A Parallel Algorithm to Price Asian Options with Multi-dimensional Assets

by

Kai Huang

A dissertation submitted in partial fulfillment of the
requirements for the degree of

Master of Science

Department of Computer Science

Faculty of Graduate Studies
University of Manitoba

THE UNIVERSITY OF MANITOBA

FACULTY OF GRADUATE STUDIES

*****

COPYRIGHT PERMISSION

A Parallel Algorithm to Price Asian Options with Multidimensional Assets

BY

Kai Huang

A Thesis/Practicum submitted to the Faculty of Graduate Studies of The University of Manitoba in partial fulfillment of the requirement of the degree

Of

Master of Science

This thesis is dedicated to my parents and my husband.

Abstract

Options are now traded actively on many exchanges across the world. Especially, some complex options such as multi-dimensional Asian options are much more widely used in the financial market. The price of an option need to be computed to help investors in deciding if it is worthwhile entering a particular contract. Pricing multi-dimensional American-style Asian options is difficult because it involves early exercise possibilities, multi-dimensional asset management, as well as the path-dependency issue. Meanwhile, study of pricing multi-dimensional American-style Asian options is critical since it is widely traded in the market place. We address this problem in the current study.

The binomial tree method is one of the traditional numerical techniques that has been used for pricing options in the financial community. Due to the size and complexity of the options traded these days, this method is slow. However, this method is amenable to parallel implementation to make it faster.

In this thesis, we have developed a parallel algorithm for pricing the American-style Asian options and improved this algorithm by introducing a mathematical transformation to handle the interrelation of multi-dimensionality of the problem as well as reduce the computational cost to deal with the multi-dimensionality and hence can lead to quicker and accurate solutions.

We have tested the new algorithm on a 16 nodes distributed computing system and compared the new algorithm with the parallel algorithm for American-style plain op-

tions. We have also provided analytical results for the computation and communication complexities of the parallel algorithm. Option values are computed and the experimental results show that the algorithm converges. The mathematical transformation is proved to be essential through our experimental results. Compared with the parallel algorithm for pricing American-style plain options, the algorithm for pricing American-style Asian options offers a better problem for parallel computing because of its huge workload in computations.

Keywords: Financial Derivatives; Option Pricing; Asian Options; Parallel Algorithm.

Acknowledgements

I have to say thanks to many people who helped me to make this thesis possible.

First of all, I would like to express my thanks to my supervisor Dr. Ruppa K. Thulasiram (Tulsi). Dr. Tulsi is a nice professor with good academic research work. He provided a lot of advice and guidance to me ranging from research, writing, speaking to general life. He introduced me with an interesting new-born research area "Computational Finance" and taught me a lot about the field of high performance computing in computational finance. Dr. Tulsi also taught me how to do research step by step and helped me to get the chance to enjoy the happiness of having some publications. Most importantly, Dr. Tulsi showed me that the way of doing the research is pursuing my own idea and then enjoy the process of discovery and solving the problem. Nobody can tell you what you should do and how to do it. This is true in research as well as in general life.

I also would like to thank my thesis committee members, Dr. Carson K. Leung in department of computer science and Dr. Shiu. H. Lui in department of mathematics, for their valuable comments and suggestions that helped me to improve the quality of the final document.

I still would like to thank Dr. Parimala Thulasiraman for her help and advice throughout my course of the thesis. Dr. Thulasiraman introduced me with the high performance computing and its application in the course project and keep giving me advice when I need help.

Most of all, I would like to thank my husband and my parents. Without their understanding and supporting, I could not have the chance to finish this thesis.

Thanks must go to every member of the Parallel Algorithm In Manitoba's Algorithms and Applications (PARIMALA) lab for giving me all kinds of help.

Last but not the least, I would like to thank all my friends for your all kinds of help to make me happy, for the days we spent together.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Over the last decade, sophisticated financial instruments called derivative securities [16] have become increasingly popular in financial markets. Futures and options are now traded actively on many exchanges. Option pricing problem poses a serious challenge both for mathematical modelling and algorithm design, especially in issues such as early exercise possibilities and multi-asset derivatives. We address the problem of pricing complex options with multi-assets using advanced high performance computing techniques in this thesis.

As we all know, in the financial market, change can occur in a very short time period. The price of all kinds of assets, such as the stock indexes, may change significantly in such a short time. For market participants, it is necessary to have information as fast as possible so that they can respond to those changes as soon as possible. Otherwise, any delay in information process could lead to huge losses.

Option pricing is a problem that handles a huge amount of information. Among all kinds of options, the Asian option is one of the most representative examples of the option that are hard to price, especially the American-style Asian option with multi-assets. This is due to the fact that pricing the American-style Asian option with multi-assets involves early exercise possibilities, multi-dimensional asset management, as well as the path-dependency issues. There is no simple closed form solution for pricing Asian op-

tions. Numerical approximation methods are suggested in the literature. The analytical formulae approach cannot handle the early exercise possibilities. Simulating price path with Monte Carlo simulation can handle the path-dependency issue, but cannot deal with the early exercise possibilities as well as it is not efficient. Intuitive methods such as the lattice method can handle the early exercise possibilities, the path-dependency issue and the multi-dimensional problem but it has a relatively slow speed. Therefore, we adopt the lattice method and design a parallel algorithm to make it faster.

For this thesis work, we are focusing on pricing American-style Asian option with multi-underlying assets. The main contribution of this thesis is the development of a parallel algorithm for pricing American-style Asian option with multi-assets. To make the algorithm efficient, a virtual tree transformation (VTT) is introduced that reflects the dependency relationship among multi-assets. We introduce the basic terminology used in this thesis in the following subsection.

1.1 Definition

There are several research areas in finance. Only those terms relevant to option pricing are presented in this section.

- *the holder*: The holder is one party in a contract who has the right to purchase/sell an asset at a set price.

