against popular lattice methods, which take at least quadratic time.

Keywords: option pricing, lattice, combinatorics, exotic option

*Department of Information and Finance Management, National Chiao-Tung University, 1001 Ta Hsueh Road, Hsinchu, Taiwan 300, ROC. E-mail:d88006@csie.ntu.edu.tw. Tel: 886-3-2653131. Fax: 886-3-2653199. The author was supported in part by NSC grant 94-2213-E-033-024.

†Department of Applied Mathematics, Chung Yuan Christian University, 22 Pu-Jen, Pu-chung Li, Chung-Li, Tao-Yuan county, Taiwan 320. The author was supported in part by NSC grant 94-2213-E-033-031.

‡Corresponding author. Department of Finance and Department of Computer Science & Information Engineering, National Taiwan University, No. 1, Sec. 4, Roosevelt Rd., Taipei, Taiwan, 106. The author was supported in part by NSC grant 95-2213-E-002-044.

1 Introduction

Derivative securities are financial instruments whose values depend on other more elementary financial "assets," such as stocks, indexes, currencies, commodities, bonds, mortgages, other derivatives, temperatures, and countless others [1]. The elementary financial instrument in this paper is assumed to be stock for convenience. Options are financial derivatives that give their buyers the right but not the obligation to buy or sell the stock for a contractual price called the exercise price. With the rapid growth and the deregulation of financial markets, many complex options have been structured to meet specific financial goals. Although financial innovations make the market more efficient, they also give rise to new problems: How do we price these options efficiently and accurately?

As an option conveys a right, it must command a positive price called premium to avoid arbitrage opportunities. How to assign a fair price to an option given a continuous-time stochastic process for the stock price has been investigated since 1900. Finally in 1973, Black and Scholes settle the pricing problem in a way that is intellectually satisfying [2]. They also derive formulas for the simplest of options, the vanilla option. Still, an option must have a unique theoretical value, but calculating that value may be intractable [3]. Most options can not be evaluated analytically and must be priced by numerical methods. Finding efficient and accurate numerical pricing methods is thus important in both theory and practice.

The lattice method is a popular numerical method for pricing options. It is a flexible way to price options since only nominal changes are needed even when the option's payoff function is nonstandard, such as the power options (to be defined later); there may not be closed-form formulas for them. A lattice divides a certain time interval into n discrete time steps and simulates the stock price discretely at each time step. Take a 4-time-step Cox-Ross-Rubinstein (CRR) lattice in Fig. 1 as an example. (The details of the CRR lattice [4] will be described later.) The time interval between the option's initial date to maturity date is evenly divided into 4 time steps. The stock price at time step 0 is S_0. For any given node located at time step i represents a possible stock price at the i-th time step. From a node with stock price S, the CRR lattice says that the stock price after one time step equals Su (the up move) with probability p and Sd (the down move) with probability $1-p$, where $d < u$ and $ud = 1$. The node reached by j down moves and $i - j$ up moves from the

root node is denoted as $N(i, j)$ for convenience. The pricing results generated by the lattice method converge to the theoretical option value as $n \to \infty$ [5]. To calculate the option prices, the naive lattice algorithm calculates the option price for each node of the lattice, working backward in time. The time complexity of such an algorithm is at least $\sim n^2/2$ since there are $(n + 1)(n + 2)/2$ nodes on a lattice. This paradigm requires only nominal changes for different payoff functions. Of course, more time is required for complex options because more states are required per node. In fact, Chalasani et al. show that an option can be so defined that its pricing problem is #P-hard [3].

The number of time steps should be large when pricing some options due to slow convergence. For example, the prices for barrier options oscillate wildly under the lattice method [6]. Figure 2 shows the oscillation phenomenon in pricing a double-barrier option on the CRR lattice. The oscillation remains significant even as n reaches 4,000. As a large n may be required to achieve acceptable accuracy, an efficient lattice algorithm that can handle large n is important. This paper uses vanilla options, power options, single-barrier options, double-barrier options, and lookback options to demonstrate how to improve upon lattice algorithms with combinatorial tools.

We now briefly survey results for the above-mentioned options. A vanilla option gives its owner the right to buy or sell stock for the exercise price and does not have other unusual features. Black and Scholes derive an analytical solution for vanilla options. A power option is basically a vanilla option with a nonstandard payoff function. Certain power options do not have analytical formulas for their prices and thus must be priced by numerical methods. A barrier option is an option whose payoff depends on whether the stock's price path ever hits certain price levels called barriers. A single-barrier option contains only one barrier, whereas a double-barrier option contains two barriers. When the payoff functions of single-barrier options follow certain forms, analytical formulas are available [7, 8]. Although double-barrier options have been extensively studied [9, 10, 11, 12], no simple closed-form formula is available. There exists a formula that expresses the price as an infinite series of cumulative normal distributions. Although the truncation of this infinite series is necessary numerically, it can lead to large pricing errors [11]. The payoff of a lookback option depends on the extreme stock price achieved during a certain period of time and the stock price at maturity. Analytical formulas are available for lookback options with a standard payoff

[13].

The above-mentioned options can all be priced by the lattice method. However, the prices of the lattice method may converge slowly or even oscillate wildly. The oscillation phenomenon for pricing vanilla options by the lattice model has been studied by Omberg [14]. Boyle and Lau suggest picking proper n's to reduce the oscillation for single-barrier options [6]. Alternatively, Ritchken provides a novel trinomial lattice model for pricing both single- and double-barrier options [15]. But his approach is costly when the barrier is very close to the initial stock price S_0 (it is called the "barrier-too-close" problem). Figlewski and Gao propose the adaptive mesh model (AMM hereafter) for pricing vanilla options and single-barrier options [16]. Their AMM solves the "barrier-too-close" problem of pricing single-barrier options. However, no efforts have been made to extend the AMM to price double-barrier options. Although the above-mentioned approaches can partially alleviate the price oscillation problem, they are not efficient as they all run in $O(n^2)$ time. An efficient lattice pricing algorithm for the lookback option is equally important since the prices converge slowly as illustrated in Fig. 3. An $O(n^3)$-time lattice algorithm is first suggested by Hull and White [17]. The performance is reduced to $O(n^2)$ by Hull [1]. But no linear-time algorithm is available before this paper.

This paper uses combinatorics to improve the performance in pricing a wide variety of options. Specifically, such combinatorial tools as recurrence relations, the reflection principle, and the inclusion-exclusion principle are applied to derive linear-time lattice algorithms for pricing all the options mentioned in the last paragraph. These new algorithms are therefore at least an order faster than existing lattice algorithms.

The combinatorial approach seems to be first emphasized by Lyuu [18, 19]. He shows that vanilla options can be priced in linear time by taking advantage of a recurrence relation (to be made more explicit later) [19]. Since power options can be priced by the lattice method for vanilla options after only nominal changes, a linear-time combinatorial algorithm for power options is easily available as well. Lyuu also develops a linear-time combinatorial algorithm for pricing single-barrier options by taking advantage of the reflection principle [18]. The reflection principle efficiently counts the number of paths from the root node of the lattice to any node at maturity while hitting a specific price level (the barrier) in the process. This property is useful for pricing options with one or two barriers. In our

combinatorial pricing algorithm for double-barrier options, the reflection principle is applied repeatedly to count the number of paths that hit either of the barriers. To prevent counting the paths more than once as some paths may hit both barriers, the inclusion-exclusion principle becomes necessary. A proof is given to show that our algorithm runs in $O(n)$ time.

