Hindawi
Mathematical Problems in Engineering
Volume 2022, Article ID 5843491, 20 pages
https://doi.org/10.1155/2022/5843491

Research Article

Efficient and Robust Combinatorial Option Pricing Algorithms on the Trinomial Lattice for Polynomial and Barrier Options

Jr-Yan Wang,[1] **Chuan-Ju Wang**,[2] **Tian-Shyr Dai**,[3] **Tzu-Chun Chen**,[4] **Liang-Chih Liu**,[5] **and Lei Zhou**[3]

[1]*Department of International Business, National Taiwan University, Taipei 106, Taiwan*
[2]*Research Center for Information Technology Innovation, Academia Sinica, Taipei 106, Taiwan*
[3]*Department of Information Management and Finance, National Yang Ming Chiao Tung University, Hsinchu 300, Taiwan*
[4]*Department of Computer Science, Queen Mary College, University of London, London E1 4NS, UK*
[5]*Department of Information and Finance Management, National Taipei University of Technology, Taipei 106, Taiwan*

Correspondence should be addressed to Liang-Chih Liu; tony919kimo@gmail.com

Received 3 August 2021; Revised 19 January 2022; Accepted 29 March 2022; Published 21 May 2022

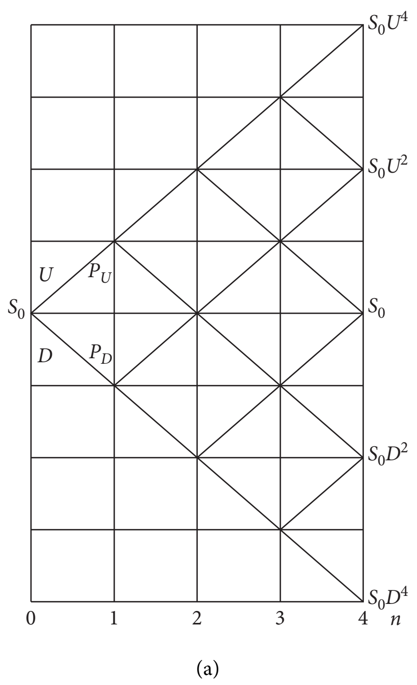
Academic Editor: Kazem Nouri

Options can be priced by the lattice model, the results of which converge to the theoretical option value as the lattice's number of time steps n approaches infinity. The time complexity of a common dynamic programming pricing approach on the lattice is slow (at least $O(n^2)$), and a large n is required to obtain accurate option values. Although $O(n)$-time combinatorial pricing algorithms have been developed for the classical binomial lattice, significantly oscillating convergence behavior makes them impractical. The flexibility of trinomial lattices can be leveraged to reduce the oscillation, but there are as yet no linear-time algorithms on trinomial lattices. We develop $O(n)$-time combinatorial pricing algorithms for polynomial options that cannot be analytically priced. The commonly traded plain vanilla and power options are degenerated cases of polynomial options. Barrier options that cannot be stably priced by the binomial lattice can be stably priced by our $O(n)$-time algorithm on a trinomial lattice. Numerical experiments demonstrate the efficiency and accuracy of our $O(n)$-time trinomial lattice algorithms.

1. Introduction

Options are important financial derivatives whose values depend on other more fundamental financial assets, which are called the underlying assets, for instance, stocks, indexes, and currencies [1]. In this study, for convenience, the underlying asset is assumed to be a stock. To improve the market completeness and efficiency, many complex options have been structured to meet specific financial goals. However, most complex options do not have analytical pricing formulae and must be priced using numerical methods [2]. Exploring efficient and accurate numerical pricing methods is thus important in both theory and practice.

The lattice method is a popular numerical pricing method. It can easily deal with American-style features like early redemption and early exercise that are found in many option contracts. It is also flexible, since only nominal changes are needed to price complex, nonstandard options, which do not have simple closed-form solutions [3] (Indeed, the flexibility of the lattice method makes it widely adopted in recent literature, such as the evaluation of American stock options [6], variable annuity products [7, 8], catastrophe equity puts [9], employee stock options [10], swing options [11], or corporate bonds [12–15]). Technically speaking, a lattice divides the option life into n discrete equal-length time steps and simulates stock price movements discretely at each time step. Four-time-step binomial and trinomial



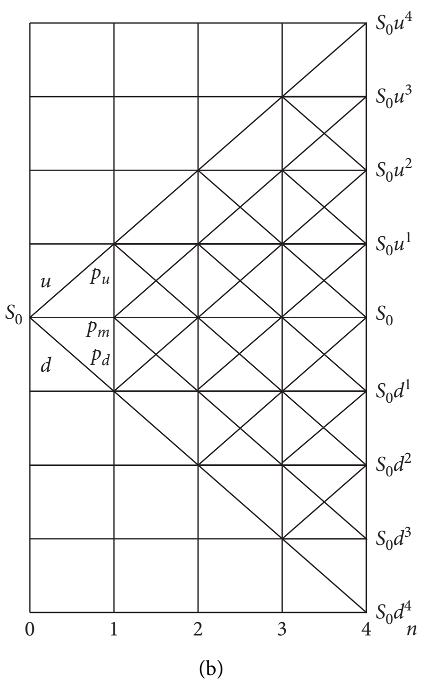


Figure 1: Illustration of binomial and trinomial lattice models. Four-time-step binomial (a) and trinomial (b) lattices are shown. The initial stock prices for both lattices are S_0. In the binomial lattice, the upward and downward multiplicative factors for the stock price are U and D, respectively, where $UD = 1$ based on Cox et al. [4], and the upward and downward branching probabilities are P_U and P_D, respectively. In the trinomial lattice, the upward and downward multiplicative factors for the stock price are u and d (assuming $ud = 1$), respectively, and the upward, middle, and downward branching probabilities are p_u, p_m, and p_d, respectively.

lattices are shown in Figure 1. While each node (except the nodes at the last time step) in the binomial lattice has two branches connected to the two following nodes at the next time step, each node in the trinomial lattice has three branches. The most classical binomial lattice is proposed by Cox, Ross, and Rubinstein [4] (CRR model hereafter) (Rubinstein [16] reviews the relation between binomial and trinomial lattices and shows that they are equivalent to explicit finite difference methods.), where a node with stock price S can move to SU or SD (UD = 1) at the next time step with probabilities P_U and P_D, respectively. Similarly, a node with price S in a trinomial lattice can move to Su, S, and Sd (assuming $ud = 1$ in Figure 1) at the next time step with probabilities p_u, p_m, and p_d, respectively. The branching probabilities and the stock price multiplicative factors (like P_U, P_D, U, and D in the binomial lattice and p_u, p_m, p_d, u, and d in the trinomial lattice) are determined to match the first two moments of distribution of the stock price process, so that the pricing results generated by lattice models converge to the theoretical value as $n \longrightarrow \infty$ [4, 5].

Pricing options on the lattice model can be achieved by dynamic programming. The value for each node in the lattice is recursively evaluated from the last-time-step back to the root node. Thus, it takes at least $O(n^2)$ time (or quadratic time of n) (For an $O(n^2)$-time pricing method, the computational time quadruples when n doubles) to price options on both the binomial and trinomial lattice models, since there are $O(n^2)$ nodes on an n-time-step lattice. The dynamic programming approach can be quite inefficient as the pricing results may converge slowly in n, and thus, a large value for n is required. Fortunately, Dai et al. [3] and Lyuu [17] proposed combinatorial approaches that can efficiently price a variety of options on the CRR binomial lattice in $O(n)$ time (or linear time of n). The computational time of an $(O(n))$-time pricing method doubles when $O(n)$ doubles. (therefore, an $O(n)$-time pricing method runs much faster than a $O(n^2\ )$ one when n is large.)

However, option pricing results generated by the lattice may oscillate wildly with n [18, 19]. For instance, when applying the CRR binomial lattice to price barrier options, the significant oscillation renders it impractical, as shown in Figure 2. To suppress this oscillation, the lattice structure should be adjusted to fit the option's specifications. Since the trinomial lattice is more flexible than the binomial lattice, the structure of the trinomial lattice can be adjusted according to the option contacts to suppress the oscillation [2, 5, 19] (In the literature, various methods are used to mitigate the oscillating convergent behavior of the CRR binomial model; for example, Tian [22] and Klassen [23] for pricing vanilla options and Bolye and Lau [24] for pricing barrier options. Note that the modifications in Tian [22] and Klassen [23] lead to a nonhorizontal binomial tree, i.e., the asset price layers are no longer parallelly horizontal. However, the wide applications of combinatorial option pricing approaches depend crucially on the existence of horizontal layers, especially when they are used to price barrier-dependent options. Consequently, this study develops combinatorial option pricing algorithms based on trinomial lattices consisting of a set of parallelly horizontal layers. Bolye and Lau [24], meanwhile, proposed a method to locate a layer of

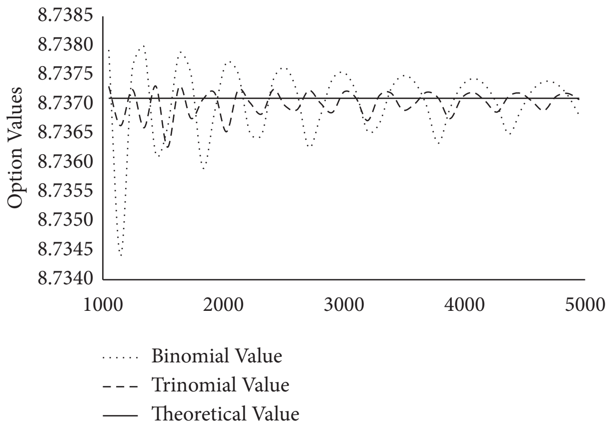


Figure 2: Comparison of convergence behavior between binomial and proposed trinomial lattice models for pricing barrier options. The x-axis and y-axis denote the number of time steps n and the option values, respectively. The dotted and dashed lines denote the pricing results generated by the binomial and our trinomial lattices, respectively. The solid line stands for the theoretical value based on the analytical pricing formula [20, 21]. Clearly, the trinomial lattice model alleviates the price oscillation problem with respect to the number of time steps, n.

nodes as close as possible to the barrier. Nevertheless, Ritchken [19] demonstrated that the trinomial lattice adopted in this study outperforms Bolye and Lau's [24] method in terms of a faster convergence rate. In addition, the exponential error convergence rate of Fourier transform methods has led to their wide adoption in recent derivative pricing literature, such as discretely monitored barrier options [25, 26], continuously monitored barrier options [27], α-quantile and perpetual early exercise options [28], and Asian options [29]. Although lattice methods have slow error convergence rates as in Talponen and Turunen [30], their flexibility has led to their wide adoption in recent literature, as mentioned in footnote 1.). Although the trinomial lattice can alleviate the price oscillation problem, no efficient combinatorial algorithms for the trinomial lattice have been proposed and the inefficient dynamic programming approach, which takes at least $O(n^2)$ time, must be applied. To address this problem, we develop combinatorial pricing algorithms on the trinomial lattice, such that this novel option pricing framework inherits both the efficiency of the combinatorial algorithm and the accuracy of the trinomial lattice.

Developing a combinatorial algorithm for the trinomial lattice is much more difficult than that for the binomial one because the additional middle branch in the trinomial lattice complicates combinatorial formulae. Consider, for example, the number of price paths that start at the root and then finally reach the terminal stock price S_0 at the fourth-time-step, as shown in Figure 1. The number of price paths based on the binomial lattice (Figure 1(a)) is $\binom{4}{2}$ because all price paths in this case should consist of exactly two upward and two downward movements. In contrast, the number of price paths based on the trinomial lattice (Figure 1(b)) is $\binom{4}{0}\binom{4}{2}+\binom{4}{2}\binom{2}{1}+\binom{4}{4}\binom{0}{0}$, where each term represents a different composition of upward, middle, and downward movements for the price paths reaching the terminal node S_0. For instance, the second term indicates that the price paths consist of two middle movements, one upward movement, and one downward movement. This complicated nature might be why no linear-time algorithm has been proposed for the trinomial lattice before this paper.

We address this first by noting that the option pricing formulae for a wide range of options on the trinomial lattice can be decomposed into certain regular summation forms. Then, we prove in two theorems that each term in the summation form can be represented by a simple recursive formula of its preceding term; thus, the summation form can be evaluated in $O(n)$ time. We take polynomial options as the first example to develop an $O(n)$-time combinatorial algorithm for the trinomial lattice because polynomial options are general and able to accommodate widely traded power and vanilla call or put options as special cases, but polynomial options do not have closed-form solutions unless their payoffs satisfy certain restricted criteria [31, 32] (Kim et al. [33] and Zhang et al. [34] price power options as special cases of polynomial options). Other reasons motivating us to consider polynomial options are their flexible applications in practice. For instance, since the unusual payoff functions of exotic path-independent options (or even option portfolios) can be approximated by a polynomial function, the pricing method for the polynomial option can be employed to approximate the values of those exotic options that might not have closed-form solutions. In addition, if a portfolio is exposed to a risk factor in a nonlinear manner, the polynomial option can provide a more appropriate payoff to hedge this portfolio; such a hedge strategy is usually much cheaper than hedging with a combination of more fundamental derivatives. As a result, developing efficient and accurate algorithms for pricing polynomial options is an important and valuable issue. Furthermore, to demonstrate the versatility of the theorems proposed in this study, we develop the reflection principle on the trinomial lattice and then extend the combinatorial pricing algorithms on the binomial lattice in the study by Dai et al. [3] to evaluate barrier options on the trinomial lattice. The numerical results verify that our pricing algorithms run in $O(n)$ time and show that our algorithms are superior in terms of accuracy and efficiency.