- *the writer*: The writer is the other party in a contract who has the obligation to fulfill the holder's decision.

- *call option*: The option allows the holder to buy the underlying asset with a certain price at or before a certain date.

- *put option*: The option gives the holder the right to sell the underlying asset with a certain price at or before a certain date.

- *strike price* or *exercise price*: The pre-decided price in the contract with which the holder will by or the writer will sell, when the option contract is executed for buy or sell.

- *maturity date* or *exercise date*: The pre-decided date on which the contract will expire.

- *underlying asset*: The financial asset such as a stock that the option holder can buy or sell.

- *European-style option*: The option that can only be exercised at the maturity date.

- *American-style option*: The option that can be exercised at any time before the maturity date.

- *Plain option* or *Vanilla option*: A normal option with standard expire date and strike price, no special or unusual features.

- *Asian option*: An option whose payoff depends on the average price of the underlying asset during the life of the option.

- *Multi-dimensional option* or *Multi-asset option* or *Basket option* or *Rainbow option*: The option with several underlying assets.

1.2 Organization

The thesis is organized as follows. First, we will introduce the background in Chapter 2. Second, we will describe the related work for pricing Asian options in Chapter 3. Third, we will present the Binomial Tree Model in Chapter 4, which is the financial model we used for this thesis work. Next, we will address the sequential algorithm in Chapter 5 followed by the mathematical transformation-Virtual Tree Transformation (VTT) in Chapter 6. Then we will explain the parallel algorithm in Chapter 7 followed by the

results with analytical results in Chapter 8.1 and the experimental results in Chapter 8.2. Finally we will present our conclusion in Chapter 9 and the future work in Chapter 10.

Chapter 2

Background

In this chapter, we would like to introduce some background information of this thesis work. The background information is important to help the reader to get a reasonable understanding of this research field. First, we will introduce derivative securities in Chapter 2.1. Second, we will present Asian options in Chapter 2.2. Third, we will address the option pricing problem in Chapter 2.3. Finally, we will describe the Black-Scholes model in Chapter 2.4.

2.1 Derivative Securities

Derivative securities are securities whose value depend on the value of other, more basic underlying assets [28]. The underlying assets include stocks, stock indices, foreign currencies, debt instruments, commodities, futures contracts and so on. For instance, a stock option is a derivative security whose value is contingent on the price of a stock.

People find it advantageous to trade in a derivative security on an asset rather than investing on the asset itself, because trading in derivative security is much better in reducing the risk associated with the fluctuating price of the asset. For example, suppose the current price of a certain stock is \$20 for one share, and an investor feels that it will rise. If the investor buys the stock now and the price rises to \$25 in 60 days, he or she

can at that time realize a profit of $5 for one share, so the return on his or her investment would be 5/20, or 25%. If the stock does not rise, the investor does not realize a profit and his or her money is locked if not lost. Now, suppose that instead of buying the stock itself, the investor buys, for $ 1, a European-style call option[1] that gives him or her the right (but not the obligation) to buy one share of the stock at $20 in 60 days. The 60^{th} day is known as the maturity date. If the stock price at the maturity date is less than $20, the investor will choose not to exercise the option, losing only the initial investment of $1 for one share. On the other hand, if the stock price at the maturity date rises to $25, the investor can exercise the option contract and buy the stock at $20 from the writer; and immediately sell that stock at $25 in the open market. Thus, the investor can realize a profit of $(25-20-1) for one share. The return on the investment is now (25-20-1)/1 or 400%. Derivatives securities, Derivatives, Options are all used interchangeably to refer to the same financial product.

2.2 Asian Options

Options on stocks were first traded on an organized exchange in 1973. Since then there has been a dramatic growth in options market. They are now traded in many exchanges throughout the world.

There are two basic types of options [16]. A *call option* gives the holder the right (but not the obligation) to buy the underlying asset by a certain date for a certain price. A *put option* gives the holder the right (but not the obligation) to sell the underlying asset by a certain date for a certain price. The price in the contract is known as the *exercise price* or *strike price*; the date in the contract is known as the *expiration date* or *maturity date*. *American-style options* can be exercised at any time before the expiration date (computing the earliest possible exercise date is a major problem in handling American

[1] A European-style call option is an option that can be exercised only on the expiration date, as already defined in Chapter 1.1.

options). *European-style options* can be exercised only on the expiration date.

A *path-dependent option* [16] (or history-dependent option) is a kind of complex option, whose payoff depends on the path followed by the price of the underlying asset, not just on the price on maturity date. For example, an *Asian option* is a kind of path-dependent option. The payoff from an Asian option depends on the average price of the underlying asset during the life of the option[2]. There are two kinds of average values: arithmetic and geometric. The arithmetic average price of the underlying asset is defined as the following:

$$A_i = \frac{1}{L}\sum_{i=0}^{L-1} S_i \tag{2.1}$$

where S_i is the asset price at time step L_i where the period of the option contract is decided into L time steps (time step 0 to $L-1$). Thus the local payoff[3] for an Asian option at any given node in level i is the following, where X represent the strike price:

$$\begin{aligned} Payoff &= Max(A_i - X, 0)\ for\ a\ call \\ &= Max(X - A_i, 0)\ for\ a\ put \end{aligned} \tag{2.2}$$

The option value at any given node is defined later in Chapter 5.1.

Compared with the standard options, Asian options have two important advantages. First, the averaging feature implies that Asian options can effectively reduce the effects of extreme price movements near or at the maturity date. Thus, Asian options can be used to avoid price manipulation. Second, Asian options are cheaper than the standard options [27].

A *multi-dimensional option*, also called *basket option* or *rainbow option*, is an option with several underlying assets. For instance, the value of a stock option with two underlying assets depends on these two different but related stocks. Multi-dimensional option is becoming increasingly popular over the last decade. We are focusing on pricing

[2]Sometimes, it may be some part of the life of the option.