This paper develops a linear-time combinatorial algorithm for pricing lookback options. The payoff of a lookback option depends on the stock price at maturity and the extreme stock price from time step 0 to time step n. Thus a way is needed to efficiently count the number of paths reaching each node at maturity, say $N(4,2)$ in Fig. 1, with an extreme historical stock price, say S_0d (the minimum price). Such paths hit the price level S_0d but not S_0d^2. Thus the desired number can be computed by applying the reflection principle twice, once with S_0d as the barrier and once with S_0d^2 as the barrier. We divide all paths into groups based on the combination of the stock price at maturity (such as node $N(4,2)$'s stock price above) and the extreme stock price (such as S_0d above). The number of price paths in each group can be efficiently counted by the method mentioned above. A recurrence relation that sums the value contributed by each group is then developed to obtain the result. This pricing algorithm runs in $O(n)$ time.

Our combinatorial algorithms can be applied to speed up various lattice models. Dai and Lyuu provide a novel lattice model, the bino-trinomial tree (BTT hereafter), that solves the oscillation problem for a wide variety of options [20]. Numerical results confirm that the BTT converges more smoothly and faster than other lattice approaches. The BTT is mainly composed of a CRR lattice. Thus our pricing algorithm can form its backbone.

Our paper is organized as follows. Background knowledge, like the assumption of the stock price process, the definitions of the options mentioned in this paper, the method to price an option under the risk-neutral probability, and the required combinatorial techniques, are introduced in section 2. How to price vanilla options, power options, and single-barrier options with the above-mentioned technique is also shown in this section. A linear-time algorithm for pricing double-barrier options is derived in section 3. In section 4, we will develop $O(n)$-time combinatorial pricing algorithms for lookback options. Numerical results in section 5 illustrates how the combinatorial techniques improve upon computational efficiency with double-barrier options and lookback options as examples.

Section 6 concludes the paper.

2 Preliminaries

2.1 Financial Background

Stock Price Process

For convenience, assume options initiate at time 0 and mature at time T. Let $S(t)$ denote the stock price at time t, where $0 \leq t \leq T$. $S(t)$ follows the log-normal diffusion process:

$$S(t + dt) = S(t) \exp[(r - 0.5\sigma^2)\, dt + \sigma\, dW_t], \tag{1}$$

where W_t denotes the Wiener process, r denotes the risk-free interest rate per annum, and σ denotes the volatility of the stock price.

The continuous-time log-normal stock price process can be discretized into a lattice model, which is basically a random walk. The structure of the CRR lattice is described below.

CRR Lattice

A CRR lattice model divides a certain time interval from time 0 to time T into n equal time steps and specifies the stock price discretely at each time step. The CRR lattice converges to the stock price process Eq. (1) if the first and second moments of the stock price process are matched at each node of the CRR lattice [5]. Consider the CRR lattice illustrated in Fig. 1. To match the first two moments, the model sets

$$\begin{aligned} u &\equiv e^{\sigma\sqrt{T/n}}, \\ d &\equiv e^{-\sigma\sqrt{T/n}}. \end{aligned}$$

Note that $ud = 1$. For pricing purpose, the probability p is set to $(e^{rT/n} - d)/(u - d)$. The stock price S resulting from j down moves and $i - j$ up moves from time step 0 equals $S_0 u^{i-j} d^j$ with probability $\binom{i}{j} p^{i-j}(1-p)^j$. This node is at time step i and is denoted as $N(i, j)$. For convenience, we use "terminal nodes" to refer to those nodes at the n-th time step where payoff occurs.

Vanilla Option

A vanilla option gives the holder the right to buy or sell the stock for price X defined in the option contract at the maturity date. A call option allows the option owner to buy the stock for X dollars at time T, while a put option allows the option owner to sell the stock for X at time T. Specifically, the payoff of a vanilla option at time T can be expressed as

$$\max(\theta S(T) - \theta X, 0), \tag{2}$$

where θ equals 1 for call options and -1 for put options.

Power Option

Two possible payoffs for power options have been suggested in the literature:

$$\max(\theta S(T)^p - \theta X, 0), \tag{3}$$

and

$$\max\left((\theta S(T) - \theta X)^p, 0\right), \tag{4}$$

where p denotes a positive constant, and θ can be 1 (for call options) or -1 (for put options).

Single-Barrier Option

A barrier option is an option whose payoff depends on whether the stock's price path ever hits certain price levels called barriers. A single-barrier option is a barrier option with only one barrier H. Assume $H > S(0)$ for convenience. Define $S_{\sup} \equiv \sup_{0 \le t \le T} S(t)$. The payoff of a single-barrier option at maturity date T is

$$\text{payoff} = \begin{cases} \max(\theta S(T) - \theta X, 0), & \text{if } S_{\sup} \ge H, \\ 0, & \text{otherwise.} \end{cases}$$

Double-Barrier Option

A double-barrier option is a barrier option with two barriers L and H. Assume that $L < S(0) < H$ for convenience and define $S_{\inf} \equiv \inf_{0 \le t \le T} S(t)$. The payoff of a double-barrier option at the maturity date is

$$\text{payoff} = \begin{cases} \max(\theta S(T) - \theta X, 0) & \text{if } S_{\sup} \ge H \text{ or } S_{\inf} \le L, \\ 0 & \text{otherwise.} \end{cases} \tag{5}$$

Lookback Option

The payoff of a lookback option depends on the maximum or minimum stock price during the life of the option. The payoff function for a lookback option at the maturity date is

$$\text{payoff} = \begin{cases} S(T) - S_{\inf} & \text{for a call option,} \\ S_{\sup - S(T)} & \text{for a put option.} \end{cases} \tag{6}$$

Pricing Options on the CRR Lattice

The theoretical value of an option equals the discounted expected payoff of the option:

$$e^{-rT}\mathrm{E}\left(\text{payoff}\right). \tag{7}$$

The dynamic-programming technique is usually applied to price the option on the CRR lattice by evaluating the option value for each node from time step n to time step 0. Take a vanilla call option as an example. Define the option value for node $N(i,j)$ as $V(i,j)$. Thus $V(n,j)$ is equal to the payoff at terminal node $N(n,j)$, i.e., $\max(S_0u^{n-j}d^j - X, 0)$. $V(i,j)$, $0 \le j \le i < n$, is calculated by

$$V(i,j) \equiv e^{-rT/n} \times \left(pV(i+1,j) + (1-p)V(i+1,j+1)\right). \tag{8}$$

The desired price is $V(0,0)$. This naive pricing method takes $O(n^2)$ time as there are $\sim n^2/2$ nodes on an n-time-step CRR lattice. If the option is changed to the power call option, we only need to change the payoff at terminal node $N(n,j)$ from $\max(S_0u^{n-j}d^j - X, 0)$ to $\max\left(\left(S_0u^{n-j}d^j\right)^p - X, 0\right)$ by Eq. (3) or $\max\left(\left(S_0u^{n-j}d^j - X\right)^p, 0\right)$ by Eq. (4) for a positive constant p.

2.2 Combinatorial Tools

This subsection introduces the combinatorial tools useful for this paper.