The structure of this study is as follows. We introduce fundamental knowledge about option pricing and the combinatorial approach in Section 2 and propose two novel combinatorial theorems for option pricing in Section 3. In Section 4, we apply these theorems to develop linear-time algorithms for polynomial, power, plain vanilla, and barrier options. We present the experimental results in Section 5 and conclude in Section 6.

2. Preliminaries

2.1. Basic Framework

2.1.1. Stock Price Process. Suppose an option initiates at time 0 and matures at time T. Let S_t denote the stock price at time

t, where $0 \le t \le T$. In addition, this study adopts the most common assumption for the S_t process on option pricing, which states that under the risk-neutral measure, S_t follows a log-normal diffusion process with drift $\mu = r - \sigma^2/2$ (this paper takes a non-dividend-paying stock for example. It is straightforward to extend our model to consider a stock paying a continuously compounding dividend yield) and volatility σ, where r is the risk-free rate.

2.1.2. Kamrad and Ritchken Lattice (KRL) Model. Kamrad and Ritchken [5] proposed a trinomial lattice model, as shown in Figure 1(b). Denote $\Delta t = T/n$ as the length of each time step. The KRL model parameterizes upward and downward multiplication factors u and d as

$$\begin{aligned} u &= e^{\lambda\sigma\sqrt{\Delta t}}, \\ d &= u^{-1}, \end{aligned} \tag{1}$$

where λ is the stretch parameter that can be adjusted to fit the option's specification to alleviate the oscillation problem. To ensure that the trinomial lattice matches the drift (μ) and volatility (σ) of the stock price process, Kamrad and Ritchken set the upward, middle, and downward branching probabilities p_u, p_m, and p_d as

$$\begin{cases} p_u = \dfrac{1}{2\lambda^2} + \dfrac{\mu\sqrt{\Delta t}}{2\lambda\sigma}, \\ p_m = 1 - \dfrac{1}{\lambda^2}, \\ p_d = \dfrac{1}{2\lambda^2} - \dfrac{\mu\sqrt{\Delta t}}{2\lambda\sigma}. \end{cases} \tag{2}$$

According to Ritchken [19], the oscillation problem can be significantly reduced by causing the lattice to coincide with a certain price level Y. Following this idea, we define λ as

$$\lambda = \begin{cases} 1.224745, & \text{if } Y = S_0 \text{ or } Y \text{ is not available}, \\ \dfrac{\left|\mathrm{In}\left(Y/S_0\right)/\sigma\sqrt{\Delta t}\,\right|}{\left|\left\lfloor \mathrm{In}\left(\left(Y/S_0\right)\right)/\sigma\sqrt{\Delta t}\,\right\rfloor\right|}, & \text{otherwise}. \end{cases} \tag{3}$$

Note that when $Y = S_0$ (there must be a stock price layer coinciding with Y) or Y is not available (there is no specific level in the option contract to fit), we follow Kamrad and Ritchken [5] in setting λ as 1.224745, such that all branching probabilities converge to 1/3 when $n \longrightarrow \infty$, which streamlines convergence for pricing options.

2.2. Polynomial Option. One major contribution of this study is the pricing of polynomial options—which do not have closed-form solutions unless polynomial payoffs satisfy some restricted criteria [31, 32]—with the trinomial lattice model in linear time. A general payoff form is considered as

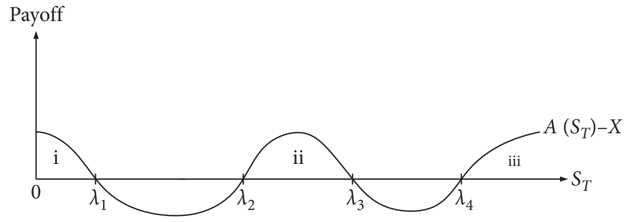


FIGURE 3: Illustration of payoff function of polynomial option. This figure shows that the holder of this polynomial option can receive a positive payoff if S_T belongs to regions i, ii, or iii, which in this paper are termed positive-payoff regions. Each positive-payoff region can be represented by a closed interval $[\lambda_l, \lambda_u]$; therefore, regions i, ii, and ii can be represented by $[0, \lambda_1]$, $[\lambda_2, \lambda_3]$, and $[\lambda_4, \infty]$, respectively.

$$\text{Payoff} = \max\left(A\left(S_T\right) - X, 0\right), \tag{4}$$

where S_T denotes the stock price on the maturity date T, and $A(S_T)$ can be any polynomial function defined as $A(S_T) \equiv \alpha_1 S_T^{q_1} + \alpha_2 S_T^{q_2} + \cdots + \alpha_D S_T^{q_D}$, where α_i's can be any real numbers and q_i's are the nonidentical real numbers. Typically, the $A(S_T) - X$ curve may intersect the S_T-axis several times as shown in Figure 3. Consequently, there may be several separate regions that can generate positive payoffs, such as regions i, ii, and iii in Figure 3.

2.3. Power Option. A power option grants the holder a right to buy or sell the power of the stock price for the exercise price X at the maturity date T. On the other hand, the holder can junk the call option (or the put option) if the power of the stock price is lower (or higher) than the exercise price X at time T. The payoff of a power option can be defined as

$$\text{Payoff} = \max\left(\theta S_T^q - \theta X, 0\right), \tag{5}$$

where q can be any real number and θ equals 1 for call options and −1 for put options.

2.4. Vanilla Option. A vanilla option, a special case of the power option by setting $q = 1$, is a standard option without unusual features. The payoff for a vanilla option at time T is

$$\text{Payoff} = \max\left(\theta S_T - \theta X, 0\right), \tag{6}$$

where θ equals 1 for call options and −1 for put options. Since the payoff function of the polynomial option can accommodate the payoffs of the power and vanilla options as special cases, it is straightforward to employ our combinatorial pricing algorithm for the polynomial option to evaluate these two options. Moreover, to alleviate the oscillation problem when pricing power and vanilla options, the KRL lattice requires a price level that coincides with X (i.e., $Y = X$ in equation (3)) [23].

2.5. Barrier Option. A barrier option is an option whose payoff depends on whether the stock price path ever hits a certain price level H called a barrier. Our study focuses on "up-and-in" barrier options; the extension to other types of barrier options is straightforward. Furthermore, we consider

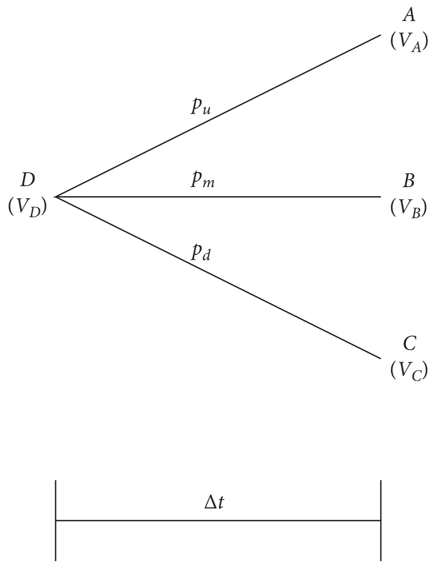


Figure 4: Dynamic programming approach for trinomial lattice. The node D has three outgoing branches to nodes A, B, and C with probability p_u, p_m, and p_d, respectively. V_y denotes the option value for any node (y). The length of each time step is denoted as Δt. In the dynamic programming approach, the value V_D can be computed by the following backward induction formula: $V_D = e^{-r\Delta t}(p_u V_A + p_m V_B + p_d V_C)$, where r is the risk-free interest rate.

the case of $H > S_0$ for convenience. Thus, an "up-and-in" barrier option can only be exercised at maturity when the stock price path has risen to hit the barrier during the option life. Define $S_{\sup} = \sup_{0\le t\le T} S_t$. The payoff function for a barrier option at time T is

$$\text{Payoff} = \begin{cases} \max(\theta S_T - \theta X, 0), & \text{if } S_{\sup} \ge H, \\ 0, & \text{otherwise,} \end{cases} \tag{7}$$

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

To alleviate the oscillation problem, Ritchken [19] suggests that a price level of the lattice should coincide with the barrier H (i.e., $Y = H$ in equation (3)).

2.6. *Option Pricing Based on the KRL Model.*

The theoretical value of an option equals the discounted expected payoff of the option under the risk-neutral measure:

$$e^{-rT}E[\text{payoff}]. \tag{8}$$

The above formula can be evaluated via dynamic programming by computing the option value of each node on the lattice model from time step n back to time step 0. Take the polynomial option, for example. The value of each node at maturity can be evaluated by equation (4). In Figure 4, the examined node, say D, has three descendant nodes, say A, B, and C. Define V_y as the option value for any node y. The value V_D of nodes other than the nodes at maturity can be computed by following backward induction formula.

The pricing result is the value of the root of the lattice computed recursively by the aforementioned formula. This naive pricing method takes at least $O(n^2)$ time as there are

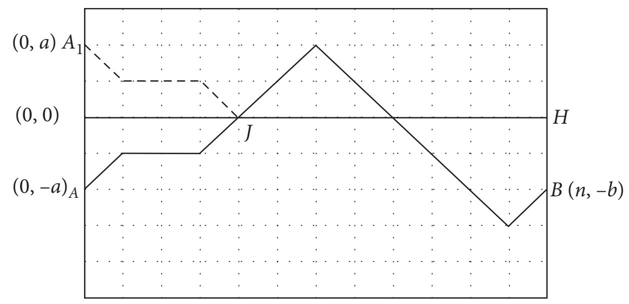


Figure 5: Reflection principle on trinomial lattice. The reflection principle can be applied to the trinomial lattice; it asserts that the number of paths that hit barrier H before arriving at node B from node A is equal to the number of paths from node A_1 to node B.

$O(n^2)$ nodes in the n-time-step KRL model. When pricing barrier options, one must consider not only the stock price but also other path-dependent variables; thus, the dynamic programming approach is more complex, and the order of computational time complexities could be higher than two [4, 35].

Instead of evaluating the option value of each node, the combinatorial approach improves efficiency by grouping stock price paths with the same payoff and calculating their contribution to the option value. Define G as the set of all possible stock paths from time step 0 to time step n. If we define $\pi_{k,z}$ as a subset of G that contains price paths with $n-k$ middle movements, $x \equiv k - z$ upward movements, and z downward movements, then the set of subsets $\{\pi_{k,z} \mid 0 \le z \le k \le n\}$ forms a partition of G. Thus, the theoretical value of a polynomial option can be expressed as

$$\begin{aligned} & e^{-rT}E[\text{payoff}] \\ & = \sum_{\tau\in\mathcal{G}} prob(\tau)\text{payoff}(\tau) \\ & = \sum_{\pi_{k,z}} |\pi_{k,z}| p_u^{k-z} p_m^{n-k} p_d^{z} \max\left(A\left(S_0 u^{k-z} d^z\right) - X, 0\right), \end{aligned} \tag{9}$$

where payoff (τ) denotes the payoff corresponding to path τ at maturity and

$$|\pi_{k,z}| = \binom{n}{n-k}\binom{k}{z}, \tag{10}$$

denotes the number of paths in the subset $\pi_{k,z}$. The second equality is based on the fact that the probability of each path in $\pi_{k,z}$ is $p_u^{k-z} p_m^{n-k} p_d^{z}$ and the payoff of each path in $\pi_{k,z}$ is the same as $\max(A(S_0 u^{k-z} d^z) - X, 0)$. The following sections will show how to evaluate equation (9) in linear time of n. Moreover, similar techniques plus the reflection principle can also be applied to develop $O(n)$ time pricing algorithms for barrier options.

2.7. *Reflection Principle on the KRL Trinomial Lattice Model.*

The reflection principle efficiently counts the number of paths that hit a specific price level before reaching a certain node at maturity in the KRL model. This principle is essential when pricing barrier options using combinatorial methods.

In contrast to applying the reflection principle on the binomial lattice to price barrier options in the study by Dai et al. [3] and Lyuu [17], this study extends their work to the trinomial lattice. The grid in Figure 5 represents the structure of the KRL model, where x and y-axes denote the time step and the stock price level, respectively. Each node (i, j) can move to node $(i+1, j+1)$ (upward movement), node $(i+1, j)$ (middle movement), or node $(i+1, j-1)$ (downward movement) at the next time step. The reflection principle can help us count the number of paths from node $A(0, -a)$ to node $B(n, -b)$ that hit the barrier H. Without loss of generality, we 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 path $\widehat{AJ}$ with respect to barrier H to obtain $\widehat{A_1J}$ (the dashed path). 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.