[3]Local payoff means the payoff obtained if we exercise the option at this particular time step based on the current average asset price at this node. The option value or in other words discounted value of future payoff is presented as equation 4.3 in Chapter 4.2

American-style Asian options with multi-assets in this thesis.

2.3 Option Pricing Problem

The holder of an option must pay a certain price, called the "premium", to the writer of the option. The option pricing problem is to determine a "fair" price to pay for an option, and to decide if the contract is worthwhile to accept [28]. At the same time, investors will have help in deciding when to exercise the option for the sake of maximal profit or minimal risk. This problem requires information, such as the interest rate, the volatility of the asset's price, the price of the underlying asset, the expiration date, cash dividends etc. In addition, in many situations, the option price has to be computed in real-time, which means, any delay in information processing could lead to a loss of profit. Therefore, option pricing problem requires high performance computing capabilities.

2.4 The Black-Scholes Model

An option is priced, or "valued", by assuming some model of the price behavior of the underlying asset (e.g., a stock). Black and Scholes [3] introduced a continuous-time model for option valuation. The Black-Scholes model is a very important milestone in the financial world generally and in the options market particularly. Most of the later models are based on this model. By assuming that the asset price to follow Brownian motion, this method is manifested as a stochastic differential equation that the option price must satisfy. The Black-Scholes model can be solved in closed form for simple options such as European-style options. However, it is very difficult to solve Black-Scholes model for closed form solutions for complex options, such as path-dependent basket options that we are considering. Therefore, numerical techniques are needed.

Chapter 3

Related Work

The major problem in pricing Asian options is that we do not know much about the distribution of the underlying asset's average price. That is why there is no simple and exact closed form solutions for pricing Asian options. Approximation methods suggested in the finance literature can be grouped into three categories: Approximation analytical formulae, Monte Carlo simulations, and the binomial tree method (lattice method). There are other scientific computing techniques used these days for option valuation such as finite-difference method, fast Fourier transformation etc. These are out of scope for the current thesis work and hence we are not discussing them here.

3.1 Pricing Asian Options using the Approximation analytical formulae

This approach intends to value the Asian Options using approximation by closed form formulae. The geometric average Asian options can be priced exactly for the Black-Scholes model since the geometric averages of lognormal variates are still lognormal.

Some related work in this category try to approximate the probability density function of the average asset price, in which the arithmetic average of the asset price is assumed to follow approximately a known distribution. Turnbull and Wakeman [26] try

to approximate the density function of the average asset price by Edgeworth series expansion and they get a closed form solution for pricing European-style geometric Asian option. Milevsky and Posner [20] approximate the distribution of the underlying asset's average price by the reciprocal gamma distribution, and they obtain a closed form analytic expression for the value of European-style arithmetic Asian option. Benhamous [2] use fast Fourier transformation to approximate the payoff function at the maturity date, and derives an efficient algorithm for European-style arithmetic Asian options, which is flexible to be adapted to non-lognormal densities. Geman and Yor [14] derive an analytical expression for the Laplace transform of the European-style arithmetic Asian calls. Numerical inversion of this transform is considered by Fu, Dilip and Wang [13], Shaw [22], Craddock, Heath and Platen [10]. Zhang [29] approximates the option value by combining an analytical closed form with a numerical adjustment (computed by finite-difference method) and derives a new analytical approximation formula for pricing the European-style arithmetic Asian option, which is more efficient and accurate than work done before.

The major problem of the approximation analytical formulae approach discussed in these work is that the American-style Asian option's value cannot be approximated since the approximate analytical formula cannot handle the early exercise probabilities easily. Therefore, a numerical method is required for the American-style Asian option pricing problem.

3.2 Pricing Asian Options using the Monte Carlo Simulation

Another approach to valuing options is Monte Carlo simulation [23, 4]. The Monte Carlo simulation approach is an established numerical tool in the finance academic community for the pricing of derivative securities. An estimated value of the derivative is found by obtaining random sample values of the derivative and calculating their mean. Monte

Carlo simulation trends to be numerically more efficient than other procedures because the running time for a Monte Carlo simulation increases approximately linearly with the number of variables, whereas the running time for most other procedures increases exponentially with the number of variables. Monte Carlo simulation is an approach that can accommodate complex payoffs. It can be used when the payoff depends on some function of the whole path followed by a variable, not just on its value on the maturity date, such as the Asian option.

Mascagni and Chi [19] explore the quasi-Monte Carlo approach, which use quasi-random sequences, and it seems that the quasi-Monte Carlo approach can provide a faster rate of convergence than Monte Carlo approach. However, the (quasi-)Monte Carlo simulation algorithm is not efficient because even with a large number of random samples, the accuracy of pricing is not satisfactory. Some related work have studied this inefficiency problem. For example, Boyle, Broadie and Glasserman [5], Broadie and Glasserman [6], Broadie, Glasserman and Kou [7], refine the random variable they use to reduce the variance of the resulting option price whereby they improve the efficiency through reduced number of Monte Carlo simulation.

Same as the approximation analytical formulae approach, another problem of the Monte Carlo simulation is that it can not handle American-style options easily. Although Longstaff and Schwartz [18] develop a least-squares Monte Carlo approach to approximate the value of the American-style plain options by simulation, nothing has been done for applying Monte Carlo simulation to approximate American-style Asian options since it involves dealing with the early exercise possibility as well as the path-dependency issue. In addition, there is no challenge for parallel algorithm design of Monte Carlo simulation, as we know, Monte Carlo simulation gives rise to an embarrassingly parallel situation. Therefore, a different approach is needed to price the American-style Asian options.

3.3 Pricing Asian Options using the Binomial Tree Method

A binomial tree [9] is an intuitive, useful technique for pricing options. This technique has been traditionally used by finance community for long time. A binomial tree is a representation of different possible paths (up or down movements of the price over various times) that might be followed by the stock price over the life of the option.