Recurrence Relations

For a sequence $\{f_n\}$, a recurrence relation defines a mathematical relationship that expresses f_n as some combination of $f_0, f_1, \ldots, f_{n-1}$. Take a vanilla call option for example. Since

the probability to reach node $N(n,j)$ is $\binom{n}{j}p^{n-j}(1-p)^j$, the price of the n-time-step CRR lattice can be derived directly from Eq. (7) as

$$e^{-rT}\sum_{j=0}^{n}\binom{n}{j}p^{n-j}(1-p)^j\max(S_0u^{n-j}d^j-X,0). \tag{9}$$

This formula can be evaluated in $O(n)$ time via recurrence relations. Define C_j and D_j as probabilities $\binom{n}{j}p^{n-j}(1-p)^j$ and terminal stock prices $S_0u^{n-j}d^j$, respectively. C_j and D_j can each be expressed by recurrence relations:

$$C_j = C_{j-1}\frac{(1-p)(n-j+1)}{pj},$$
$$D_j = D_{j-1}\frac{d}{u}.$$

Both C_j and D_j can be evaluated from C_{j-1} and D_{j-1} with only O(1) arithmetic operations. We start with $C_0 = p^n$ and $D_0 = S_0u^n$ and then evaluate each summand in Eq. (9) sequentially in constant time by the recurrence relations above. Equation (9) can thus be evaluated in $O(n)$ time as there are $n+1$ summands.

A power option can also be priced in $O(n)$ time by simply replacing the payoff of a vanilla option, Eq. (2), with the payoff of a power call option, Eq. (3) or (4). To be more precise, the value of a power call option is obtained by replacing $\max(S_0u^{n-j}d^j-X,0)$ in Eq. (9) with $\max\left(\left(S_0u^{n-j}d^j\right)^p-X,0\right)$ or $\max\left((S_0u^{n-j}d^j-X)^p,0\right)$ to obtain, respectively:

$$e^{-rT}\sum_{j=0}^{n}\binom{n}{j}p^{n-j}(1-p)^j\max\left(\left(S_0u^{n-j}d^j\right)^p-X,0\right), \tag{10}$$
$$e^{-rT}\sum_{j=0}^{n}\binom{n}{j}p^{n-j}(1-p)^j\max\left((S_0u^{n-j}d^j-X)^p,0\right).$$

The Reflection Principle

The reflection principle can help us efficiently count the number of paths that hit a specific price level before reaching a certain node at maturity in a CRR lattice. This property is essential for pricing barrier-like options. We now derive a useful combinatorial formula with the help of the grid in Fig. 4. This grid reflects the structure of a CRR lattice: The x-coordinate denotes the time step of the CRR lattice, and the y-coordinate denotes the stock price level. Each step on the grid from vertex (i,j) can either go to vertex $(i+1,j+1)$ (the up move) or vertex $(i+1,j-1)$ (the down move). The question is, how many paths

from node A ($(0, -a)$) that end at node B ($(n, -b)$) will hit barrier H? Without loss of generality, assume that $a, b \geq 0$.

Consider one such path, $\widehat{AJB}$, that hits barrier H at node J for the first time. We can reflect the initial path $\widehat{AJ}$ with respect to the H-axis to get $\widehat{A_1J}$ (the dashed curve). Each path from node A to node J maps to a unique path from node A_1 to node J, and vice versa. Thus the number of paths from node A to node J equals the number of paths from node A_1 to node J. As a result, the desired number of paths moving from node A to node B and hitting barrier H equals the number of paths from node A_1 to node B. This is the celebrated reflection principle [21].

Assume that x up moves and y down moves are required to go from node A_1 to node B. Thus $x + y = n$ and $x - y = -a - b$. These two equations give $x = \frac{n-a-b}{2}$ and $y = \frac{n+a+b}{2}$. Thus the number of paths that hit H before arriving at B is

$$\binom{n}{\frac{n-a-b}{2}} \quad \text{for even, non-negative } n - a - b \tag{11}$$

and zero otherwise.

Lyuu derives an $O(n)$-time combinatorial algorithm for pricing a single-barrier option with barrier H by taking advantage of Eq. (11) [18]. Let the barrier H be $S_0 u^{n-2h}$. We place a CRR lattice on a grid so barrier H coincides with the x-axis of the grid as illustrated in Fig. 5. The option value is obtained by taking the discounted expected payoff of the option at maturity as in Eq. (7). This is done by accumulating the values contributed by the terminal nodes. Two kinds of terminal nodes are considered. First, for a terminal node that is above barrier H (inclusive) like node A, all the paths that reach node A must hit barrier H. The option value contributed by these terminal nodes is

$$\sum_{i=0}^{h} \binom{n}{i} p^{n-i} (1-p)^i e^{-rT} \max(S_0 u^{n-i} d^i - X, 0). \tag{12}$$

Second, for a terminal node that is below barrier H (exclusive) like node B, the number of paths that hit barrier H before reaching B can be efficiently computed by Eq. (11). The option value contributed by these terminal nodes is

$$\sum_{i=h+1}^{n} \binom{n}{2h-i} p^{n-i} (1-p)^i e^{-rT} \max(S_0 u^{n-i} d^i - X, 0). \tag{13}$$

The price of the single-barrier option is (12) + (13).

The Inclusion-Exclusion Principle

The inclusion-exclusion principle states that if $A_1, \ldots, A_n$ are finite sets, then

$$\left|\bigcup_{i=1}^{n} A_i\right| = \sum_{i=1}^{n} |A_i| - \sum_{i<j} \left|A_i \bigcap A_j\right| + \sum_{i<j<k} \left|A_i \bigcap A_j \bigcap A_k\right| - \ldots + (-1)^{n-1} \left|\bigcap_{i=1}^{n} A_i\right|,$$

where $|A|$ denotes the cardinality of set A [21]. This principle will help us efficiently count the number of paths that hit either barrier L or H before reaching a certain terminal node. The details will be given in the next section.

3 An $O(n)$-Time Algorithm on a CRR Lattice for Double-Barrier Options

This section derives a useful combinatorial formula with the combination of the reflection principle and the inclusion-exclusion principle to build up a pricing algorithm for double-barrier options.

3.1 A Combinatorial Formula

Our goal here is a combinatorial formula that counts the number of paths that hit one of two given price levels (L and H) before reaching a certain node at the n-th time step on a CRR lattice. This formula will be instrumental in deriving an $O(n)$-time algorithm for pricing double-barrier options. Consider the grid in Fig. 6. How many price paths from node A $(0, -a)$ to node B $(n, -b)$ will hit either barrier L or barrier H?

Before solving this problem, a simplified problem is considered first: How many price paths moving from node A to node B will hit barrier H before one hit of barrier L? One such path may hit barrier H first at J and barrier L later at K. We can first reflect the path $\widehat{AJ}$ with respect to the H-axis to obtain path $\widehat{A_1J}$. The reflection principle says that the number of paths moving from node A to node B while hitting barrier H equals the number of paths moving from node A_1 to node B. The reflection principle can be applied repeatedly. The curve $\widehat{A_1K}$ can be reflected with respect to the L-axis to obtain $\widehat{A_2K}$. By the reflection principle again, the number of paths moving from node A_1 to node B while hitting barrier L equals the number of paths from node A_2 to node B. Thus the number of

paths moving from node A to node B while hitting barrier H at least once before one hit of barrier L equals the number of paths from A_2 to B.