Suppose the number of upward and downward movements is x and z, respectively, and the sum of upward and downward movements is k (given the number of middle movements to be $n-k$). Thereby,

$$\begin{cases} x + z = k, \\ z - x = a + b. \end{cases} \tag{11}$$

The first equation is from the definition of x, z, and k, and the second equation ensures that the number of downward movements exceeds the number of upward movements by $a+b$, which is the distance in y-axis between node A_1 and node B. By solving the aforementioned system of equations for x and z, we have

$$\begin{cases} x = \dfrac{(k-(a+b))}{2}, \\ z = \dfrac{(k+(a+b))}{2}. \end{cases} \tag{12}$$

Thus, the number of paths that hit barrier H before arriving node B from A and that have $n-k$ middle movements is

$$\binom{n}{n-k}\binom{k}{x} = \binom{n}{n-k}\binom{k}{\dfrac{(k-(a+b))}{2}}, \tag{13}$$

where $\binom{k}{(k-(a+b))/2} \equiv 0$ if $(k-(a+b))/2$ is not a nonnegative integer. Thus, the total number of paths hitting barrier H before arriving B from A is

$$\sum_{k=0}^{n} \binom{n}{n-k}\binom{k}{\dfrac{(k-(a+b))}{2}}. \tag{14}$$

3. Linear-Time Algorithms for Evaluating Summations in Option Pricing Formulae

Trinomial lattice option pricing formulae such as equation (9) can be expressed in terms of a linear combination of summations $\sum_{k=c}^{n} \omega(k)$, where $\omega(k)$ follows certain forms (introduced later) and c is a nonnegative integer smaller than n, the number of time steps in the lattice model. These summations can be evaluated in linear time (i.e., in $O(n)$ time) if the first term $\omega(c)$ can be evaluated in $O(n)$ time and each following term (i.e., $\omega(k)$ for $c < k \leq n$) can be iteratively evaluated in $O(1)$ time. The aforementioned linear-computation property is the main idea to construct our linear-time option pricing algorithms. Define the set F to contain all the sequences (e.g., $\{\omega(k)\}$) that have this linear-computation property. The following proofs show that the series expressed in certain forms (this will be used to develop pricing algorithms later) belong to F.

Lemma 1. *Define $V(k)$ as $\beta\binom{n}{k}\alpha^k$, where β can be evaluated in $O(n)$ time, α is a real number, and k is a nonnegative running integer variable. Then, the sequence $\{V(k)\}$ belongs to the set F.*

Proof. Suppose $V(c) = \beta\binom{n}{c}\alpha^c = \beta n!/c!(n-c)!\alpha^c$ is the first term of the sequence $\{V(k)\}$, where c is a nonnegative integer smaller than n. Then, $V(c)$ can be evaluated in $O(n)$ time since β, $n!$, $c!$, $(n-c)!$, and α^c can all be evaluated in $O(n)$ time. In addition, each following term $V(k)$ can be iteratively computed in $O(1)$ time by the recurrence formula $V(k) = V(k-1)\alpha(n-k+1)/k$ for $c < k \leq n$. Consequently, we can obtain $\{V(k)\} \in F$, and thus, $\sum_{k=c}^{n} V(k)$ can be evaluated in $O(n)$ time. □

Lemma 2. *Define$\omega(k) \equiv V(k)\sum_{z=0}^{M(k)}\binom{k}{z}a^z b^{k-z}$, where k and z are the nonnegative running integer variables, a and b are the real-number constants, and $M(k)$ is a nonnegative integer-valued function. Then, the following formula holds:*

$$V(k)\sum_{z=1}^{M(k)}\binom{k+1}{z}a^z b^{k+1-z} = (b+a)\omega(k) - V(k)\binom{k}{0}a^0 b^{k+1} - V(k)\binom{k}{M(k)}a^{M(k)+1}b^{k-M(k)}. \tag{15}$$

Proof. First, $b\omega(k)$ and $a\omega(k)$ can be expressed as

$$\begin{aligned} b\omega(k) &= V(k)\sum_{z=0}^{M(k)}\binom{k}{z}a^z b^{k+1-z} \\ &= V(k)\binom{k}{0}a^0 b^{k+1} + V(k)\sum_{z=1}^{M(k)}\binom{k}{z}a^z b^{k+1-z}. \end{aligned} \quad (16)$$

$$\begin{aligned} a\omega(k) &= V(k)\sum_{z=0}^{M(k)}\binom{k}{z}a^{z+1} b^{k-z} \\ &= V(k)\sum_{z=0}^{M(k)-1}\binom{k}{z}a^{z+1} b^{k-z} V(k)\binom{k}{M(k)}a^{M(k)+1}b^{k-M(k)}. \end{aligned} \quad (17)$$

The sum of the second term in equation (16) and the first term in equation (17) can be simplified as

$$\begin{aligned} &V(k)\sum_{z=1}^{M(k)}\binom{k}{z}a^z b^{k+1-z} + V(k)\sum_{z=0}^{M(k)-1}\binom{k}{z}a^{z+1}b^{k-z} \\ &= V(k)\sum_{z=1}^{M(k)}\binom{k}{z}a^z b^{k+1-z} + V(k)\sum_{z=1}^{M(k)}\binom{k}{z-1}a^z b^{k-(z-1)} \\ &= V(k)\sum_{z=1}^{M(k)}\left[\binom{k}{z}+\binom{k}{z-1}\right]a^z b^{k-z+1} \\ &= V(k)\sum_{z=1}^{M(k)}\binom{k+1}{z}a^z b^{k-z+1}, \end{aligned} \quad (18)$$

where the third equality is due to the combinatorial addition identity. Thus,

$$\begin{aligned} (b+a)\omega(k) = V(k)\binom{k}{0}a^0 b^{k+1} + V(k)\sum_{z=1}^{M(k)}\binom{k+1}{z}a^z b^{k-z+1} \\ + V(k)\binom{k}{M(k)}a^{M(k)+1}b^{k-M(k)}. \end{aligned} \quad (19)$$

As a result, Lemma 2 can be obtained by rewriting the above formula.

The following theorems and corollaries show that the summation in certain formats can be evaluated in $O(n)$ time, which is useful for deriving linear-time combinatorial option pricing algorithms. □

Theorem 1. *If the sequence $\{V(k)\} \in F$, then the sequence $\{\omega(k)\}$ defined in Lemma 2 also belongs to F under the premise that $M(k+1) - M(k) = 1$, that is, $\sum_{k=c}^{n}\omega(k)$ can be evaluated in $O(n)$ time given that c is a nonnegative integer smaller than n.*

Proof. To show that $\{\omega(k)\} \in F$, we first prove that the first term of the sequence, $\omega(c) = V(c)\sum_{z=0}^{M(c)}\binom{c}{z}a^z b^{c-z}$, can be evaluated in $O(n)$ time. This can be achieved by showing that the sequence $\left\{\delta(z) \equiv \binom{c}{z}a^z b^{c-z}\right\}$, the terms in the summation, belongs to F. Since $\delta(0) = \binom{c}{0}a^0 b^{c-0} = b^c$ can be evaluated in $O(n)$ time and $\delta(z) = \delta(z-1)(c-z+1/z)(a/b)$ suggests that each $\delta(z)$ can be evaluated in $O(1)$ time given $\delta(z-1)$ is known, we can infer that $\{\delta(z)\} \in F$. Thus, $\omega(c) = V(c)\sum_{z=0}^{M(c)}\binom{c}{z}a^z b^{c-z} = V(c)\sum_{z=0}^{M(c)}\delta(z)$ can be evaluated in $O(n)$ time since both $V(c)$ and $\sum_{z=0}^{M(c)}\delta(z)$ can be evaluated in $O(n)$ time.

The next task is to express $\omega(k+1)$ in terms of $\omega(k)$, such that $\omega(k+1)$ can be iteratively computed in $O(1)$ time given the value of $\omega(k)$. Rewrite $\omega(k+1)$ as

$$\begin{aligned} \omega(k+1) &= V(k+1)\sum_{z=0}^{M(k+1)}\binom{k+1}{z}a^z b^{k+1-z} \\ &= \frac{V(k+1)}{V(k)}\left(V(k)\sum_{z=0}^{M(k)+1}\binom{k+1}{z}a^z b^{k+1-z}\right) \\ &= \frac{V(k+1)}{V(k)}\left(\begin{matrix} V(k)\binom{k+1}{0}a^0 b^{k+1} + V(k)\sum_{z=1}^{M(k)}\binom{k+1}{z}a^z b^{k+1-z} \\ +V(k)\binom{k+1}{M(k)+1}a^{M(k)+1}b^{k-M(k)} \end{matrix}\right). \end{aligned} \quad (20)$$

By replacing $V(k)\sum_{z=1}^{M(k)}\binom{k+1}{z}a^z b^{k+1-z}$ with Lemma 2, $\omega(k+1)$ can be expressed in terms of $\omega(k)$ as

$$\begin{aligned}\omega(k+1) &= \frac{V(k+1)}{V(k)}\left(\begin{array}{c} V(k)\binom{k+1}{0}a^0 b^{k+1} + (b+a)\omega(k) - V(k)\binom{k}{0}a^0 b^{k+1} \\ -V(k)\binom{k}{M(k)}a^{M(k)+1}b^{k-M(k)} + V(k)\binom{k+1}{M(k)+1}a^{M(k)+1}b^{k-M(k)} \end{array}\right) \\ &= \frac{V(k+1)}{V(k)}\left((b+a)\omega(k) + V(k)\binom{k}{M(k)}\frac{k-M(k)}{M(k)+1}a^{M(k)+1}b^{k-M(k)}\right) \\ &= \frac{V(k+1)}{V(k)}(b+a)\omega(k) + V(k+1)\binom{k}{M(k)}\frac{k-M(k)}{M(k)+1}a^{M(k)+1}b^{k-M(k)}. \end{aligned} \tag{21}$$

Define functions f_1 and g_1 as

$$\begin{aligned} f_1(k) &= \left(\frac{V(k+1)}{V(k)}\right)(b+a), \\ g_1(k) &= V(k+1)\binom{k}{M(k)}\frac{k-M(k)}{M(k)+1}a^{M(k)+1}b^{k-M(k)} \\ &= g_1(k-1)\frac{V(k-1)}{V(k)}\frac{ak}{M(k)+1}. \end{aligned} \tag{22}$$

Then, we have $\{f_1(k)\} \in F$, since $\{V(k)\} \in F$ and each $f_1(k)$ can be parallelly evaluated in constant time during the calculation of $V(k+1)$ given $V(k)$. Following the same logic, $\{g_1(k)\} \in F$ because given $g_1(k-1)$, $g_1(k)$ can be evaluated in constant time in parallel when evaluating $V(k+1)$ given $V(k)$.

Since $\omega(k+1)$ can be expressed in terms of $\omega(k)$ as

$$\omega(k+1) = f_1(k)\omega(k) + g_1(k), \tag{23}$$

$\sum_{k=c}^{n}\omega(k)$ can be computed in $O(n)$ time through equation (23) by evaluating $\{f_1(k)\}$ and $\{g_1(k)\}$ in parallel.

In the following corollaries, we consider the different integer-valued function $M(k)$ and show the resulting sequence $\{\omega(k)\} \in F$. □

Corollary 1. *$\sum_{k=c}^{n}\omega(k)$ can be computed in $O(n)$ time when $M(k+1) - M(k) = 0$.*

Proof. Similarly, $\omega(k+1)$ can be expressed as equation (20) except that $M(k+1) = M(k)$:

$$\begin{aligned}\omega(k+1) &= \frac{V(k+1)}{V(k)}\left(V(k)\sum_{z=0}^{M(k)}\binom{k+1}{z}a^z b^{k+1-z}\right) \\ &= \frac{V(k+1)}{V(k)}\left(V(k)\binom{k+1}{0}a^0 b^{k+1} + V(k)\sum_{z=1}^{M(k)}\binom{k+1}{z}a^z b^{k+1-z}\right). \end{aligned} \tag{24}$$

By replacing $V(k)\sum_{z=1}^{M(k)}\binom{k+1}{z}a^z b^{k+1-z}$ with the formula in Lemma 2, $\omega(k+1)$ can be further expressed as

$$\omega(k+1)=\frac{V(k+1)}{V(k)}\left(\begin{array}{c}V(k)\binom{k+1}{0}a^{0}b^{k+1}+(b+a)\omega(k)-V(k)\binom{k}{0}a^{0}b^{k+1}\\ -V(k)\binom{k}{M(k)}a^{M(k)+1}b^{k-M(k)}\end{array}\right)$$
$$=\frac{V(k+1)}{V(k)}(b+a)\omega(k)-V(k+1)\binom{k}{M(k)}a^{M(k)+1}b^{k-M(k)}. \tag{25}$$

Define $g_2(k)$ to be a function as follows:

$$g_2(k)=V(k+1)\binom{k}{M(k)}a^{M(k)+1}b^{k-M(k)}$$
$$=g_2(k-1)\frac{V(k+1)}{V(k)}\frac{bk}{k-M(k)}. \tag{26}$$

Clearly, $\{g_2(k)\}\in F$ since $\{V(k)\}\in F$ and $g_2(k)$ can be evaluated in constant time given $g_2(k-1)$ in parallel with the calculation of $V(k+1)$ given $V(k)$. Since $\omega(k+1)$ can be expressed in terms of $\omega(k)$ as

$$\omega(k+1)=f_1(k)\omega(k)-g_2(k), \tag{27}$$

$\sum_{k=c}^{n}\omega(k)$ can be computed in $O(n)$ time via equation (27) by parallelly evaluating $\{f_1(k)\}$ and $\{g_2(k)\}$. □