The literature on the use of binomial trees for pricing options is very vast. In the rest of this section, we are focusing only on the use of this technique for pricing Asian options. The detailed description of the binomial tree method will be given in Chapter 4.

The Asian option can be priced by the tree approach. However, the pricing algorithm is much more complex than the plain option since the value of the Asian option is influenced by the historical average price of the underlying asset. A successful approximation is suggested by Hull and White [17]. This approximation limits the number of possible prefix sums[1] at each node to a manageable magnitude k. The option values for the missing prefix sums are estimated by interpolation. The Hull-White approximation has been analyzed and extended in later studies. For example, Barraquand and Pudet [1] apply numerical technique structured similar to binomial tree method known as finite-difference to solve the option pricing problem. Another work by Roger and Shi [21] uses the finite-difference technique to price further complicated Asian options with floating and fixed strike price. Inspired by the ideas of Rogers and Shi, Chalasani, Jha, and Varikooty [8] derive accurate lower and upper bounds for Asian option, using a binomial tree model. In addition, as an alternative to binomial trees, trinomial trees have been used by Dai and Lyuu [11] to price Asian options, as well. The big advantage of the binomial tree is that it can be used to accurately price American-style options. This is because with the binomial tree it's possible to check at every point in an option's life (i.e. at every step of the binomial tree) for the possibility of early exercise. On the other hand, the limitation

[1]The prefix sums will be defined in Chapter 4.3.

from the binomial tree is its relatively slow speed. Therefore, the binomial tree could be a good model to be used for American-style Asian option pricing as long as we make some modification to increase its computational efficiency.

Chapter 4

Binomial Tree (BT) Method for Asian Option Pricing

Since the binomial tree could be a good technique to be used for American-style Asian option pricing, we will describe the binomial tree method in detail in this chapter. First, we will introduce the non-recombining binomial tree in Chapter 4.1. Second, we will present the binomial tree in Chapter 4.2. Next, we will address the binomial tree for Asian options in Chapter 4.3 followed by a numerical example in Chapter 4.4. Finally, we will describe the limitation of the binomial tree in Chapter 4.5.

4.1 Non-recombining Binomial Tree

Recall that a binomial tree represents stock price movements as new edges. In a short time interval, a stock price can either increase by a proportional amount u, or decrease by a proportional amount d. The size of u and d and their associated probabilities are chosen so that the proportional change in the stock price has correct mean and standard deviation in a risk-neutral world[1]. Option prices are computed by starting at the end

[1]Risk-neutrality means that the investment on options is assumed to yield at least equivalent to the returns from a bank investment at a fixed interest rate.

of the tree (maturity date) and working backward. Figure 4.1 is a price movement of an underlying asset in a non-recombining fashion. From Figure 4.1, it can be observed

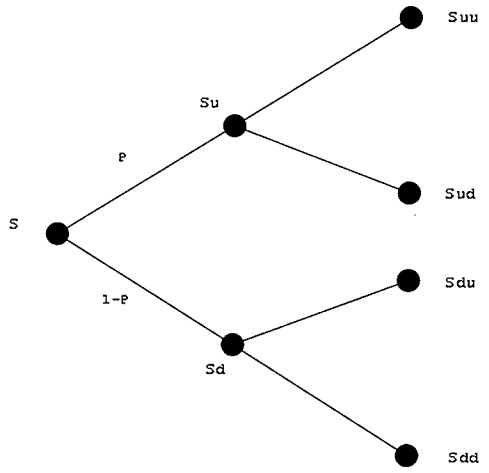


Figure 4.1: The two step non-recombining binomial tree

that two step non-recombining binomial tree has four leaf nodes, which represent that there are four possible stock prices after two time steps. In the non-recombining binomial tree, the number of leaf nodes is increasing exponentially. On time step L, the number of leaf nodes is 2^L. This huge number may restrict the usefulness of algorithm based on the non-recombining binomial tree. Therefore Cox, Ross and Rubinstein popularized the recombining binomial tree in 1979 [9].

4.2 The Binomial Tree

The well-known Cox-Ross-Rubinstein (CRR) binomial tree model is introduced in figure 4.2. Here the fact that the tree is recombining is used, which means that in figure 4.1 we exploit the fact that $S_{ud} = S_{du}$ and then get the recombining binomial tree in figure 4.2. In the recombining binomial tree, the number of leaf nodes on time step L is only $L+1$. In the following sections of this thesis, wherever we mention the binomial tree, it represents the CRR binomial tree, while we will explicitly mention wherever we

use non-recombining tree.

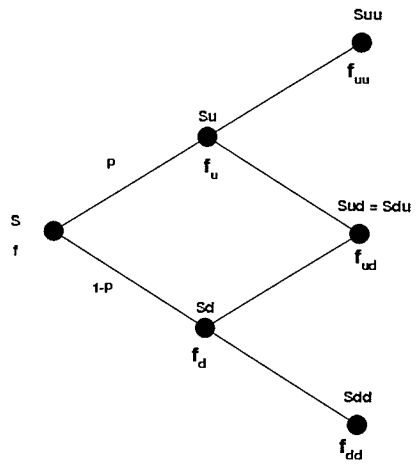


Figure 4.2: The two step binomial tree

A binomial tree can be used to capture the asset price movement directly: A stock price S at the beginning of a period, can increase (by a multiplicative factor) to Su with probability p or may decrease to Sd with complementary probability $(1-p)$ after one time interval (time step). When the asset price moves up from S to Su, the payoff from the derivative is f_u while the payoff is f_d when the price goes down from S to Sd. We construct a portfolio with a long position (the state of actually owning a security or commodity) in Δ share of stocks and a short position (the promise to sell a fixed amount of goods at a fixed price in the future) in one call option then we can calculate the value of Δ which makes the portfolio risk-free.