Assume that x up moves and y down moves are required to move from node A_2 (with coordinate $(0, -(a+2s))$) to node B (with coordinate $(n, -b)$). Thus we have $x + y = n$ and $x - y = a - b + 2s$, which yield $x = (n + a - b + 2s)/2$. So the answer to our simplified problem is

$$\binom{n}{\frac{n+a-b+2s}{2}} \quad \text{for even, non-negative } n + a - b. \tag{14}$$

Note that a path counted by Eq. (14) may hit L first before hitting H. The point is that among the hits, there must exist one hit of H that appears before one hit of L for the path to be counted.

The problem of counting the number of paths that will hit either barrier L or barrier H before arriving at node B is now within reach. It is useful to consider a function f that maps a path to a string. Each string contains information about the barrier hitting sequence. For example, $f(\widehat{AB}) = HHL$ since the path $\widehat{AB}$ hits the barrier H twice before hitting the barrier L. Next, we define α_i as the set of paths whose f value *contains* $\overbrace{H^+L^+H^+\cdots}^{i}$ with $i \geq 1$. L^+ and H^+ denote a sequence of Ls and Hs, respectively. Obviously, the path $\widehat{AB}$ belongs to both set α_1 and set α_2. Similarly, define β_i as the set of paths whose f value contains $\overbrace{L^+H^+L^+\cdots}^{i}$ with $i \geq 1$. Thus the path $\widehat{AB}$ belongs to set β_1. The number of elements in sets α_i and β_i can be calculated by repeatedly using the reflection principle as mentioned above. The number of elements in each set is found to be:

$$|\,\alpha_i\,| = \begin{cases} \binom{n}{\frac{n+a+b+(i-1)\,s}{2}} & \text{for odd } i \\ \binom{n}{\frac{n+a-b+is}{2}} & \text{for even } i \end{cases} \qquad |\,\beta_i\,| = \begin{cases} \binom{n}{\frac{n-a-b+(i+1)\,s}{2}} & \text{for odd } i \\ \binom{n}{\frac{n-a+b+is}{2}} & \text{for even } i \end{cases} \tag{15}$$

Note that each path that hits the barrier may belong to more than one set. For example, $\widehat{AB}$ in Fig. 6 belongs to sets α_1, α_2, and β_1. Finally, the inclusion-exclusion principle is used to calculate the number of paths moving from A to B while hitting either barrier L or H as follows:

$$\mathbf{N}(a, b, s) \equiv \sum_{i=1}^{\lceil \frac{n}{s} \rceil} (-1)^{i+1} (|\alpha_i| + |\beta_i|). \tag{16}$$

3.2 The $O(n)$-Time Pricing Algorithm

The construction of the pricing algorithm can be divided into several cases. We first consider the case $X \geq H$. If the payoff $\max(S(T) - X, 0) > 0$, we must have $S_{\sup} \geq S(T) > X \geq H$, meaning the barrier H must be hit. Thus the value of the double-barrier call option is

$$\begin{aligned}
& e^{-rT}\mathrm{E}(\text{Payoff}) \\
= \; & e^{-rT}\mathrm{E}\left(\max(S(T) - X, 0)1_{\{S_{\sup} \geq H \text{ or } S_{\inf} \leq L\}}\right) \\
= \; & 1_{\{S(T)>X\}}e^{-rT}\mathrm{E}\left(\max(S(T) - X, 0)1_{\{S_{\sup} \geq H \text{ or } S_{\inf} \leq L\}}\right) \\
& +1_{\{S(T)\leq X\}}e^{-rT}\mathrm{E}\left(\max(S(T) - X, 0)1_{\{S_{\sup} \geq H \text{ or } S_{\inf} \leq L\}}\right) \\
= \; & 1_{\{S(T)>X\}}e^{-rT}\mathrm{E}\left(\max(S(T) - X, 0)\right) + 0 \\
= \; & e^{-rT}\mathrm{E}(\max(S(T) - X, 0)) \\
= \; & \text{ value of an otherwise identical vanilla call option.}
\end{aligned}$$

As this has been priced by Eq. (9), we focus on $X < H$ from now on.

We proceed to develop efficient pricing algorithms for other, non-degenerate cases. We first place the CRR lattice on a grid as displayed in Fig. 7. The barriers L and H equal $S_0u^{n-h}d^h = S_0u^{n-2h}$ and $S_0u^{n-l}d^l = S_0u^{n-2l}$, respectively. There is also a unique integer a such that $S_0u^{n-a}d^a \leq X < S_0u^{n-a+1}d^{a-1}$.

Next we analyze the option value contributed by a price path that reaches terminal node $N(n, j)$. The probability for this price path is $p^{n-j}(1-p)^j$. The payoff at node $N(n, j)$ is $\max(S_0u^{n-j}d^j - X, 0)$. Thus the value contributed by this price path is

$$p(j) \equiv e^{-rT}p^{n-j}(1-p)^j \max(S_0u^{n-j}d^j - X, 0) \tag{17}$$

if this price path hits either barrier L or H. Furthermore, the number of price paths that reach node $N(n, j)$ is $\binom{n}{j}$. If node $N(n, j)$ is above the barrier H (inclusive) or below the barrier L (inclusive), the value contributed by this node is $\binom{n}{j}p(j)$. This is because all the price paths that reach this node must also hit barrier L or H.

The combinatorial algorithm for pricing the double-barrier call option is now within reach. First, a table is built for storing the values $\binom{n}{k}$, $0 \leq k \leq n$. Thereafter, $\binom{n}{k}$ can be evaluated in constant time by a simple table lookup. Next, two non-degenerate cases are considered as follows.

Case 1. $L < X < H$:

The option value can be decomposed into two parts: (1) the value contributed by the terminal nodes between X and H (exclusive), and (2) the value contributed by the terminal nodes above H (inclusive).

The first part of the option value is the sum of the values contributed by the terminal nodes, $N(n, j)$, $h < j < a$, between X and H. The number of paths that hit one of the barriers before reaching node $N(n, j)$ is $\mathbf{N}(n - 2h, 2j - 2h, 2l - 2h)$. The value contributed by such a path is $p(j)$ by Eq. (17). Therefore, the value contributed by node $N(n, j)$ is $\mathbf{N}(n - 2h, 2j - 2h, 2l - 2h)p(j)$. The desired sum is therefore

$$\sum_{j=h+1}^{a-1} \mathbf{N}(n - 2h, 2j - 2h, 2l - 2h)p(j). \tag{18}$$

The second part of the option value is the sum of the values contributed by the terminal nodes $N(n, i)$, $0 \le i \le h$, above the barrier H (inclusive). The value contributed by node $N(n, i)$ is $\binom{n}{i}p(i)$. Therefore, the second part of the option value is

$$\sum_{i=0}^{h} \binom{n}{i} p(i) = e^{-rT} \sum_{i=0}^{h} \binom{n}{i} p^{n-i}(1-p)^i \max(S_0 u^{n-i} d^i - X, 0). \tag{19}$$

The value of a double-barrier call option is thus (18)+(19).