Corollary 2. *$\sum_{k=c}^{n}\omega(k)$ can be computed in $O(n)$ time when $M(k+1)-M(k)=0$ or 1 (Note that the value $M(k+1)-M(k)$ is determined by the nonnegative integral running variable k. It is not a random variable).*

Proof. Since $\{f_1(k)\}$, $\{g_1(k)\}$, $\{g_2(k)\}\in F$, and the first term of $\{\omega(k)\}$ and $\omega(c)$, can be evaluated in $O(n)$ time as mentioned in Theorem 1, each following term $\omega(k+1)$ (note that $k+1>c$.) can be recursively evaluated from the precedent term $\omega(k)$ in $O(1)$ time by equation (21) or (23) if $M(k+1)-M(k)=1$ or 0, respectively. Thus, $\sum_{k=c}^{n}\omega(k)$ can be computed in $O(n)$ time. □

Theorem 2. *Let $\{V(k)\}$ belonging to F, k, and z be the nonnegative running integer variables, a and b be the real-number constants, and both $M(k)$ and $m(k)$ be the nonnegative integer-valued functions with the properties $M(k+1)-M(k)=0$ or 1 and $m(k+1)-m(k)=0$ or 1. Then, a more general series $\left\{\omega\prime(k)\right\}$ with the definition $\omega\prime(k)\equiv V(k)\sum_{z=m(k)}^{M(k)}\binom{k}{z}a^{z}b^{k-z}$ belongs to F, given that c is a nonnegative integer smaller than n.*

Proof. We can rewrite $\omega\prime(k)$ as

$$\omega'^{(k)}=V(k)\sum_{z=0}^{M(k)}\binom{k}{z}a^{z}b^{k-z}-V(k)\sum_{z=0}^{m(k)-1}\binom{k}{z}a^{z}b^{k-z}. \tag{28}$$

We next define $\omega_M(k)\equiv V(k)\sum_{z=0}^{M(k)}\binom{k}{z}a^{z}b^{k-z}$ and $\omega_m(k)\equiv V(k)\sum_{z=0}^{m(k)-1}\binom{k}{z}a^{z}b^{k-z}$, since both $\sum_{k=c}^{n}\omega_M(k)$ and $\sum_{k=c}^{n}\omega_m(k)$ can be evaluated in $O(n)$ time by specifying the upper bounds of the running variable z to be $M(k)$ and $m(k)-1$ in Theorem 1 (including Corollary 1 and 2), respectively. As a consequence, $\sum_{k=c}^{n}\omega\prime(k)$ can be evaluated in $O(n)$ time and $\left\{\omega\prime(k)\right\}\in F$. □

3.1. Combinatorial Option Pricing Algorithms. We will first develop a linear-time pricing algorithm for polynomial options (with the payoff in equation (4)) by calculating the discounted expected payoffs with the aforementioned theorems and corollaries. Since power and vanilla options are special cases of polynomial options (their payoffs are shown in equations (5) and (6), respectively), our pricing algorithm for polynomial options can be easily extended to price power options or vanilla options. In addition, the reflection principle on the KRL trinomial lattice is applied to price barrier options. Our algorithms focus on call options; the extension to put options is straightforward.

4. Linear-Time Algorithm for Polynomial Options on the KRL Model

For convenience, we first introduce useful notation for developing our combinatorial pricing algorithm. On the KRL trinomial lattice, S_0, u, and $d\equiv u^{-1}$ denote the initial stock price, the upward and the downward multiplication factors, respectively. In addition, each price path is described with $n-k$ middle movements, $x(=k-z)$ upward movements, and z downward movements; thus the corresponding stock price at maturity is $S_0u^{k-z}d^{z}=S_0u^{k-2z}$, for $0\le z\le k\le n$. As a consequence, there are a total of $2n+1$ attainable stock prices at maturity listed from the highest to the lowest as follows: $S_0u^{n}, S_0u^{n-1},\ldots,S_0u^{0},S_0u^{-1},\ldots,S_0u^{-n}$. Thus, we number the stock price layer from n to $-$n according to the exponent of u of the attainable stock prices at maturity. Finally, we identify the stock price layer c_l and c_u that makes $[S_0u^{c_l}, S_0u^{c_u}]$ a largest interval bracketed by a positive-payoff region $[\lambda_l, \lambda_u]$ through the following formulae:

$$c_l = \left\lceil \frac{\ln(\lambda_l/S_0)}{\ln u} \right\rceil,$$
$$c_u = \left\lfloor \frac{\ln(\lambda_u/S_0)}{\ln u} \right\rfloor. \quad (29)$$

Note that neither c_l nor c_u can be less than −n or greater than n since the highest and the lowest stock prices attainable at maturity are S_0u^n and S_0u^{-n}, respectively. Therefore, to ensure that equation (29) is well-defined under all scenarios, two extreme cases are considered in our model: if $0 \le \lambda_l \le S_0u^{-n}$, c_l is set to −n, and if $S_0u^n \le \lambda_u < \infty$, c_u is set to n.

Next, we classify each positive-payoff region of a polynomial option into one of the following three cases: $\lambda_l < \lambda_u \le S_0$, $S_0 \le \lambda_l < \lambda_u$, and $\lambda_l < S_0 < \lambda_u$. For each case, we can develop a combinatorial algorithm to evaluate the option value contributed by the positive-payoff region in linear time of n. Thus, the total value of a polynomial option can also be evaluated in linear time of n by accumulating the option values contributed by all positive-payoff regions.

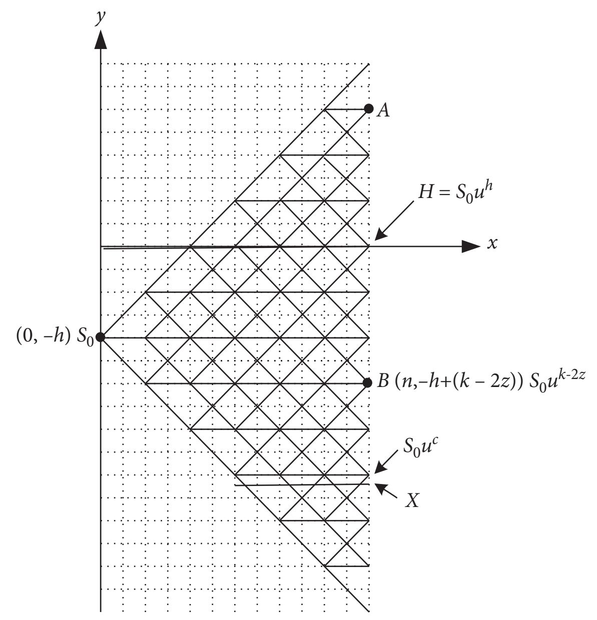


Figure 6: KRL model placed on grid for pricing barrier options. The KRL model drawn in solid lines is placed on a dashed line grid, where x-axis represents the stock price layer that coincides with barrier H and y-axis represents time step 0. The coordinates of root S_0 and node B are $(0, -h)$ and $(n, -h + (k-2z))$, respectively.

Case 1. $\lambda_l < \lambda_u \le S_0$

We first derive the constraints for the price paths that end up in the positive-payoff region. Due to the conditions $\lambda_l < \lambda_u \le S_0$, the corresponding price layers c_{1l} and c_{1u} can be derived based on equation (29), and should satisfy $-n \le c_{1l} \le c_{1u} \le 0$. Since the stock price at maturity S_T can be expressed as $S_0u^{k-z}d^z = S_0u^{k-2z}$, and S_T must be within the range $[S_0u^{c_{1l}}, S_0u^{c_{1u}}]$, we derive the first constraint for k and z as

$$c_{1l} \le k - 2z \le c_{1u} \Rightarrow \frac{k - c_{1u}}{2} \le z \le \frac{k - c_{1l}}{2}. \quad (30)$$

In addition, since the number of downward movements z should be no less than 0 and no larger than the sum of the numbers of upward and downward movements, k, we derive the second constraint for k and z as $0 \le z \le k$. Finally, a stock price path should have at least $-c_{1u}$ downward movements to ensure that $S_T \le S_0u^{c_{1u}}$; thus the sum of the numbers of upward and downward movements, k, should be no less than $-c_{1u}$ and no larger than the number of time steps, n, i.e., $-c_{1u} \le k \le n$, which is the third constraint. The option value contributed from the positive-payoff region $[\lambda_l, \lambda_u]$ can be expressed as the sum of the values contributed by the stock price paths that satisfy the above three constraints as

$$e^{-rT} \sum_{k=-c_{1l}}^{n} \binom{n}{n-k} p_m^{n-k} \sum_{z=U(k,c_{1u})}^{L(k,c_{1l})} \binom{k}{z} p_u^{k-z} p_d^z \,\text{payoff}\left(S_0u^{k-z}d^z\right)$$
$$+ e^{-rT} \sum_{k=-c_{1u}}^{-c_{1l}-1} \binom{n}{n-k} p_m^{n-k} \sum_{z=U(k,c_{1u})}^{k} \binom{k}{z} p_u^{k-z} p_d^z \,\text{payoff}\left(S_0u^{k-z}d^z\right), \quad (31)$$

where

$$L(k,c) \equiv \left\lfloor \frac{(k-c)}{2} \right\rfloor,$$
$$U(k,c) \equiv \left\lceil \frac{(k-c)}{2} \right\rceil, \quad (32)$$

and payoff $(S_0u^{k-z}d^z)$ denotes the payoff of the polynomial option given that the maturity stock price is $S_0u^{k-z}d^z$.

Case 2. $S_0 \le \lambda_l < \lambda_u$

According to the condition $S_0 \le \lambda_l < \lambda_u$ and the definitions in equation (29), we can obtain the stock price layers c_{2l} and c_{2u} corresponding to λ_l and λ_u, respectively, and they should satisfy $0 \le c_{2l} \le c_{2u} \le n$. Similar to Case 1, we can derive three constraints for k and z as $k - c_{2u}/2 \le z \le k - c_{2l}/2$, $0 \le z \le k$, and $c_{2l} \le k \le n$. As a consequence, the option value contributed from the region $[\lambda_l, \lambda_u]$ can again be expressed as the sum of the values contributed by the paths that satisfy the above three constraints as

$$e^{-rT} \sum_{k=c_{2u}}^{n} \binom{n}{n-k} p_m^{n-k} \sum_{z=U(k,c_{2u})}^{L(k,c_{2l})} \binom{k}{z} p_u^{k-z} p_d^z \,\text{payoff}\left(S_0u^{k-z}d^z\right)$$
$$+ e^{-rT} \sum_{k=c_{2l}}^{c_{2u}-1} \binom{n}{n-k} p_m^{n-k} \sum_{z=0}^{L(k,c_{2l})} \binom{k}{z} p_u^{k-z} p_d^z \,\text{payoff}\left(S_0u^{k-z}d^z\right). \quad (33)$$

Case 3. $\lambda_l < S_0 < \lambda_u$

Since $\lambda_l < S_0 < \lambda_u$, the corresponding stock price layers c_{3l} and c_{3u} derived from equation (29) should satisfy

$-n \le c_{3l} \le 0 \le c_{3u} \le n$. Next, we simply decompose the positive-payoff region into two parts: one includes the stock price layers from c_{3l} to -1 and the other includes the stock price layers from 0 to c_{3u}. The first part can be evaluated as Case 1 by setting c_{1u} as -1 and c_{1l} as c_{3l}, and the second part can be evaluated as Case 2 by setting c_{2u} as c_{3u} and c_{2l} as 0. Finally, the option value contributed from region $[\lambda_l, \lambda_u]$ can be evaluated as the sum of the values contributed by the aforementioned two parts.

To show the efficiency of our algorithm, we next prove that Case 1 and Case 2 can be evaluated in linear time of n. Note that this implies that Case 3 can be evaluated in linear time since Case 3 can be expressed as a combination of Case 1 and Case 2. The value of a polynomial option is simply the sum of the values contributed by all positive-payoff regions and thus can be evaluated in linear time of n. Our proofs take advantage of Theorems 1 and 2 developed in Section 3.