When the stock price increases to Su, the value of the portfolio at the maturity date of the derivative will be $Su\Delta - f_u$ and if the stock price decrease, the value of the portfolio will be $Sd\Delta - f_d$. The portfolio is risk-free if the value of Δ is chosen so that the final value of the portfolio is the same for both of the alternative stock prices. This means that:

$$Su\Delta - f_u = Sd\Delta - f_d$$

$$\begin{aligned} S(u-d)\Delta &= f_u - f_d \\ \Delta &= \frac{f_u - f_d}{S(u-d)} \end{aligned} \tag{4.1}$$

In this case the portfolio is riskless and must earn the risk-free interest rate. Let r represent the risk-free interest rate. The present value of the portfolio is $(Su\Delta - f_u)e^{-r\Delta t}$. In addition, the cost of setting up the portfolio is $S\Delta - f$. For a risk neutral portfolio, therefore,

$$S\Delta - f = (Su\Delta - f_u)e^{-r\Delta t} \tag{4.2}$$

Substituting from equation (4.1) we get

$$f = e^{-r\Delta t}(pf_u + (1-p)f_d) \tag{4.3}$$

where

$$p = \frac{e^{r\Delta t} - d}{u - d} \tag{4.4}$$

Equation (4.3) and equation (4.4) enable an option to be priced using a one step binomial tree.

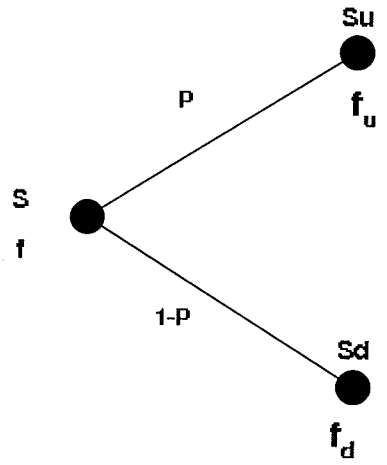


Figure 4.3: The one step CRR binomial tree

4.3 Binomial Tree for Asian Options

The Asian option can be priced by the tree approach. However, the pricing algorithm is much more complex as the value of the Asian option is influenced by the historical average price of the underlying asset. For most nodes, there is more than one possible option value at a node since there is more than one price paths reaching this node and most of these price paths carry distinct historical average prices. For convenience, *path prefix* is introduced as follows: A *path prefix* is defined as a partial price $(S_0, S_1, S_2, \ldots, S_j)$ that starts at time 0 and ends at time j. The sum of this *path prefix* is defined by $\sum_{i=0}^{j} S_i$, called a *prefix sum.* For a node N illustrated in Figure 4.4, we can find a path prefix that has the maximum prefix sum among the path prefixes that ends at N. This maximum path prefix is denoted by the upper path (in thick lines) that ends at N. Similarly, the path prefix that has the minimum prefix sum is denoted by the lower path (in thick lines) that ends at N. The prefix sum range for N is defined as the range between the maximum and the minimum prefix sums for N.

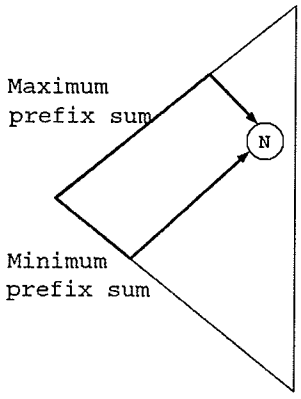


Figure 4.4: The maximum and minimum prefix sum

4.4 A Numerical Example

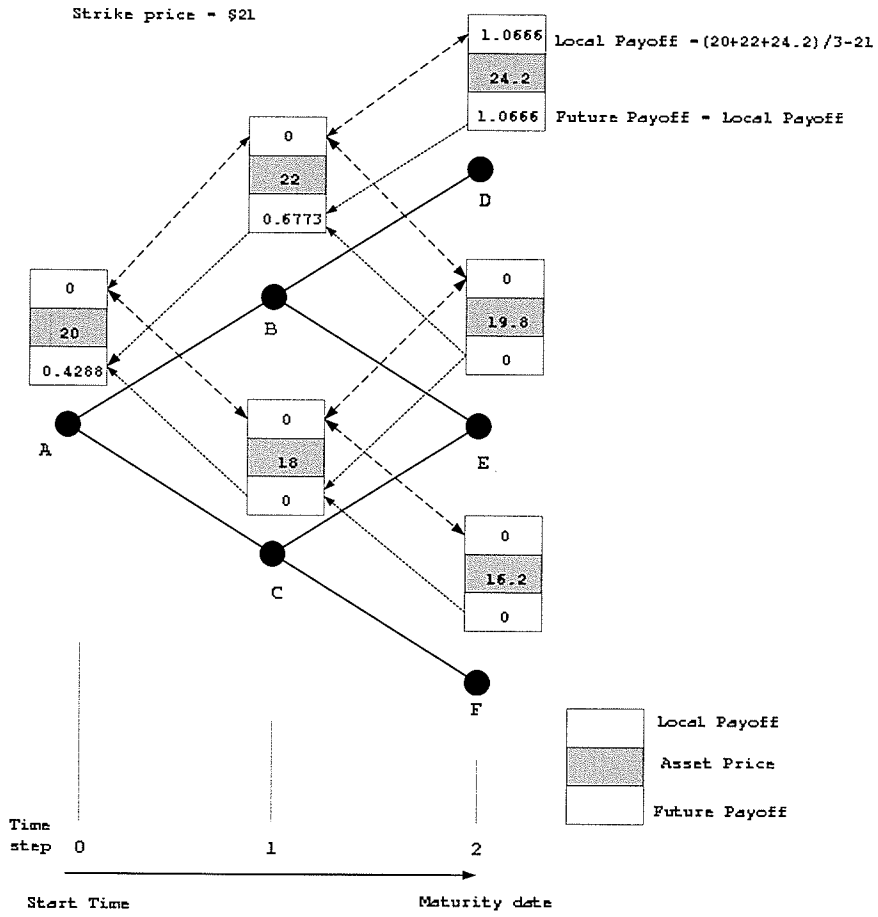


Figure 4.5: Numerical example of a two step binomial tree for Asian Call

Figure 4.5 is a pricing example of the Asian option on a two step binomial tree. The whole life of the option is divided into 2 periods evenly. The start date is time step 0, the intermediate date is time step 1, and the maturity date is time step 2. The whole lifetime of the option is 6 months.