Case 2. $X \le L$:

The option value consists of three distinct parts: (1) the terminal nodes between L (exclusive) and H (exclusive), (2) the terminal nodes above H (inclusive), and (3) the terminal nodes between L (inclusive) and X.

The first part of the option value is the sum of the values contributed by the terminal nodes between L and H:

$$\sum_{j=h+1}^{l-1} \mathbf{N}(n - 2h, 2j - 2h, 2l - 2h)p(j). \tag{20}$$

The second part of the option value is the sum of the values contributed by the terminal nodes above the barrier H (inclusive) and equals Eq. (19). The third part of the option value is the sum of the values contributed by the terminal nodes, says $N(n, k)$, $l \le k < a$, between L (inclusive) and X. This part of the option value is

$$\sum_{k=l}^{a} \binom{n}{k} p(k) = e^{-rT} \sum_{k=l}^{a} \binom{n}{k} p^{n-k}(1-p)^k \max(S_0 u^{n-k} d^k - X, 0). \tag{21}$$

Thus the value of a double-barrier call option is (19)+(20)+(21).

3.3 Time Complexity

We next prove that our algorithm runs in $O(n)$ time. Note that the degenerate case can be priced in $O(n)$ time by Eq. (9). So we focus on non-degenerate cases.

Our pricing algorithm can be divided into three parts. The first part denotes the construction of the table for $\binom{n}{k}$, $0 \le k \le n$, the second part denotes the evaluation of Eqs. (19) and (21), and the last part denotes the evaluation of Eq. (18) in case 1 or Eq. (20) in case 2. The linear-time complexity of our algorithm is established by showing that each part mentioned above can be computed in $O(n)$ time.

The first part of our algorithm could be computed in $O(n)$ time by the recurrence equation $\binom{n}{k} = \binom{n}{k-1} \times (n-k+1)/k$. Equations (19) and (21) can also be calculated in $O(n)$ time by the recurrence relations used to calculate Eq. (9). Finally, we proceed to show that both Eqs. (18) and (20) can be calculated in $O(n)$ time. As the terms $|\alpha_i|$ and $|\beta_i|$ defined in Eq. (15) can be represented in $\binom{n}{k}$ form, $|\alpha_i|+|\beta_i|$ can be evaluated in constant time by looking up the table storing $\binom{n}{k}$. Thus $\mathbf{N}(a, b, s)$ in Eq. (16) can be solved in $O(\lceil \frac{n}{s} \rceil)$ time. In both Eqs. (18) and (20), $\mathbf{N}(n-2h, 2j-2h, 2l-2h)$ can be evaluated in less than $l-h$ steps.[1] Consequently, it takes $O(\frac{n}{2(l-h)}(l-h)) = O(n)$ time to evaluate Eqs. (18) and (20). Thus the option value (18)+(19) in case 1 or (19)+(20)+(21) in case 2 can each be calculated in $O(n)$ time and our algorithm runs in $O(n)$ time.

4 An $O(n)$-Time Algorithm for Lookback Options

A lookback option can also be priced by summing the values contributed by all the terminal nodes. The major hurdle is that all the price paths that reach a terminal node, $N(n, i)$, do not have the same payoff, depending on their extreme stock prices during the life of the option (recall Eq. (6)). To derive the value contributed by $N(n, i)$, we divide the price paths reaching $N(n, i)$ into groups by their extreme stock prices. Clearly, the price paths in the same group have the same payoff, thus contributing equally to the option value. The number of price paths in each group is then found by the reflection principle. The value contributed by a group is the value contributed by a price path in that group times the number of price paths in that group. Finally, a terminal node $N(n, i)$'s contribution to the

[1] Note that $a < l$ in Eq. (18) since $L < X$ in case 1.

option value equals the sum of the values contributed by the groups.

It turns out that the values contributions by adjacent terminal nodes are linked by simple recurrence relations. The option value can be evaluated in linear time by taking advantage of these relations. To keep the analysis simple, we focus on the pricing of lookback call options. The number of time steps of the CRR lattice n is furthermore assumed to be an even number. The extension to the lookback put options and odd numbers of time steps is straightforward.

We now divide the price paths reaching $N(n,i)$ into groups by their extreme stock prices. Recall that the stock price of the terminal node $N(n,i)$ is S_0u^{n-2i}. For that node, and that node only, $x = n - i$ up moves and $y = i$ down moves are taken to reach it as $x - y = n - 2i$. Let $\wp$ denote the set of price paths that end at $N(n,i)$. Clearly, $|\wp| = \binom{n}{i}$. Define $S_{\inf}(o)$ as the minimum stock price during the time interval $[0,T]$ of price path o. Then $\min_{o\in\wp} S_{\inf}(o) = S_0d^i = S_0u^{-i}$, which is achieved by the price path that takes i consecutive down moves and then $n - i$ consecutive up moves to reach $N(n,i)$. We also have $\max_{o\in\wp} S_{\inf}(o) = \min(S_0u^{n-2i}, S_0)$. This value is achieved by the price path that takes $n - i$ consecutive up moves and then i consecutive down moves to reach $N(n,i)$. Thus $\wp$ can be divided into groups with extreme stock prices

$$S_0u^{-i}, S_0u^{-i+1}, \ldots, \min(S_0u^{n-2i}, S_0).$$

To evaluate the value contributed by $N(n,i)$, two different cases are considered as follows:

Case 1. $i \le n/2$:

In this case, the maximum among the minimum stock prices $\max_{o\in\wp} S_{\inf}(o)$ equals $\min(S_0u^{n-2i}, S_0) = S_0$. Let us start with the group of price paths with minimum stock price S_0u^{-i}; price paths in this group must hit the price level S_0u^{-i}. The cardinality of this group can be computed by applying Eq. (11) to get $\binom{n}{0}$=1. The contribution of this group to the option value is

$$e^{-rT}p^{n-i}(1-p)^i(S_0u^{n-2i} - S_0u^{-i}).$$

Next consider the group of price paths with minimum stock price S_0u^{-i+1}. Price paths of this group must hit the price level S_0u^{-i+1}, but not S_0u^{-i}. The cardinality of this group can be computed by applying Eq. (11) twice, once with S_0u^{-i+1} as the barrier and once

with S_0u^{-i} as the barrier, to get $\binom{n}{1} - \binom{n}{0}$. Thus the value contributed by this group is

$$e^{-rT}p^{n-i}(1-p)^i\left(\binom{n}{1} - \binom{n}{0}\right)(S_0u^{n-2i} - S_0u^{-i+1}).$$

In general, the value contributed by a group with minimum stock price S_0u^{-i+j} as follows:

$$e^{-rT}p^{n-i}(1-p)^i\left(\binom{n}{j} - \binom{n}{j-1}\right)(S_0u^{n-2i} - S_0u^{-i+j}),$$

where $j \leq i$. Thus the value contributed by the terminal node $N(n,i)$ is the sum of the above values:

$$\begin{aligned} & e^{-rT}p^{n-i}(1-p)^i\binom{n}{0}(S_0u^{n-2i} - S_0u^{-i}) \\ + \quad & e^{-rT}\sum_{j=1}^{i} p^{n-i}(1-p)^i\left(\binom{n}{j} - \binom{n}{j-1}\right)(S_0u^{n-2i} - S_0u^{-i+j}). \end{aligned}$$