The linear-time pricing algorithm for each positive-payoff region can be proved by showing that equations (31) and (33) can both be evaluated in linear time of n. We first show that the second summation term in equation (33) can be evaluated in linear time according to Theorem 1. The polynomial payoff of this term can be decomposed into the sum of its $(D+1)$ component terms as

$$e^{-rT}\sum_{k=c_{2l}}^{c_{2u}-1}\binom{n}{n-k}p_m^{n-k}\sum_{z=0}^{L(k,c_{2l})}\binom{k}{z}p_u^{k-z}p_d^{z}\,\text{payoff}\left(S_0u^{k-z}d^z\right) = \sum_{k=c_{2l}}^{c_{2u}-1}w_1(k)+\sum_{k=c_{2l}}^{c_{2u}-1}w_2(k)+\cdots+\sum_{k=c_{2l}}^{c_{2u}-1}w_D(k)-\sum_{k=c_{2l}}^{c_{2u}-1}\varphi(k), \tag{34}$$

where

$$\begin{aligned} w_i(k) &= e^{-rT}\binom{n}{n-k}p_m^{n-k}\sum_{z=0}^{L(k,c_{2l})}\binom{k}{z}p_u^{k-z}p_d^{z}\alpha_i\left(S_0u^{k-z}d^z\right)^{q_i} \\ &= \alpha_i S_0^{q_i}e^{-rT}\binom{n}{n-k}p_m^{n-k}\sum_{z=0}^{L(k,c_{2l})}\binom{k}{z}\left(u^{q_i}p_u\right)^{k-z}\left(d^{q_i}p_d\right)^{z}, \end{aligned} \tag{35}$$

for $1 \le i \le D$, and

$$\varphi(k) = Xe^{-rT}\binom{n}{n-k}p_m^{n-k}\sum_{z=0}^{L(k,c_{2l})}\binom{k}{z}p_u^{k-z}p_d^{z}. \tag{36}$$

Note that each of the $(D+1)$ summation terms in equation (34) can be evaluated in $O(n)$ time. Specifically, $\sum_{k=c_{2l}}^{c_{2u}-1}w_i(k)$ can be evaluated in $O(n)$ time according to Theorem 1 by setting $a = d^{q_i}p_d, b = u^{q_i}p_u, V(k) = \alpha_i S_0^{q_i}e^{-rT}\binom{n}{n-k}p_m^{n-k} = \alpha_i S_0^{q_i}e^{-rT}p_m^n\binom{n}{k}(p_m^{-1})^k$, (According to Lemma 1, $\{V(k)\} \in F$ can be obtained by setting $\beta = \alpha_i S_0^{q_i}e^{-rT}p_m^n$ and $\alpha = p_m^{-1}$) and $M(k) = L(k, c_{2l})$. In addition, $\sum_{k=c_{2l}}^{c_{2u}-1}\varphi(k)$ also can be evaluated in linear time by setting $a = p_d, b = p_u, V(k) = Xe^{-rT}\binom{n}{n-k}p_m^{n-k} = Xe^{-rT}p_m^n\binom{n}{k}(p_m^{-1})^k$, (According to Lemma 1, $\{V(k)\} \in F$ can be obtained by setting $\beta = Xe^{-rT}p_m^n$ and $\alpha = p_m^{-1}$.) and $M(k) = L(k, c_{2l})$ in Theorem 1. As a result, the second summation term in equation (33) can be evaluated in linear time of n.

For both of the first summation terms in equations (31) and (33), since the lower bounds for the index z are integer-valued functions of another running variable k rather than constants, Theorem 2 is used to prove that they can be evaluated in linear time of n. We set $M(k) = L(k, c_{il})$ and $m(k) = U(k, c_{iu})$, for $i = 1, 2$, and then mimic the method to evaluate equation (34) by specifying the appropriate terms for a, b, and $V(k)$ to evaluate each component term in the polynomial payoff in linear time with Theorem 2. Similarly, for the second summation terms in equation (31), we set $M(k) = k$, $m(k) = U(k, c_{1u})$, and a, b, and $V(k)$ to appropriate terms and next apply Theorem 2 to evaluate each component term in the polynomial payoff in linear time of n.

As a result, each of the four summation terms in equation (31) of Case 1 and equation (33) of Case 2 can be evaluated in linear time of n. Therefore, Case 3 can be evaluated in linear time of n as well. Since the option value of any polynomial option contributed from each positive-payoff region can be computed as either Case 1, 2, or 3 in linear time of n, the total value of any polynomial option can be obtained in $O(n)$ time with our combinatorial pricing algorithm. Thus our approach is more efficient than other lattice models, which price polynomial options in at least $O(n^2)$ time.

Note that the above algorithm is quite general and able to price not only polynomial options but also power and vanilla call or put options in $O(n)$ time. To the best of our knowledge, this paper is also the first in the literature to obtain $O(n)$-time combinatorial pricing algorithms for polynomial, power, and vanilla options on the trinomial lattice model.

4.1. Linear-Time Algorithm for Barrier Options on the KRL Model.

Pricing a barrier option is much more complex than pricing a polynomial option since the barrier option is a path-dependent option and can be exercised only if the stock price has hit H during the life of the option. To mitigate the oscillation problem when pricing barrier options on the trinomial lattice model, the parameter λ in equation (3) is adjusted by setting $Y = H$, such that the barrier H can coincide with a layer of the lattice, as shown in Figure 6. Before introducing our combinatorial algorithm, we first define $h = ln(H/S_0)/ln\,u$ to be the number of net upward movements needed to reach H from S_0 and also define $c = \min\{i | i \in \mathbb{Z}, \quad S_0u^i \ge X\}$ where S_0u^c can represent the lowest stock price layer which is above or coinciding with the strike price X. In the following paragraphs, our linear-time algorithm for barrier options discusses two different cases regarding the relation between the strike price X and the barrier H.

Case 4. $X \ge H$

In this case, a positive payoff (i.e., $\max(S_T - X, 0) > 0$) at maturity guarantees the hit on barrier H during the option life, so the barrier option degenerates into a vanilla option. The reason behind this phenomenon is explained as follows.

$$\begin{aligned} e^{-rT}E[\text{payoff}] &= e^{-rT}E\Big[\max(S_T - X, 0)1_{\{S_{\sup}\ge H\}}\Big] \\ &= e^{-rT}E\Big[(S_T - X)1_{\{S_T > X\}}1_{\{S_{\sup}\ge H\}}\Big] \\ &= e^{-rT}E\Big[(S_T - X)1_{\{S_T > X\}}\Big] \\ &= e^{-rT}E[\max(S_T - X, 0)] \\ &= \text{value of a vanilla call option}, \end{aligned} \tag{37}$$

where the first equality is the combination of equations (7) and (9). Since $S_{\sup} \ge S_T$ is true by definition and $X \ge H$ holds in Case 4, $S_T > X$ implies $S_{\sup} \ge H$. Thus, the second equality can be reduced to the third one. As a consequence, the barrier option in the case of $X \ge H$ can be priced by the algorithm for the vanilla option.

Case 5. $X < H$

In this case, the option value can be further decomposed into two parts: the value contributed by the terminal nodes above H (inclusive) and the value contributed by the terminal nodes between H (exclusive) and X (The value contributed by the nodes below X is not taken into account since the option payoff at these nodes is 0.).

First, consider a terminal node, say A in Figure 6, above or equal to the barrier H. Note that all price paths reaching node A must also hit the barrier H. Thus, the payoff for this kind of node is the same as the payoff for the vanilla call, i.e., $\max(S_T - X, 0)$. Furthermore, since $X < H \le S_T$ in this case, the payoff for nodes above H can be simplified to be $S_T - X$. In addition, for those paths reaching a node above H (inclusive), we have $S_0u^{k-2z} \ge S_0u^h$, which implies

$$k - 2z \ge h \Rightarrow z \le \frac{k-h}{2} \Rightarrow z \le \left\lfloor \frac{k-h}{2} \right\rfloor = L(k,h), \tag{38}$$

where the third inequality is due to the fact that z is an integer and the last equality is according to the definition in equation (31). In addition, since the number of downward movements z is a nonnegative integer, the number of the sum of the upward and downward movements k should satisfy

$$k = x + z \ge (z + h) + z \ge h. \tag{39}$$

Thereby, the option value contributed by these terminal nodes is

$$e^{-rT}\sum_{k=h}^{n}\binom{n}{n-k}p_m^{n-k}\sum_{z=0}^{L(k,h)}\binom{k}{z}p_u^{k-z}p_d^{z}\left(S_0u^{k-2z} - X\right), \tag{40}$$

where the lower bound of the first summation is from equation (39) and the upper bound of the second summation is from equation (38).

Next, we employ Corollary 1 to develop an $O(n)$-time algorithm for evaluating equation (40), which can be rewritten as

$$\sum_{k=h}^{n}\omega(k) - \sum_{k=h}^{n}\varphi(k), \tag{41}$$

where

$$\begin{aligned} \omega(k) &= S_0e^{-rT}\binom{n}{n-k}p_m^{n-k}\sum_{z=0}^{L(k,h)}\binom{k}{z}(up_u)^{k-z}(dp_d)^{z}, \\ \varphi(k) &= Xe^{-rT}\binom{n}{n-k}p_m^{n-k}\sum_{z=0}^{L(k,h)}\binom{k}{z}p_u^{k-z}p_d^{z}. \end{aligned} \tag{42}$$

The summation term, $\sum_{k=c}^{n}\omega(k)$, can be evaluated in $O(n)$ time through Corollary 2 by setting $a = dp_d, b = up_u$, $V(k) = S_0e^{-rT}\binom{n}{n-k}p_m^{n-k} = S_0e^{-rT}p_m^n\binom{n}{k}(p_m^{-1})^k$, and $M(k) = L(k,h)$, where $\{V(k)\}$ can be proved to belong to F through Lemma 1 by setting $\beta = S_0e^{-rT}p_m^n$ and $\alpha = p_m^{-1}$. Similarly, the summation term, $\sum_{k=c}^{n}\varphi(k)$, can be evaluated in O(n) time through Corollary 2 by setting $a = p_d$, $b = p_u, V(k) = Xe^{-rT}\binom{n}{n-k}p_m^{n-k} = Xe^{-rT}p_m^n\binom{n}{k}(p_m^{-1})^k$, and $M(k) = L(k,h)$. Note that $\{V(k)\} \in F$ can be proved using Lemma 1 by setting $\beta = Xe^{-rT}p_m^n$ and $\alpha = p_m^{-1}$.

Second, consider terminal nodes that are below the barrier H (exclusive) but above or equal to X, like node B in Figure 6. For those terminal nodes, they should satisfy $S_0u^c \le S_0u^{k-2z} < S_0u^h$. Therefore, a pair of upper and lower bounds for z can be derived as

$$\begin{aligned} &c \le k - 2z < h \Rightarrow \frac{k-c}{2} \ge z > \frac{k-h}{2} \Rightarrow \left\lfloor \frac{k-c}{2} \right\rfloor \ge z \ge \left\lfloor \frac{k-h}{2} \right\rfloor \\ &\quad + 1 \Rightarrow L(k,c) \ge z \ge L(k,h) + 1. \end{aligned} \tag{43}$$

In addition, for a price path that first reaches the barrier H (which takes h upward movements) and then falls below the barrier at maturity (which takes at least 1 downward movement), the sum of the number of downward movements and upward movements, k, should satisfy

$$k \ge h + 1. \tag{44}$$

Moreover, in order to hit H during the option life, the number of upward movements, x, must be larger than h, which suggests another constraint for z:

$$x \ge h \Rightarrow k - z \ge h \Rightarrow z \le k - h. \tag{45}$$

Finally, the number of paths, which starts from the root node, hits the barrier H during the option life, and reaches the terminal node below the barrier H that can be computed by the reflection principle mentioned in equation (13). Note that in Figure 6, the root of the lattice is h step lower than H (i.e., $a = h$ in Figure 5) and the vertical distance between the terminal node and the root is $k - 2z$; thus, the y-coordinate for node B is $-h + (k - 2z)$, which implies $b = h - (k - 2z)$ in Figure 5. The number of the paths that satisfy the aforementioned constraints can be obtained by substituting

a and b into equation (13) to get $\binom{n}{n-k}\binom{k}{k-z-h}$. Thus, the option value contributed by the terminal nodes that are below the barrier H but above or equal to X is

$$e^{-rT}\sum_{k=h+1}^{n}\binom{n}{n-k}p_m^{n-k}\sum_{z=L(k,h)+1}^{\min(k-h,L(k,c))}\binom{k}{k-z-h}p_u^{k-z}p_d^{z}\left(S_0u^{k-z}d^{z}-X\right), \tag{46}$$

where the lower bound of the first summation is from equation (44) and the lower and upper bounds of the second summation are from equations (43 and 45), respectively.

To evaluate equation (46) in O(n) time, the first step is to substitute j for $k-z-h$, and then, we can rewrite equation (46) by employing j as the running variable for the inner summation. Because

$$\begin{aligned} &L(k,h)+1\le z\le \min(k-h,L(k,c))\\ &\quad\Rightarrow \max(h-k,-L(k,c))\le -z\le -L(k,h)-1\\ &\quad\Rightarrow k-h+\max(h-k,-L(k,c))\\ &\quad\le k-z-h\le k-h-L(k,h)-1\\ &\quad\Rightarrow (k-h-L(k,c))^{+}\le j\le U(k,h)-1, \end{aligned} \tag{47}$$

where $U(k,h)=\lceil (k-h)/2\rceil$, equation (46) can be rewritten as

$$\begin{aligned} &e^{-rT}(up_u)^h(dp_d)^{-h}\sum_{k=h+1}^{n}\binom{n}{n-k}p_m^{n-k}\sum_{j=(k-h-L(k,c))^+}^{U(k,h)-1}\binom{k}{j}p_u^jp_d^{k-j}S_0u^jd^{k-j}\\ &\quad-e^{-rT}p_u^hp_d^{-h}\sum_{k=h+1}^{n}\binom{n}{n-k}p_m^{n-k}\sum_{j=(k-h-L(k,c))^+}^{U(k,h)-1}\binom{k}{j}p_u^jp_d^{k-j}X\\ &\quad\equiv\sum_{k=h+1}^{n}\omega(k)-\sum_{k=h+1}^{n}\varphi(k), \end{aligned} \tag{48}$$

where

$$\begin{aligned} \omega(k)&=S_0e^{-rT}(up_u)^h(dp_d)^{-h}\binom{n}{n-k}p_m^{n-k}\sum_{j=(k-h-L(k,c))^+}^{U(k,h)-1}\binom{k}{j}(up_u)^j(dp_d)^{k-j},\\ \varphi(k)&=Xe^{-rT}p_u^hp_d^{-h}\binom{n}{n-k}p_m^{n-k}\sum_{j=(k-h-L(k,c))^+}^{U(k,h)-1}\binom{k}{j}p_u^jp_d^{k-j}. \end{aligned} \tag{49}$$

The first term in equation (48), $\sum_{k=h+1}^{n}\omega(k)$, can be evaluated in $O(n)$ time through Theorem 2 by setting $a=up_u$, $b=dp_d$,$V(k)=S_0\,e^{-rT}(up_u)^h(dp_d)^{-h}\binom{n}{n-k}p_m^{n-k}$, $M(k)=U(k,\ h)-1$, and $m(k)=(k-h-L(k,c))^+$. Note that $\{V(k)\}\in F$ can be proved via Lemma 1 by rewriting $V(k)=S_0\,e^{-rT}(up_u)^h(dp_d)^{-h}p_m^n\binom{n}{k}(p_m^{-1})^k$ and setting $\beta=S_0e^{-rT}(up_u)^h(dp_d)^{-h}p_m^n$ and $\alpha=p_m^{-1}$.