We set the start price is \$20, the risk-free interest rate is 12% per year. The strike price is set to \$21.

Node A is the current start time. Node B and C are the intermediate nodes in the

option's lifetime and nodes D, E, F are the possible terminal nodes that the stock will reach on the maturity date. The periods are assumed to be equal for simplicity and the up and down movements are also assumed to be at a constant rate (of 10%) where $u = 1.1$ and $d = 0.9$. Generally, the increase and decrease of the stock price do not need to be equal as are assumed in this numerical example. In figure 4.5, the number in the gray cell represents the price of the underlying asset of the corresponding node. The number in the white cell above the grey cell denotes the local payoff at that node, which is the payoff from the option as soon as any intermediate node is reached. The number in the white cell below the grey cell denotes the future payoff at that node.

For node D, the option price (future payoff) calculated equals to the local payoff. It is calculated from the average stock price of node A, B, D minus the strike price. Therefore, at D, *the option price = average stock price of node* A, B *and* $D-$ *strike price*$= (20+22+24.2)/3-21 = \$1.0666$. For any terminal node this value is local payoff as well as the option value. At nodes E, since the *average stock price of the nodes on the path((20+22+19.8)/3=20.6) < strike price(21)*, the option value is \$0. Similarly, the option values at node F and node C are \$0. At node B, the local payoff can be calculated to be $0(averagepathprice - strikeprice = (20+22)/2-21 = 0)$. Assuming a 10% increase and decrease (which means $u = 1.1$ and $d = 0.9$), at node B, p can be computed using equation(4.4) to be 0.6523 and accordingly the option value at B can be calculated as 0.6733 using equation(4.3). After calculating the stock price and option value at nodes B and C, option value at node A is calculated as 0.4288 using equation(4.3). In this example, we assume that the up and down movements are equal and this assumption can be reset in a more realistic way.

4.5 Limitation of the Binomial Tree

The big advantage of the binomial tree method is that it can be used to accurately price American-style options. This is because with the binomial tree it is possible to check at

every point in an option's life (i.e. at every step of the binomial tree) for the possibility of early exercise.

On the other hand, the limitation of the binomial tree is its relatively slow speed because of the huge workload. Also, as we know, high speed computing is required to beat the competition in the market place and any delay in the computing may lead to loss of money. These two reasons naturally lead us to the use of parallel computing. Since the structure of a binomial tree lends itself to a parallel implementation, there are good opportunities for parallel algorithms research, see [25], for example. In the current research, we would like to develop a parallel algorithm for the binomial tree approach to price American-style Asian option with multi-assets.

Chapter 5

Sequential Algorithm for Pricing Asian Options

Binomial tree method involves two major phases: (I) constructing the tree, and (II) backtracking the tree to compute the option values. First, in this chapter, we will discuss the sequential algorithm which presents the idea of pricing the Asian option in general. When there are many underlying assets in a given option, many trees need to be constructed with appropriate dependencies on the asset they are supposed to depend on. This is quite expensive to handle. In Chapter 6, we propose a novel way to cut the tree construction time by developing a mathematical transformation, called *Dependent Tree transformation* or *Virtual Tree Transformation (VTT)*, which will create tree for all the underlying assets virtually. Then, we will focus on the parallel algorithm in Chapter 7 to expedite the computing process.

Recall that European-style options can be exercised only on the expiration date while American-style options can be exercised at any time before the expiration date. The pricing algorithm of American-style options can be designed by adding features such as to handle the early exercise possibilities into the pricing algorithm of European-style options. Therefore, we will explain the sequential algorithm for European-style Asian options in Chapter 5.1 followed by the sequential algorithm for American-style Asian

options in Chapter 5.2.

5.1 Sequential Algorithm for Pricing European-style Multi-asset Asian Options

The goal of this pricing problem is to find the value of the European-style Asian option. To compute the option value using binomial tree, first, the time interval [0, T] (the life of the option) is divided into L smaller intervals (time steps) evenly, each of length $\Delta t = T/L$. For instance, in Figure 4.3, when the time step is equal to 0 (representing current time), there is only one node on this level, which is the root. When the time step is equal to $1(\Delta t)$, there are two nodes on level 1 which are children of the root. Over each sub-interval, the asset price is assumed to move up (for example, from value S to S_u [1]), or down (for example from S to S_d [2]), with probabilities p and $1-p$, respectively.

Two step non-recombining binomial tree has four leaf nodes, which represents that there are four possible stock prices after two time steps (see Figure 4.1). Recall that in the non-recombining binomial tree, the number of nodes is increasing exponentially and at time step L, the number of nodes is 2^L. This huge number may restrict the usefulness of the algorithm based on the non-recombining binomial tree. In order to reduce the number of leaf nodes, we make use of the fact that the tree is recombining, which means that in Figure 4.1 we exploit the fact that $S_{ud} = S_{du}$ and then get the recombining binomial tree in Figure 4.2. In the recombining binomial tree, the number of leaf nodes on time step L is only $L+1$.

This algorithm will proceed in two phases, phase I and phase II. Phase I involves two steps: (i) Spawn the binomial tree; and (ii) Compute the prefix sum of each path of the tree[3]. The computation starts from the initial node (root) with the given asset price S_0.

[1] S_u is the node where the asset price is Su

[2] S_d is the node where the asset price is Sd

[3] This step is needed since we are pricing an Asian option, which is path-dependent.