The above formula can be rewritten by separating the two terms making up the payoff function:

$$\begin{aligned} & e^{-rT}p^{n-i}(1-p)^i\left[\binom{n}{0}S_0u^{n-2i} + \sum_{j=1}^{i}\left(\binom{n}{j} - \binom{n}{j-1}\right)S_0u^{n-2i}\right] \\ - \quad & e^{-rT}p^{n-i}(1-p)^i\left\{\binom{n}{0}S_0u^{-i} + \sum_{j=1}^{i}\left[\left(\binom{n}{j} - \binom{n}{j-1}\right)S_0u^{-i+j}\right]\right\}. \end{aligned}$$

Denote the first term and the second term as $F(i)$ and $G(i)$, respectively. $F(i)$ can be expressed by the following recurrence relation:

$$F(i) = F(i-1)p_xu^{-2} + e^{-rT}p^{n-i}(1-p)^i\left(\binom{n}{i} + \binom{n}{i-1}\right)S_0u^{n-2i}, \tag{22}$$

where $p_x = (1-p)/p$. $G(i)$ can be expressed by the following recurrence relation:

$$G(i) = G(i-1)p_xu^{-1} + e^{-rT}p^{n-i}(1-p)^i\left(\binom{n}{i} + \binom{n}{i-1}\right)S_0. \tag{23}$$

The option value contributed by $N(n,i)$ is $F(i) - G(i)$.

Case 2. $i > n/2$:

In this case, the maximum among the minimum stock prices $\max_{o\in\wp} S_{\inf}(o)$ equals $\min(S_0u^{n-2i}, S_0) = S_0u^{n-2i}$. Thus the value contributed by the terminal node $N(n,i)$ is the sum of the values contributed by the groups with minimum stock price $S_0u^{-i}, S_0u^{-i+1}, \ldots, S_0u^{n-2i}$

as follows:

$$\begin{aligned} & e^{-rT}p^{n-i}(1-p)^i\binom{n}{0}(S_0u^{n-2i}-S_0u^{-i}) \\ + \quad & e^{-rT}\sum_{j=1}^{n-i}p^{n-i}(1-p)^i\left(\binom{n}{j}-\binom{n}{j-1}\right)(S_0u^{n-2i}-S_0u^{-i+j}). \end{aligned}$$

Again, rewrite the above formula by separating the two terms making up the payoff function:

$$\begin{aligned} & e^{-rT}p^{n-i}(1-p)^i\left[\binom{n}{0}S_0u^{n-2i}+\sum_{j=1}^{n-i}\left(\binom{n}{j}-\binom{n}{j-1}\right)S_0u^{n-2i}\right] \\ - \quad & e^{-rT}p^{n-i}(1-p)^i\left\{\binom{n}{0}S_0u^{-i}+\sum_{j=1}^{n-i}\left[\left(\binom{n}{j}-\binom{n}{j-1}\right)S_0u^{-i+j}\right]\right\}. \end{aligned}$$

Denote the first term and the second term as $F(i)$ and $G(i)$, respectively. Then the series $F(i)$ and $G(i)$ will satisfy the following two recurrence relations:

$$F(i)=F(i-1)p_x/u^2-e^{-rT}p^{n-i}(1-p)^i\left(\binom{n}{n-i+1}-\binom{n}{n-i}\right)S_0u^{n-2i}, \tag{24}$$

$$G(i)=G(i-1)p_x/u-e^{-rT}p^{n-i}(1-p)^i\left(\binom{n}{n-i+1}-\binom{n}{n-i}\right)S_0d^{2i-n-1}. \tag{25}$$

The $O(n)$-time pricing algorithm for pricing lookback options is now within reach. The value of a lookback option is the sum of the values contributed by all the terminal nodes,

$$\sum_{i=0}^{n}\left(F(i)-G(i)\right). \tag{26}$$

The above formula can be evaluated in $O(n)$ time if each $F(i)-G(i)$ can be evaluated in constant time. Toward that end, two tables are constructed. The first table stores $\binom{n}{i}$, $0\le i\le n$. The second table stores $p^i(1-p)^{n-i}$, $0\le i\le n$. Both tables can be filled in $O(n)$ time by the recurrence relations $\binom{n}{k}=\binom{n}{k-1}\times(n-k+1)/k$ and $p^i(1-p)^{n-i}=p^{i-1}(1-p)^{n-i+1}\times p/(1-p)$, respectively. Note that for any arbitrary integer i between 0 and n, both $\binom{n}{i}$ and $p^i(1-p)^{n-i}$ can be obtained in constant time by looking up the two tables mentioned above. $F(i)$ and $G(i)$ can now be calculated in constant time by Eqs. (22)–(25) given $F(i-1)$ and $G(i-1)$. We conclude that Eq. (26) can be calculated in $O(n)$ time.

5 Experimental Results

A power option can be viewed as a vanilla option with a nonstandard payoff function. While there is no analytical formula for a power option with payoff function Eq. (4), the power option can be priced by the lattice method with only nominal changes. However, the lattice method suffers from slow convergence and a large n is required to obtain accurate results. Figure 8 compares the $O(n)$-time combinatorial approach (see Eq. (10)) and the $O(n^2)$-time dynamic-programming approach (see Eq. (8)). The x- and the y-axes denote the running time and the pricing result, respectively. For example, it costs 0.0157 second to compute with a 400-time-step CRR lattice by the dynamic-programming approach and obtain the price 2.6589 (point A). It takes almost the same time to compute with a 1600-time-step CRR lattice by the combinatorial approach and obtain the price 2.6669 (point B). The combinatorial approach obviously converges faster in terms of CPU time.

Pricing barrier options on a CRR lattice will result in significant oscillations (recall Fig. 2). To alleviate them, Ritchken proposes a novel trinomial lattice model [15], but his model is not quite efficient since it runs in $O(n^2)$ time. Alternatively, Dai and Lyuu propose a novel trinomial lattice, the BTT (bino-trinomial tree) model [20]. A CRR lattice comprises the bulk of the BTT. Thus pricing barrier options on the BTT can be done in $O(n)$ time by taking advantage of the combinatorial algorithm in this paper. Numerical results for pricing a double-barrier option are illustrated in Fig. 9. The x- and y-axes denote the running time and the option price, respectively. For example, it costs 0.01538 second to compute with a 75-time-step Ritchken's trinomial lattice to obtain the price 10.20447 (point A). It costs almost the same time to compute with a 800-time-step BTT to obtain the price 10.20137 (point B). The accurate value obtained by running a 20,000-time-step BTT is about 10.1993. It can be observed that the BTT converges more smoothly and faster than Ritchken's trinomial lattice model.

Pricing lookback options on a lattice model suffers from slow convergence, which can be observed in Fig. 3. Finding an algorithm that can handle a large n is thus important. Hull suggests an $O(n^2)$-time algorithm to price the lookback option on the CRR lattice [1]. This paper provides a combinatorial algorithm that can solve the same problem in $O(n)$ time. The running times for both algorithm are listed in Fig. 10. Obviously, our algorithm is much more efficient than Hull's.

6 Conclusions

Combinatorial methods have found wide applicability in many fields. This paper extends their use in improving the performance in pricing a wide variety of options. In particular, it describes how to derive $O(n)$-time combinatorial pricing algorithms for vanilla options, power options, single-barrier options, double-barrier options, and lookback options. These algorithms compare favorably against many other lattice methods, which take at least quadratic computational time.