Similarly, the second term $\sum_{k=h+1}^{n}\varphi(k)$ can be evaluated in $O(n)$ time through Theorem 2 by setting $a=p_u$, $b=p_d$, $V(k)=Xe^{-rT}p_u^hp_d^{-h}\binom{n}{n-k}p_m^{n-k}$, $M(k)=U(k,h)-1$, and $m(k)=(k-h-L(k,c))^+$. Again, $\{V(k)\}\in F$ can be proved by Lemma 1 by rewriting $V(k)=Xe^{-rT}p_u^hp_d^{-h}p_m^n\binom{n}{k}(p_m^{-1})^k$ and setting $\beta=Xe^{-rT}p_u^hp_d^{-h}p_m^n$ and $\alpha=p_m^{-1}$. Thereby, equations (48) and (46) can be evaluated in O(n) time.

The price of the barrier option of Case 5 is the sum of equations (40) and (46). We hereby propose an $O(n)$-time algorithm to price barrier options on the KRL model.

5. Experimental Results

In this section, we compare our linear-time combinatorial algorithms with the standard backward induction procedure (which is indeed the dynamic programming approach mentioned above) for pricing polynomial, vanilla, and barrier options on the KRL trinomial lattice. The results demonstrate the superiority of our combinatorial model in terms of efficiency and accuracy.

Since there is no simple closed-form pricing formula for polynomial options, it is necessary to price them numerically. However, pricing polynomial options via dynamic programming suffer from slow convergence, and thus, a

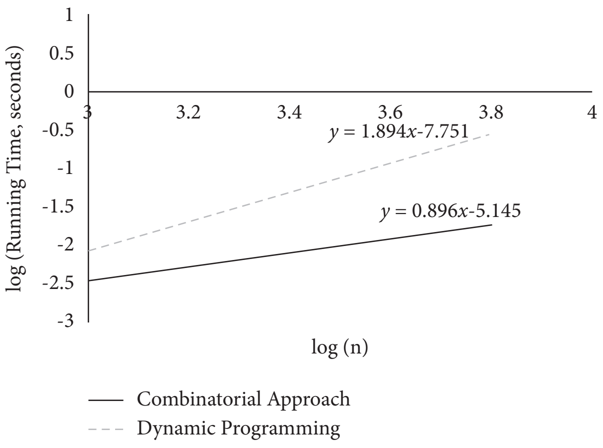


Figure 7: Time complexity for pricing polynomial options. This figure is a log-log plot of the running time versus the number of time steps. The solid and dashed lines denote the combinatorial and the dynamic programming approaches, respectively. The linear regression formulae are listed next to the lines. The initial stock price is 5, the exercise price is 5, the risk-free rate is 10% per annum, the volatility of the stock price is 25%, and the time to maturity (T) is one year. The payoff of the examined polynomial option is $\max(S_T^4 - 22S_T^3 + 179S_T^2 - 638S_T + 845 - X, 0)$. The results demonstrate that our combinatorial approach successfully reduces the time complexity for pricing polynomial options on the trinomial lattice to $O(n)$ time, comparing to the $O(n^2)$-time complexity of the dynamic programming approach.

large number of n is required. Figure 7 shows the comparison of the running time between our $O(n)$-time trinomial combinatorial approach and the $O(n^2)$-time dynamic programming approach. The x and y-axes denote the logarithmic values of n and running time, respectively. The slopes of linear regression formulae denote the estimations for the order of time complexities. Note that the combinatorial approach (slope = 0.896) is much more efficient than the dynamic programming approach (slope = 1.894). Moreover, we compare the convergence rates of absolute pricing errors for binomial and trinomial lattices, as shown in Figure 8. Our trinomial lattice generates smaller absolute pricing errors and shows a faster convergence rate than the CRR binomial lattice for pricing polynomial options. To further confirm the stable and fast convergence of our combinatorial algorithm, Figure 9 shows comprehensive sensitivity analyses for the polynomial option analyzed in Figure 7. The absolute pricing errors for the combinatorial approaches based on both the binomial and trinomial lattices are much smaller than the dynamic programming approach on the trinomial lattice and the explicit finite difference method (The implementation issues about applying the finite difference and Fourier transform methods to price polynomial options are discussed in Appendix.) for different settings of the risk-free rates r, stock price volatilities σ, times to maturity T, exercise prices X, and payoff functions. In addition, the combinatorial approach based on the trinomial lattice performs better than that on the binomial one in terms of less oscillations, as shown in Figure 10.

A vanilla option is a special case of a polynomial option, and its value can be analytically evaluated by the Black–Scholes formula [36]. With this property, we can analyze the pricing errors generated by our combinatorial algorithm. Figure 11 shows the comparison of the absolute pricing errors and running times between the dynamic

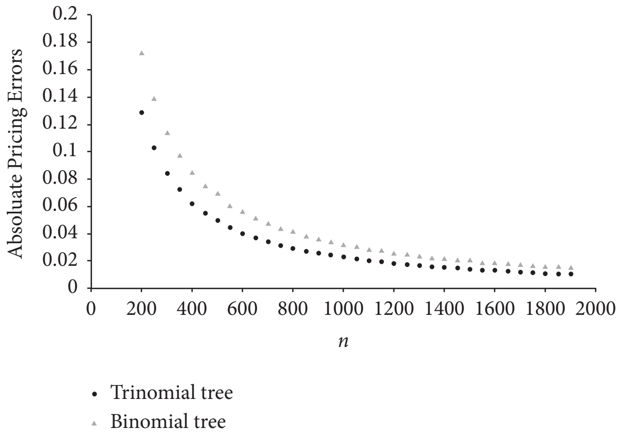
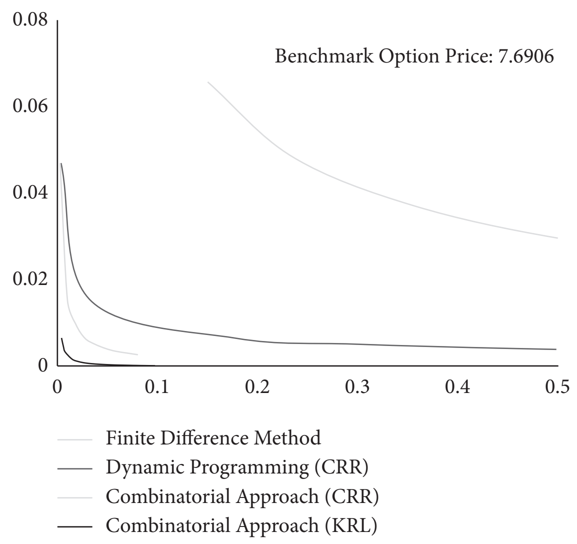


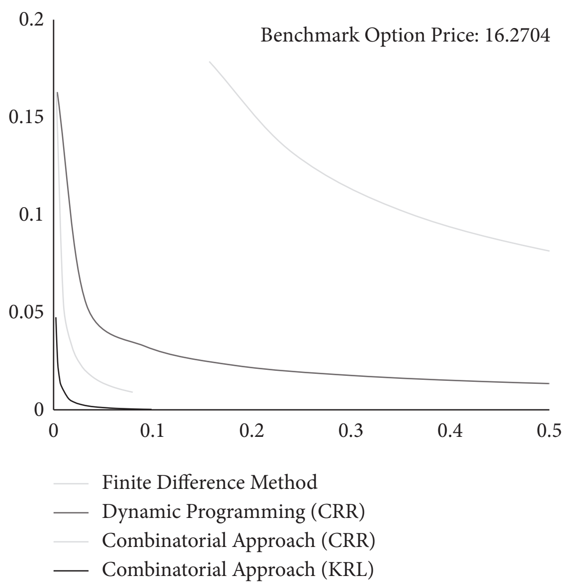
Figure 8: Convergence of absolute pricing errors for pricing polynomial options. The black dots indicate the pricing errors generated by our trinomial lattice, and the gray triangles represent the pricing errors of the CRR binomial lattice. The benchmark option price, 10.8856, is generated by the CRR binomial lattice with a very large $n = 10{,}000$. All parameters are identical to those in Figure 7. The trinomial lattice adopted in this study converges faster than the CRR binomial lattice does for pricing polynomial options.

programming approach and the combinatorial approach for pricing vanilla call options. The x and y-axes denote the running time and absolute pricing errors, respectively. The results show that the absolute pricing errors of the combinatorial approach decrease more smoothly and rapidly than the dynamic programming approach.

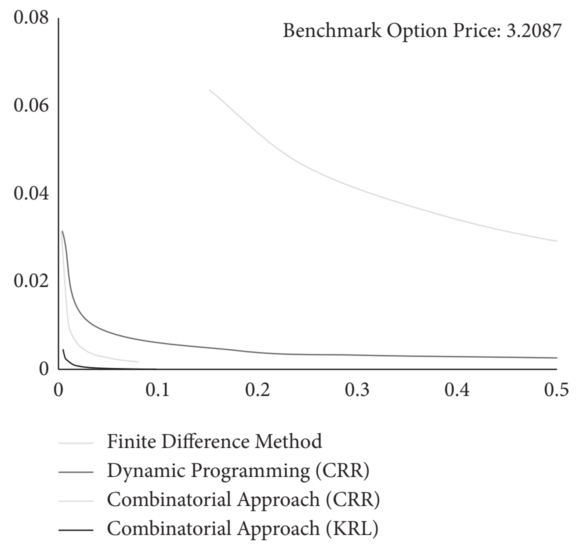
Figure 12 shows the relation between the running time and the number of time steps n for our combinatorial and dynamic programming approaches on pricing up-and-in barrier call options. The x and y-axes denote the values after taking the logarithm of the number of time steps and



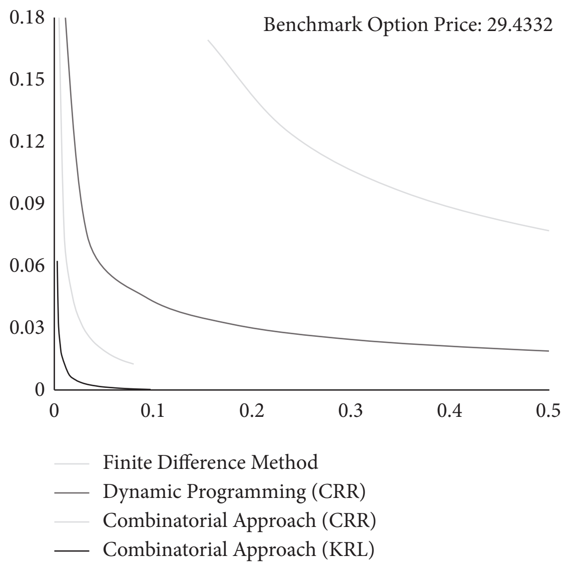
(a)



(b)

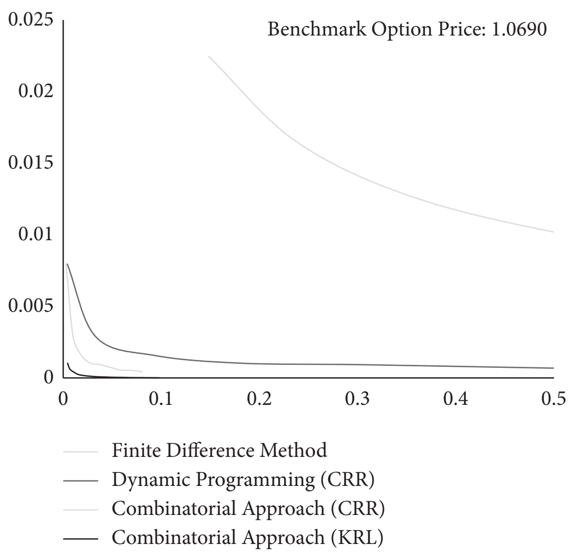


(c)