New nodes (called children nodes) are generated corresponding to the immediate future dates (i.e., in one time-step). Then the asset prices are computed at these nodes by either multiplying by "u" or by "d" for the price movements of the underlying asset. In step (ii) the sum of each path is obtained consequently. Further stepping is done towards the maturity date by repeating steps (i) and (ii) at all subsequent nodes and paths, until the leaf nodes are reached.

During phase II, the option value is computed. This phase also has two steps:

- Step (i): computes the local payoff. The payoff is computed by the equations (2.1) and (2.2).

- Step (ii): the algorithm computes the option values at the time step t=$L-1$ using the 1-step binomial formula given in equation (4.3), which is.

$$f_{L-1} = (e^{-r\Delta t})\{pf_u^L + (1-p)f_d^L\}$$

where r is the interest rate; Δt is the time step; p is the probability of up movement of the asset price; f_u^L is the option value at the child node corresponding to an up movement; f_d^L is the option value at the child node corresponding to a down movement of the asset price. This is known as *discounted value* of possible future payoff, and the 1^{st} form $(e^{-r\Delta t})$ is known as discounting factor and the 2^{nd} form $(pf_u^L + (1-p)f_d^L)$ is the future payoff. For our reference, we call equation (4.3) simply as *future payoff*.

The computation starts at the leaf nodes ($t = T$) where the option value (also the local payoff) is computed using equation (2.2). Then, repeat the computation in steps (i) and (ii) of phase II at each time step retracing the path (a backward march) to the initial node. The value of the option computed at the initial node (that is, time=0) is the option value of the problem under study, for European-style Asian options.

5.2 Sequential Algorithm for Pricing American-style Multi-asset Asian Options

However, more work is required for American-style Asian options, since it involves finding the best node for exercising the option. Besides two phases mentioned in Chapter 5.1, in order to find the best node,

- At each time step, $t = L_i$, the local payoff at each node j of time step L_i denoted as $f^j_{L_i}$ is calculated as ${}_{local}f^j_{L_i} = S^j_{L_i} - X$ where $S^j_{L_i}$ is a net price at the node j on level L_i and X represents the strike price. For Asian option, $S^j_{L_i}$ is $A^j_{L_i}$.
- As soon as the option value is computed using equation 4.3, a comparison is made between option value and the local payoff (from previous step) to find the maximum.
- If the maximum is the local payoff ${}_{local}f^i_L$ at a particular node , as soon as the node is reached, the option can be exercised without waiting for another time step.
- Such comparison is repeated at each node of every level to find a global maximum. Wherever the global maximum occurs is the node where an American Asian option can be exercised.

Both the two phases mentioned in Chapter 5.1 and four points mentioned above are needed for pricing American-style Asian options. They need to be repeated for every asset in a multi-dimensional option. This is computationally costly. We have devised a transformation that accounts for all the underlying assets, without having to repeat the construction of a tree.

Chapter 6

Virtual Tree Transformation (VTT)

One of the previous studies [24] computed the option with ten assets using binomial tree method. The approach involves 6 steps.

1. The asset prices at each node on the binomial tree of the first asset will be generated in the standard procedure.

2. The option price of the first asset will be computed by applying the pricing algorithm as described in the previous chapter.

3. The third step is generating the binomial tree of the second underlying asset that depends on the asset price of the corresponding node of the first tree. The dependent relationship between two corresponding nodes of the two trees could be set to follow a proportional trend. For example, asset price at each node in the second tree is 90% of the asset price of the corresponding node of the first tree.

4. The next step is to compute the option price for the second asset.

5. Then, repeat steps 3 and 4 until all the underlying assets are represented in trees and option values are computed.

6. Finally, obtain the value of the option with ten underlying assets by calculating the mean value (as a simple rule) of these ten option values.

In [24] ten individual binomial trees were constructed separately, which is time consuming. Therefore, we have devised a mathematical transformation that eliminates the tree construction time where the asset prices are computed for each node of the tree (Phase I in the algorithm).

For example, for an option with two underlying assets, tree corresponding to the first asset will be constructed normally as described before. For the second asset a virtual tree is created using the relation $S_L^{*i} = \alpha S_L^i$, where S_L^{*i} is the asset price for the second asset at node i of time step L, S_L^i is the asset price for the first asset at node i of time step L and α is a dependency factor, $0 < \alpha < 1$.

This is a simple and straightforward transformation. That is, the second asset assumed to have a dependence on the corresponding node of the first asset. This needs not be so in the real world. Therefore, a partial proportional dependence on each node of time step L could be more practical. In other words,

$$S_L^{*i} = \alpha S_L^i + \beta S_L^j + \gamma S_L^k + ... \tag{6.1}$$

Where α, β, γ etc are all between 0 and 1 and they are the dependency factors at various nodes of the second tree. We can rewrite the above formula to

$${}^{\Gamma}S_L^{*i} = \sum_{i=1}^{n} \alpha_i S_L^i \tag{6.2}$$

where ${}^{\Gamma}$ is the asset number ${}^{\Gamma}$ =1, 2, 3..., 10, α_i are constants ($0 < \alpha < 1$) and n is the number of nodes on level L. With the above virtual tree transformation (VTT) in equation 6.2 developed to capture price movement of all the assets, option values for multi-dimensional European-style Asian options can be computed. For American-style Asian option, further complications arising from the possibility of exercising the option earlier than the maturity date, are explained in Chapter 5.2. Pricing computation for such complicated options in real-time becomes intensive. Parallel computing could help in this process.

Chapter 7

Parallel Algorithm for Pricing Asian Options

In practical option pricing applications, problem size is large. For example, a large number of time steps lead to exponential growth of the tree. Parallel computing is preferred especially when real-time solution is needed.