Acknowledgement

We thanks Chi-Shang Draw for some of the programming works.

References

[1] J. Hull, Options, Futures, and Other Derivatives, 5th ed., Prentice-Hall Press, Englewood Cliffs, New Jersey, 2003.

[2] F. Black, and M. Scholes, The Pricing of Options and Corporate Liabilities, J. Political Econom. 81 (1973) 637–659.

[3] P. Chalasani, S. Jha, and I. Saias, Approximate Option Pricing, Algorithmica 25 (1999) 2–21.

[4] J. Cox, S. Ross, and M. Rubinstein, Option Pricing: A Simplified Approach, J. of Financial Econom. 7 (1979) 229–264.

[5] D. Duffie, Dynamic Asset Pricing Theory, 2nd ed., Princeton University Press, Princeton, NJ, 1996.

[6] P. Boyle, and S. Lau, Bumping Against the Barrier with the Binomial Method, J. Derivatives, 1 (1994) 6–14.

[7] R. Merton, Theory of Rational Option Pricing, Bell J. Econom. and Management 4 (1973) 141–183.

[8] E. Reiner, and M. Rubinstein, Breaking Down the Barriers, Risk 4 (1991) 28–35.

[9] H. Geman, and M. Yor, Pricing and Hedging Double-Barrier Options: A Probabilistic Approach, Math. Finance 6 (1996) 365–378.

[10] N. Kunitomo, and M. Ikeda, Pricing Options with Curved Boundaries, Math. Finance 2 (1992) 275–298.

[11] L. Luo, Various Types of Double Barrier Options, J. Comput. Finance 4 (2001) 125–138.

[12] J. Sidenius, Double Barrier Options: Valuation by Path Counting, J. Comput. Finance 1 (1998) 63–79.

[13] B. Goldman, H. Sosin, and M. A. Gatto, Path-Dependent Options: Buy at the Low, Sell at the High, J. Finance 34 (1979) 1111–1127.

[14] E. Omberg, A Note on the Convergence of Binomial-Pricing and Compound-Option Models, J. Finance 42 (1987), 463–469.

[15] P. Ritchken, On Pricing Barrier Options, J. Derivatives 3 (1995) 19–28.

[16] S. Figlewski, and B. Gao, The Adaptive Mesh Model: A New Approach to Efficient Option Pricing, J. Financial Econom. 53 (1999) 313–351.

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

[18] Y.-D. Lyuu, Very Fast Algorithms for Barrier Option Pricing and the Ballot Problem, J. Derivatives 5 (1998) 68–79.

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

[20] T.-S. Dai, and Y.-D. Lyuu, The Bino-trinomial Tree Model – a Simple Model for Efficient and Accurate Option Pricing, Announced in the Asian FA/FMA 2006 Meeting, Auckland, New Zealand.

[21] J.H. Lint, and R.M. Wilson, A Course in Combinatorics. Cambridge Univ. Press, Cambridge, U.K., 1994.

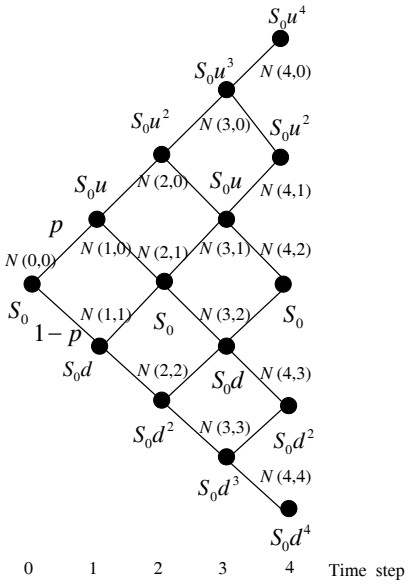

Figure 1: **The CRR Lattice.** The initial stock price is S_0. The upward and downward multiplicative factors for the stock price are u and d, respectively. The upward and downward branching probabilities are p and $1 - p$, respectively.

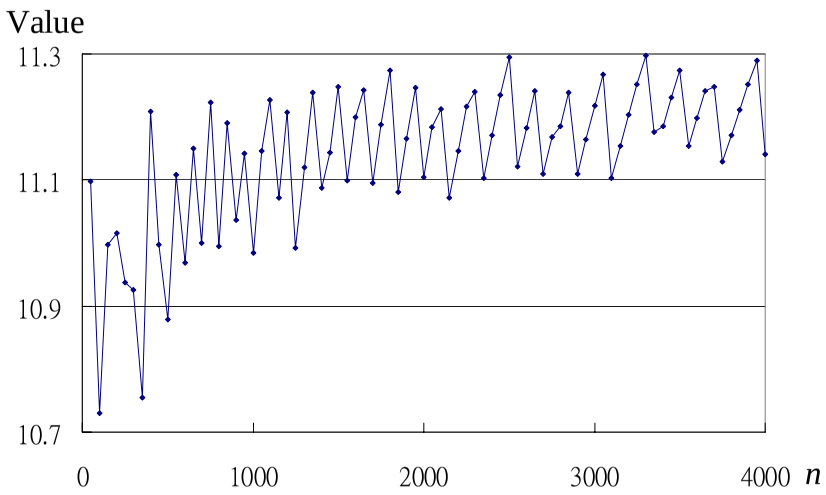

Figure 2: **The Oscillation Phenomenon.** The x-axis and the y-axis denote the number of time steps of the lattice model and the prices for a double-barrier call option, respectively. The numerical settings are as follows: The initial stock price is 95, the exercise price is 97, the time to maturity is 0.75 year, the risk-free rate is 15%, the volatility is 25%, and the two barriers are 80 and 120, respectively.

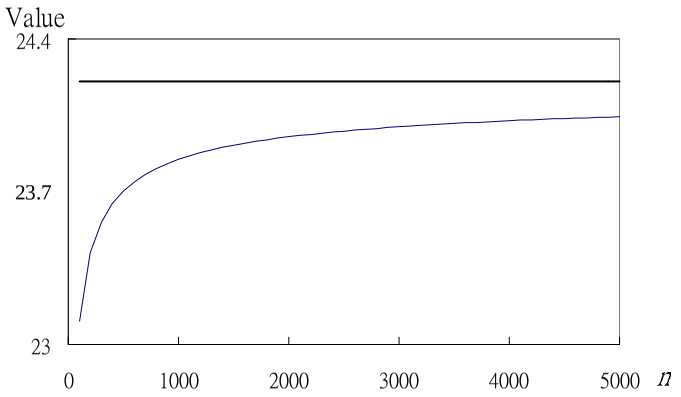

Figure 3: **Slow Convergence.** The x-axis and the y-axis denote the number of time steps of the lattice model and the pricing result, respectively. The theoretical option value 24.20385 priced by the analytical formula [13] is denoted by the flat line. The prices from the lattice method are on the concave curve. The numerical settings are as follows: the initial stock price is 100, the risk-free rate is 6%, the volatility is 30%, and the time to maturity is 1 year.