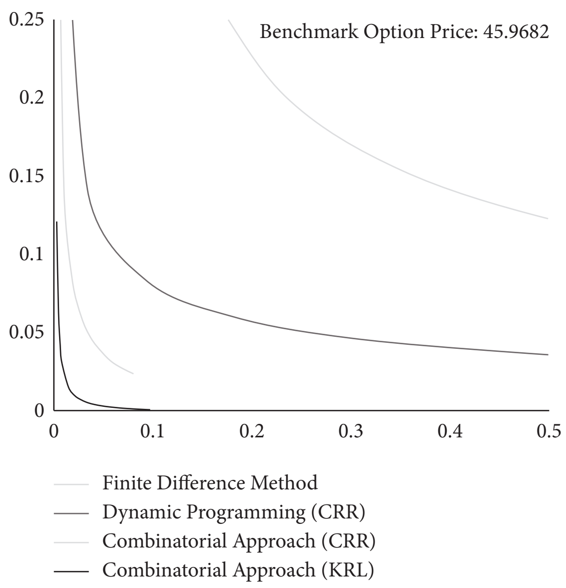


(d)

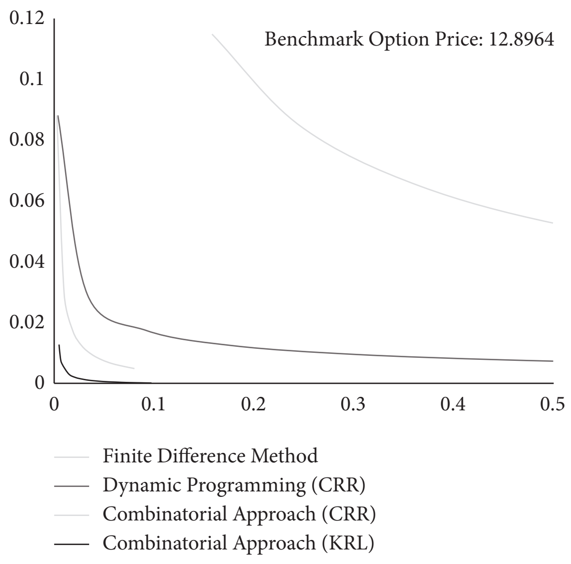
Figure 9: Continued.



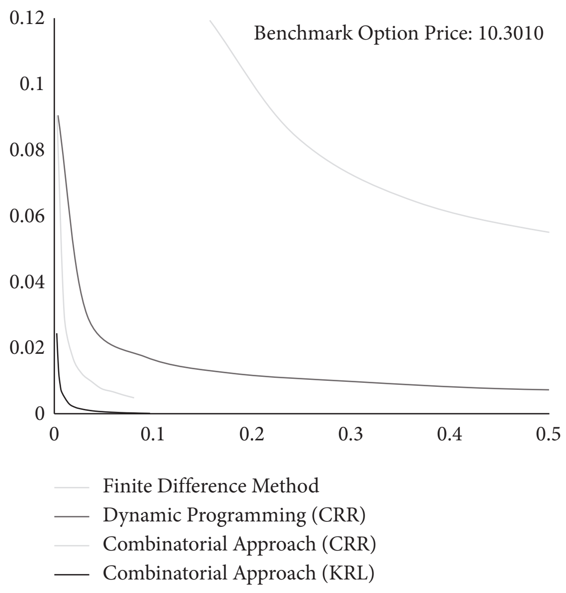
(e)



(f)



(g)



(h)

FIGURE 9: Sensitivity analyses for polynomial options. The x and y-axes represent the running time and the absolute pricing error, respectively. The black and light gray thin curves denote the pricing errors generated by the combinatorial algorithms based on the KRL trinomial and CRR binomial lattice, respectively. The dark gray and light gray thick curves indicate the pricing errors generated by the dynamic programming approach on the CRR binomial lattice and explicit finite difference method. All parameters in (a)–(h) are identical to those in Figure 7 except the parameters specified in the legend. Specifically, the risk-free rates (r) in (a) and (b) are set to 5% and 15%; the volatilities of the stock price σ in (c) and (d) is 20% and 30%; the times to maturity (T) in (e) and (f) are 0.5 and 1.5 years; the exercise prices (X) in (g) and (h) are 2.5 and 7.5, respectively. The benchmark option value for each parameter setting is listed on the top right corner of each subfigure. (a) $r = 5\%$. (b) $r = 15\%$. (c) $\sigma = 20\%$. (d) $\sigma = 30\%$. (e) $T = 0.5$. (f) $T = 1.5$. (g) $X = 2.5$. (h) $X = 7.5$.

for the corresponding running time, respectively. The regression formula for the dynamic programming approach is $y = 2.039x - 7.591$, whereas the regression formula for the combinatorial approach is $y = 0.971x - 5.938$. The slope of the dynamic programming approach is twice as high as that of the combinatorial one. These results again demonstrate that we successfully reduce the time complexity of pricing barrier options on the trinomial lattice from $O(n^2)$ by the dynamic programming approach to $O(n)$ by this novel trinomial combinatorial approach. Figure 13 shows the comparison of the absolute pricing errors versus running time for pricing up-and-in barrier call options between the binomial combinatorial approach and the trinomial combinatorial and dynamic programming approaches based on the KRL trinomial lattice. The binomial combinatorial approach is the linear-time

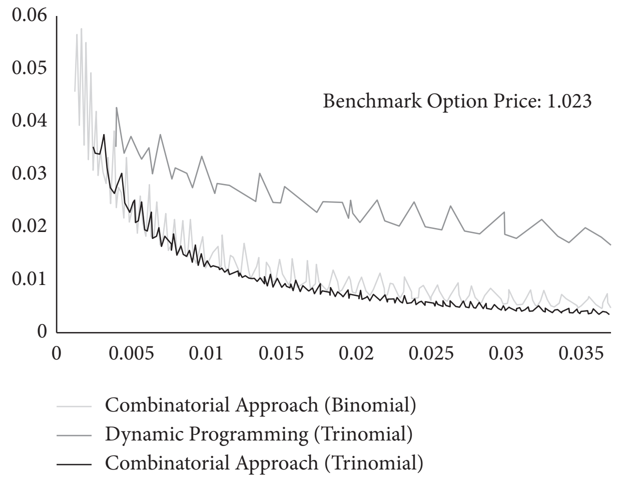


Figure 10: Convergence oscillations for pricing polynomial options. The x-axis and y-axis represent the running time and the absolute pricing error, respectively. The black and light-gray curves denote the pricing errors generated by the combinatorial algorithms based on the KRL trinomial and CRR binomial lattice, respectively. The dark-gray thick curves indicate the pricing errors generated by the dynamic programming approach on the KRL trinomial lattice. The payoff function for the polynomial option is specified as $\max((S_T - 0.7X)(S_T - 1.3X) - 5000, 0)$, where the initial stock price S0, the exercise price X, the risk-free rate r, the time to maturity T, and the volatility of the stock price σ are set to 50, 50, 10%, 0.5 years, and 40%, respectively. The benchmark option value for each parameter setting is listed on the top right corner of the figure.

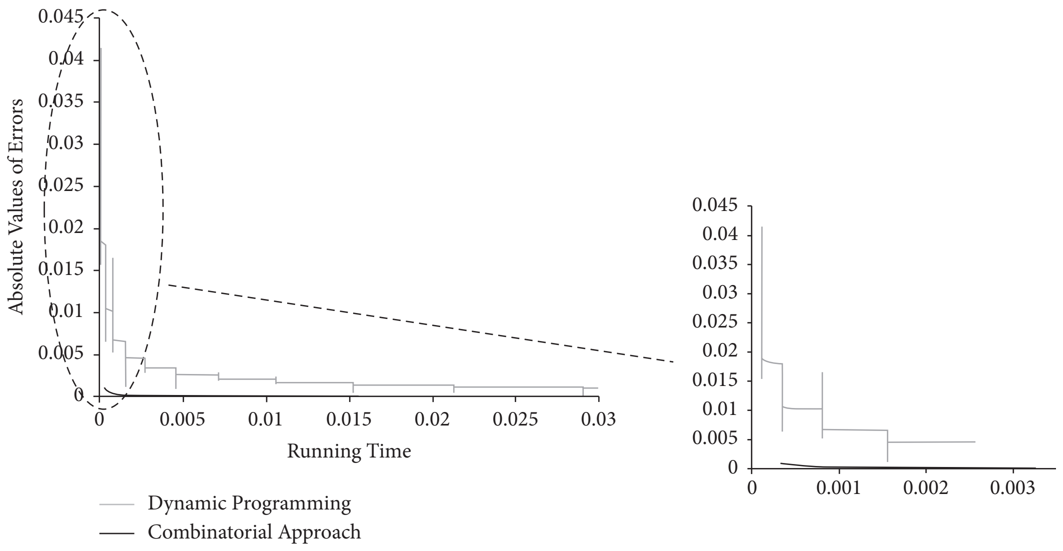


Figure 11: Comparison of absolute pricing error versus running time for pricing vanilla call options. For the examined vanilla call option, the initial stock price is 90, the exercise price is 100, the risk-free rate is 10% per annum, the volatility of the stock price is 25%, and the time to maturity is 1 year. The benchmark option price is obtained based on Black and Scholes [36]. The black line indicates the absolute pricing errors of our combinatorial approach, and the gray line represents the absolute pricing errors of the dynamic programming approach. The right-hand side diagram is a more detailed illustration for the running time shorter than 0.003 seconds. Compared with the dynamic programming approach, our combinatorial approach shows more rapidly decreasing errors and smoother convergence behavior.

pricing algorithm developed in Dai et al. [3] based on the CRR binomial lattice. Even equipped with this efficient linear-time binomial combinatorial approach, the oscillation problem is clearly more pronounced than the other two trinomial lattice-based models. Moreover, the trinomial combinatorial approach converges faster and more smoothly than the trinomial lattice dynamic programming approach and the binomial combinatorial approach in terms of running time.

Note that the analyses in Figure 2 demonstrate that the pricing results of the binomial combinatorial approach are more inclined to oscillate and less accurate than those of the trinomial combinatorial approach in terms of both the number of time steps and running time. These comparisons

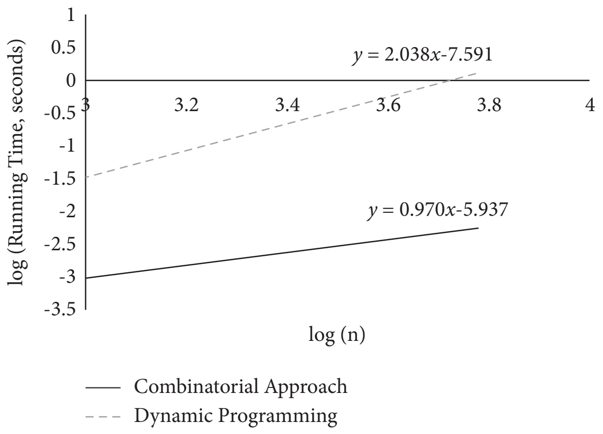


Figure 12: Time complexity comparison of pricing up-and-in barrier call options. This figure represents the log-log plot of the running time versus the number of time steps. The solid and dashed lines denote the results for the combinatorial approach and the dynamic programming approach, respectively. The linear regression formulae are listed next to the lines. The initial stock price is 90, the exercise price is 85, the higher barrier is 95, the risk-free rate is 10% per annum, the volatility of the stock price is 25%, and the time to maturity is 1 year. The results demonstrate that our combinatorial approach successfully reduces the time complexity for pricing barrier options on the trinomial lattice to $O(n)$ time, whereas the time complexity of the dynamic programming approach is $O(n^2)$.

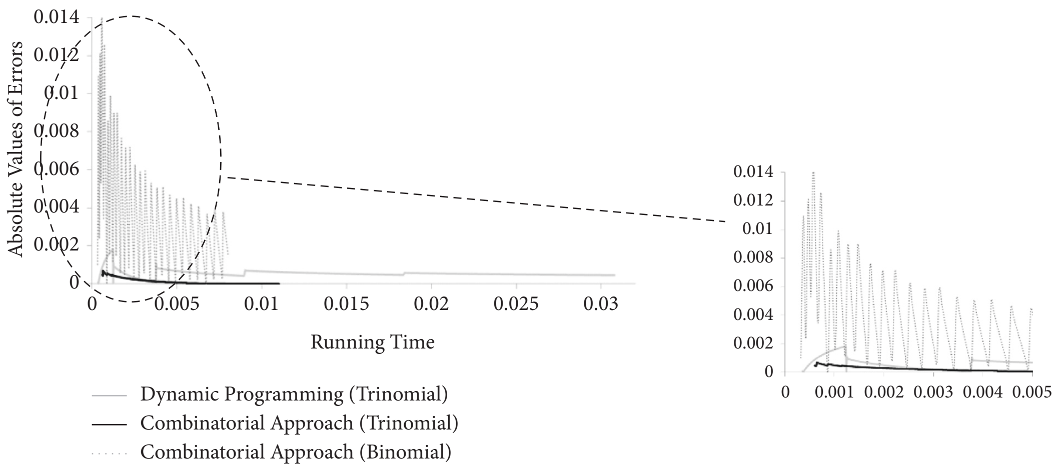


Figure 13: Comparison of absolute pricing error versus running time for pricing up-and-in barrier call options. The black and gray solid lines indicate the absolute errors of our combinatorial approach and the dynamic programming approach based on the KRL trinomial lattice. The dotted line represents the absolute errors of the linear-time binomial lattice combinatorial approach proposed in the study by Dai et al. [3]. The benchmark option price is derived based on Rich [20] and Rubinstein and Reiner [21]. All parameters are the same as those in Figure 12. The right-hand side diagram is a more detailed illustration for the running time shorter than 0.005 seconds. The oscillation problem of the binomial combinatorial approach is more pronounced than those of the other two trinomial lattice-based models. Moreover, our trinomial combinatorial approach shows most rapidly decreasing errors and smoothest convergence behavior among the three models.

between binomial and trinomial combinatorial approaches support the necessity and contribution of proposing combinatorial option pricing approaches based on the trinomial lattice in this study.