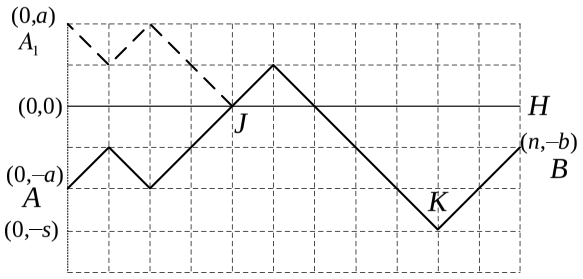

Figure 4: **Counting the Number of Paths That Hit Barrier H by the Reflection Principle.** Barrier H (horizontal line $y = 0$) is denoted by solid line.

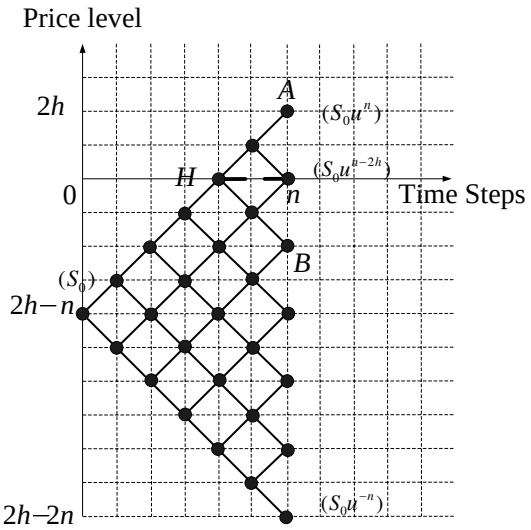

Figure 5: **Placing a CRR Lattice on a Grid for Pricing a Single-Barrier Option.** A CRR lattice drawn in thick solid lines and circles is placed on a grid. The x-axis and the y-axis of this grid are denoted by thin solid lines. The coordinate of the root node of the CRR lattice is $(0, 2h - n)$. The barrier H is denoted by a thick dotted line. The values in parentheses denote stock prices.

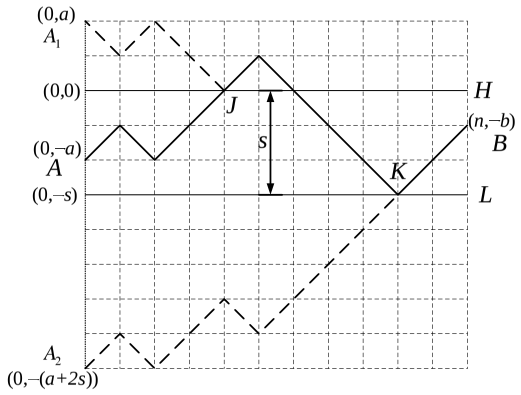

Figure 6: **Counting the Number of Paths That Hit Barrier L or H by the Reflection Principle and the Inclusion-Exclusion Principle.** Both barrier H (horizontal line $y = 0$) and L (horizontal line $y = -s$) are denoted by solid lines.

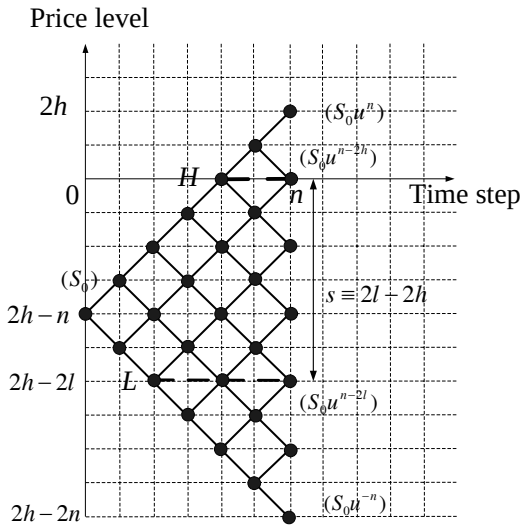

Figure 7: **Placing a CRR Lattice on a Grid for Pricing a Double-Barrier Option.** A CRR lattice drawn in thick solid lines and circles is placed on a grid. The x-axis and the y-axis of this grid are denoted by thin solid lines. The coordinate of the root node of the CRR lattice is $(0, 2h-n)$. The two barriers H and L are denoted by thick dotted lines, and s denotes the distance between H and L. The values in parentheses denote stock prices.

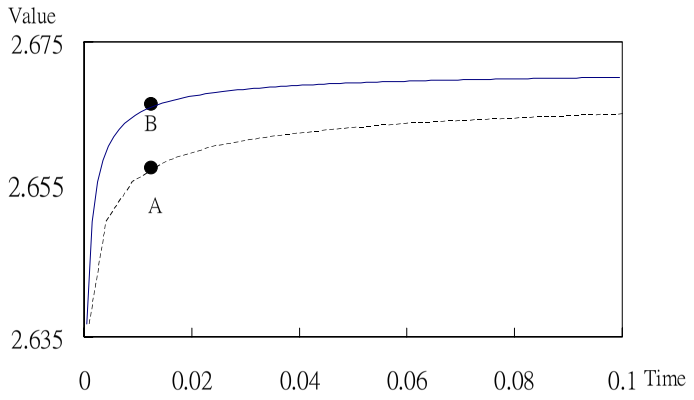

Figure 8: **Pricing a Power Call Option.** The initial stock price is 100, the exercise price is 100, the risk-free rate is 10% per annum, the volatility of the stock price is 30%, and the time to maturity is 1 year. The payoff of the power call option follows Eq. (4) with $p = 0.5$. The solid line and the dotted line denote the combinatorial approach and the dynamic-programming approach, respectively.

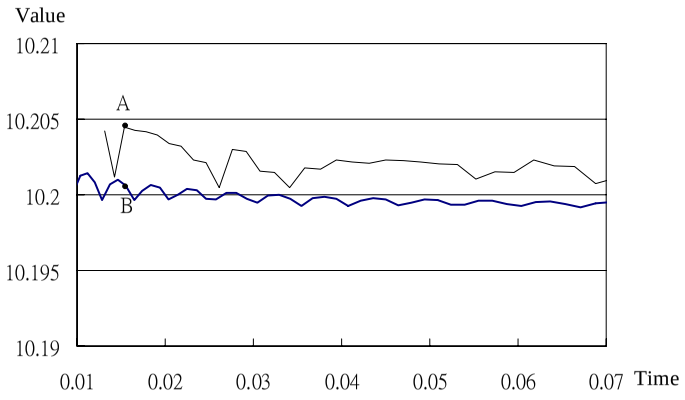

Figure 9: **Pricing a Double-Barrier Call Option.** The initial stock price is 95, the exercise price is 100, the risk-free rate is 10% per annum, the volatility of the stock price is 25%, the time to maturity is 1 year, and the two barriers are 140 and 90, respectively. The thin line and the thick line denote Ritchken's trinomial lattice model and the BTT, respectively.

n	Value	Time (seconds)	
		Hull	Combinatorics
1000	23.848133	1.250	0.002
2000	23.951535	5.015	0.003
3000	23.997554	11.313	0.005
4000	24.025047	20.172	0.006
5000	24.043836	31.687	0.008
Accurate Value: 24.203853			

Figure 10: **Running-Time Comparison for Pricing a Lookback Call Option.** The numerical settings are the same as in Fig. 3. The variable n denotes the number of time steps of the CRR lattice. "Value" denotes the pricing result of the CRR lattice. "Hull" denotes the $O(n^2)$-time algorithm mentioned in [1]. "Combinatorics" denotes the combinatorial algorithm in this paper.