6. Conclusion

Although the trinomial lattice framework for option pricing has been shown to provide better convergence behavior than the binomial lattice framework, only the dynamic programming approach, whose complexity is at least $O(n^2)$ time, has been developed under the trinomial lattice framework to price options. This study proposes a novel combinatorial approach to reduce the time complexity for pricing options on the trinomial lattice of Kamrad and Ritchken [5]. The notion is to decompose the complex option payoff functions into certain regular summation forms, which can be evaluated in $O(n)$ time by simple recursive formulae. We successfully develop $O(n)$-time pricing algorithms for a wide range of options which require at least $O(n^2)$ time when priced by the original dynamic programming approach.

Appendix

A. Issues about Pricing Polynomial Options with the Finite Difference and Fourier Transform Methods

Note that for vanilla options, since they are deeply in- and out-of-the-money at the two ends of the examined stock price range, well-performing boundary conditions can be determined based on this property for the finite difference method (abbreviated as the FD method). However, the property may not hold for complex polynomial payoff functions because the in- and out-of-the-money regions may distribute alternately. Therefore, in the FD methods, it is a challenging task to set the boundary conditions for pricing polynomial options. Recall that the payoff of a European call option is $(S_T - X)^+$, where S_T and X denote the price of the underlying asset at maturity and the strike price, respectively. Thus, the option on the boundary with very high underlying asset value S_t at time t is very likely to be in-the-money at maturity; therefore, the option value on the boundary can be trivially solved as $e^{-r(T-t)}E[S_T - X | S_t]$ by dropping the "+" operator. However, taking advantage of this in-the-money property may lead to pricing errors for complex polynomial option payoff functions. Feasible settings for the boundary position and corresponding option values are also nontrivial. In Figure 9, we set the upper bound to a high stock price level and examine a considerable number of stock prices to ensure the convergence of option values. Perhaps, this is why the FD method converges slightly slower than the dynamic programming method shown in Figure 9.

On the other hand, pricing polynomial options with the Fourier transform method (abbreviated as the FT method) could be an interesting but challenging future work due to complex payoff functions and in-the-money regions. This may explain why there are no FT-based pricing methods for polynomial options to our knowledge. We sketch the FT derivations proposed in Carr and Madan [38] for pricing a European-style vanilla call to illustrate the challenge. They price a European call option with payoff $(S_T - X)^+$. In this study, they argue that the vanilla call option price $C_T(x)$ can be expressed as an inverse Fourier transform

$$C_T(v) = \frac{e^{-\alpha x}}{2\pi} \int_{-\infty}^{\infty} e^{ivx} \psi_T(v) \mathrm{d}v, \tag{A.1}$$

where $x \equiv ln(X)$ and $\psi_T(v)$ denotes the Fourier transform of the call price multiplied by $e^{\alpha x}$, defined as

$$\psi_T(v) = \frac{e^{-\alpha x}}{2\pi} \int_{-\infty}^{\infty} e^{ivx} \int_{x}^{\infty} e^{\alpha x} e^{-rT} \left(e^{s} - e^{x}\right) q_T(s) \mathrm{d}s \mathrm{d}x, \tag{A.2}$$

where $s \equiv ln(S_T)$ and q_T denote the density function of S_T. One can change the integration order of equation (A.2) to yield

$$\int_{-\infty}^{\infty} e^{-rT} q_T(s) \int_{-\infty}^{s} \left(e^{s+\alpha x} - e^{(1+\alpha)x}\right) e^{ivx} \mathrm{d}x \mathrm{d}s, \tag{A.3}$$

and then analytically evaluate the integration for later substitution into equation (A.1) to solve the option price. Now, we replace the payoff function of a vanilla option $(S_T - X)^+$ with

$$\left(\sum_{i=1}^{n} b_i S_T^i - b_0\right)^+, \tag{A.4}$$

the payoff function of a polynomial option. Note that b_0 is analogous to the strike price X; thus, directly applying the above approach requires a positive b_0 to ensure that x $(\equiv ln(X))$ is defined. However, a nonpositive b_0 does not cause any problem in the lattice-based approach. In addition, the ranges defined by the lower and upper limits of the inner integrals like $x \le s \le \infty$ and $-\infty \le x \le s$ in equations (A.2) and (A.3) reflect the in-the-money region of the vanilla option. Multiple in-the-money regions of polynomial options could make the change of the integration order and analytical integration of equation (A.3) intractable. This is because, on the s-x plane, the contours of in-the-money regions are disconnected and nonpiecewise linear. In addition, on pages 587–588 of the study by Kwok et al. [37], we find that the singularity of the integration function for pricing vanilla options arises from the kink in the payoff function. Carr and Madan [38] solved this by multiplying the payoff with an exponential decay function $e^{-\alpha x}$. However, more singularities must be circumvented for pricing polynomial options due to the multiple roots of equation (A.4). These reasons all show why developing a fast convergent FT pricing method is a challenging task.

Data Availability

No data were used to support this study.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

References

[1] J. Hull, *Options, Futures, and Other Derivatives*Prentice-Hall Press, Englewood Cliffs, NJ, USA, 2014.

[2] S. Figlewski and B. Gao, "The adaptive mesh model: a new approach to efficient option pricing," *Journal of Financial Economics*, vol. 53, no. 3, pp. 313–351, 1999.

[3] T.-S. Dai, L.-M. Liu, and Y.-D. Lyuu, "Linear-time option pricing algorithms by combinatorics," *Computers & Mathematics with Applications*, vol. 55, no. 9, pp. 2142–2157, 2008.

[4] J. C. Cox, S. A. Ross, and M. Rubinstein, "Option pricing: a simplified approach," *Journal of Financial Economics*, vol. 7, no. 3, pp. 229–263, 1979.

[5] B. Kamrad and P. Ritchken, "Multinomial approximating models for options with k state variables," *Management Science*, vol. 37, no. 12, pp. 1640–1652, 1991.

[6] Q. Shang and B. Byrne, "American option pricing: an accelerated lattice model with intelligent lattice search," *Journal of Derivatives*, vol. 27, no. 1, pp. 92–108, 2019.

[7] T.-S. Dai, S. S. Yang, and L.-C. Liu, "Pricing guaranteed minimum/lifetime withdrawal benefits with various provisions under investment, interest rate and mortality risks,"

Insurance: Mathematics and Economics, vol. 64, pp. 364–379, 2015.

[8] L. Goudenege, A. Molent, and A. Zanette, "Pricing and hedging GMWB in the Heston and in the Black-Scholes with stochastic interest rate models," *Computational Management Science*, vol. 16, no. 1-2, pp. 217–248, 2019.

[9] C.-J. Wang and T.-S. Dai, "An accurate lattice model for pricing catastrophe equity put under the jump-diffusion process," *IEEE Computational Intelligence Magazine*, vol. 13, no. 4, pp. 35–45, 2018.

[10] L. Sun and M. Widdicks, "Why do employees like to be paid with Options?: a multi-period prospect theory approach," *Journal of Corporate Finance*, vol. 38, pp. 106–125, 2016.

[11] Z. Ahmadi, S. M. Hosseini, and A. F. Bastani, "A new lattice-based scheme for swing option pricing under mean-reverting regime-switching jump-diffusion processes," *Journal of Computational and Applied Mathematics*, vol. 383, Article ID 113132, 2021.

[12] C.-J. Wang, T.-S. Dai, and Y.-D. Lyuu, "Evaluating corporate bonds with complicated liability structures and bond provisions," *European Journal of Operational Research*, vol. 237, no. 2, pp. 749–757, 2014.

[13] L.-C. Liu, T.-S. Dai, and C.-J. Wang, "Evaluating corporate bonds and analyzing claim holders' decisions with complex debt structure," *Journal of Banking & Finance*, vol. 72, pp. 151–174, 2016.

[14] X. Zhuo, G. Xu, and H. Zhang, "A simple trinomial lattice approach for the skew-extended CIR models," *Mathematics and Financial Economics*, vol. 11, no. 4, pp. 499–526, 2017.

[15] Z. Ahmadi, S. M. Hosseini, and A. F. Bastani, "A lattice-based approach to option and bond valuation under mean-reverting regime-switching diffusion processes," *Journal of Computational and Applied Mathematics*, vol. 363, pp. 156–170, 2020.

[16] M. Rubinstein, "On the relation between binomial and trinomial option pricing models," *Journal of Derivatives*, vol. 8, no. 2, pp. 47–50, 2000.

[17] Y.-D. Lyuu, "Very fast algorithms for barrier option pricing and the ballot problem," *Journal of Derivatives*, vol. 5, no. 3, pp. 68–79, 1998.

[18] E. Omberg, "A note on the convergence of binomial-pricing and compound-option models," *The Journal of Finance*, vol. 42, no. 2, pp. 463–469, 1987.

[19] P. Ritchken, "On pricing barrier options," *Journal of Derivatives*, vol. 3, no. 3, pp. 19–28, 1995.

[20] D. Rich, "The mathematical foundations of barrier option pricing theory," *Advances in Futures and Options Research*, vol. 7, pp. 267–311, 1994.

[21] M. Rubinstein and E. Reiner, "Breaking down the barriers," *Risk*, vol. 4, no. 8, pp. 28–35, 1991.

[22] Y. S. Tian, "A flexible binomial option pricing model," *Journal of Futures Markets*, vol. 19, no. 7, pp. 817–843, 1999.

[23] T. Klassen, "Simple, fast and flexible pricing of Asian options," *Journal of Computational Finance*, vol. 4, no. 3, pp. 89–124, 2001.

[24] P. P. Boyle and S. H. Lau, "Bumping up against the barrier with the binomial method," *Journal of Derivatives*, vol. 1, no. 4, pp. 6–14, 1994.

[25] G. Fusai, G. Germano, and D. Marazzina, "Spitzer identity, Wiener-Hopf factorization and pricing of discretely monitored exotic options," *European Journal of Operational Research*, vol. 251, no. 1, pp. 124–134, 2016.

[26] C. E. Phelan, D. Marazzina, G. Fusai, and G. Germano, "Hilbert transform, spectral filters and option pricing," *Annals of Operations Research*, vol. 282, no. 1, pp. 273–298, 2019.

[27] C. E. Phelan, D. Marazzina, G. Fusai, and G. Germano, "Fluctuation identities with continuous monitoring and their application to the pricing of barrier options," *European Journal of Operational Research*, vol. 271, no. 1, pp. 210–223, 2018.

[28] C. E. Phelan, D. Marazzina, and G. Germano, "Pricing methods for α-quantile and perpetual early exercise options based on Spitzer identities," *Quantitative Finance*, vol. 20, no. 6, pp. 899–918, 2020.

[29] G. Fusai, D. Marazzina, and M. Marena, "Pricing discretely monitored Asian options by maturity randomization," *SIAM Journal on Financial Mathematics*, vol. 2, no. 1, pp. 383–403, 2011.

[30] J. Talponen and M. Turunen, "Option pricing: a yet simpler approach," *Decisions in Economics and Finance*, pp. 1–25, 2021.

[31] S. Macovschi and F. Quitard-Pinon, "On the pricing of power and other polynomial options," *Journal of Derivatives*, vol. 13, no. 7, pp. 61–71, 2006.

[32] K. Ravindran, *Exotic Options: The Basic Building Blocks and Their Applications," the Handbook Of Exotic Options: Instruments, Analysis, and Applications*, McGraw-Hill, USA, First edition, 1995.

[33] J. Kim, B. Kim, K. S. Moon, and I. S. Wee, "Valuation of power options under Heston's stochastic volatility model," *Journal of Economic Dynamics and Control*, vol. 36, no. 2, pp. 1796–1813, 2012.

[34] Z. Zhang, W. Liu, and Y. Sheng, "Valuation of power option for uncertain financial market," *Applied Mathematics and Computation*, vol. 286, pp. 257–264, 2016.

[35] J. Hull and A. White, "Efficient procedures for valuing European and American path-dependent options," *Journal of Derivatives*, vol. 1, no. 1, pp. 25–40, 1993.

[36] F. Black and M. Scholes, "The pricing of options and corporate liabilities," *Journal of Political Economy*, vol. 81, no. 3, pp. 646–659, 1973.

[37] Y. K. Kwok, K. S. Leung, and H. Y. Wong, "Efficient options pricing using the fast fourier transform," in *Handbook of Computational Finance*, pp. 579–604, Springer, Berlin, Heidelberg, 2012.

[38] P. Carr and D. Madan, "Option valuation using the fast Fourier transform," *Journal of Computational Finance*, vol. 2, no. 4, pp. 61–73, 1999.