This algorithm starts from level L (time step L, i.e. leaf nodes). The number of levels in the tree and the number of processors are assumed to be power of two. For a recombining tree, the number of leaf nodes is always equal to the number of levels plus one, which is $(L+1)$. All leaf nodes are evenly distributed among the processors but the last processor receives an additional node, see Figure 7.1 for example. Initially the option prices at the leaf nodes are calculated by finding the difference between the average prefix sum price and strike (exercise) price, for example $(A_L - X)$ for a call, which is the same as the local payoff at these nodes. Every processor i, except the processor 0, sends the value of its boundary node to processor $(i-1)$ (Higher numbered processor sends data to the immediate lower numbered neighbors). In a given processor, for every pair of adjacent nodes at a certain level L_i, the processor computes option price for the pair's parent node, which is at level L_{i-1}. The computation proceeds to the previous level L_{i-1}, and option price computation is repeated with additional exchange of boundary node values.

Eventually, the processor 0 computes the option price at the level $L = 0$. For example, in Figure 7.1, the tree is divided to work on 4 processors. Node A is on processor P1, and node E is on processor P0. In order to compute the option value on node B, processor P1 has to send the local payoff value of node A to processor P0 and compute the option value of node B by equation(4.3).

At each time step t, this algorithm operates in two modes simultaneously: computation and communication. In the communication mode, processors send and receive data on option values. Processor 0 only receives data from processor 1. Processor 1 through processor $(P-2)$ (P is the number of processors) receive data from the processor's immediate higher numbered neighbor and send data to their immediate lower numbered neighbor. Processor $(P-1)$ only sends data to its immediate lower numbered neighbor.

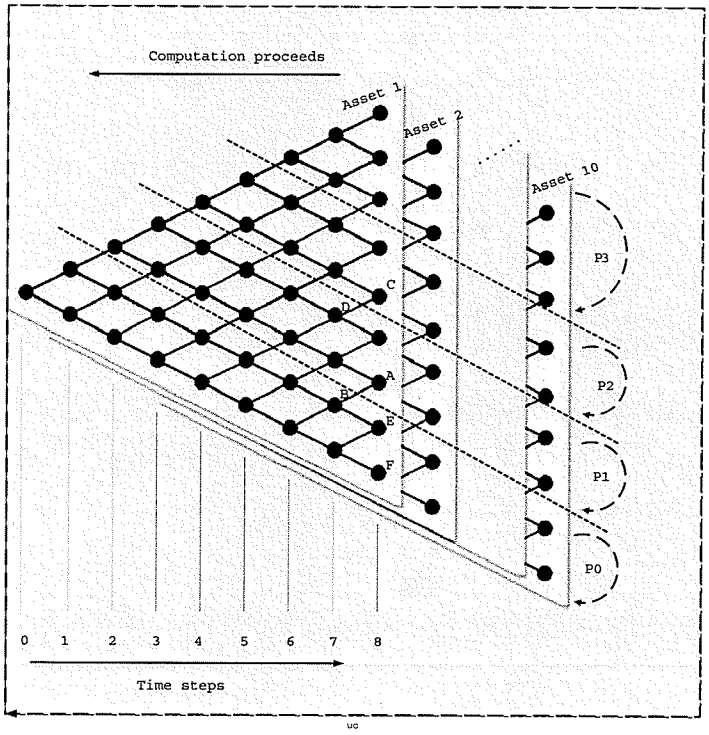


Figure 7.1: Parallel Algorithm

For American-style options, because of the early exercise possibilities, there is a need for an indexing scheme to keep track of the best option value and the corresponding node information. For example, for every node in the binomial tree, if the option value on one node is bigger than a certain value, this option value will be recorded as well as the node position in this tree. Finally, all the recorded option value will be compared to each other. The maximum value will be picked as the option value and the position of the node which gives the maximum option value will also be reported as the best node to exercise the option. Since we are designing the parallel algorithm for this problem, the data of whole tree is assigned to several processors and several processors are working simultaneously. Then how to keep track of the best option value and the corresponding node position is tricky among multi-processors. With one index on the node we will clearly distinguish between two nodes of a given level of the tree and with another index we will distinguish two different level of the tree. The index $^{\Gamma}$ defined in equation(6.2) will form a third index to distinguish a node of a tree corresponding to one asset from the corresponding node of a tree of another asset. This indexing scheme can be tracked among various processors, with a unique index for each node.

We use the C programming language together with MPI for our implementation on a distributed memory architecture. MPI (Message Passing Interface) is a library specification for message-passing, proposed as a standard by a broadly based committee of vendors, implementors, and users. The complete details on MPI is available at *http://www.lam-mpi.org.*

Chapter 8

Results

In this chapter, we present some analytical results of the algorithms first followed by the experimental results.

8.1 Analytical Results

Here, we explain the total number of computations and communications required among various processors to compute the option value. To make the explanation easier, we use an example binomial tree with 8 levels and 2 processors and generalize the complexity analysis for L levels and P processors.

8.1.1 Total Computations

First, we are interested in the total number of computations performed on each processor. In order for comparison, we will first present the analysis of computations for plain options, followed by that of computations for Asian options.

Computations for Plain Options (TCP): First, let us consider the total computations for the plain options. In Figure 8.1, let us consider the processor P1. Processor P1 has five leaf nodes (at level 8). The longest path from any of the five leaf nodes ends at level 4. The number of nodes allocated to this processor initially is $L/P + 1(= 5)$.

Each of these nodes have their parent nodes at level 7. One of the four nodes at level 7 are boundary nodes (node G; boundary node is one node that needs to be communicating to complete the computation of option pricing done at a given level in the tree). Note that in our algorithm, the communication is from a higher numbered processor to a lower numbered neighboring processor. Similarly, these four nodes at level 7 have parent nodes at level 6. The three nodes at level 6 have two parent nodes at level 5 and these two nodes have one parent node at level 4 (node H), which is also a boundary node local to processor P1. Note that all the nodes we have just discussed form a triangle starting at level 4 and ends at level 8 (collected within a circle). These are the nodes computed by processor P1 locally. The number of computation, therefore, is $1+2+3+\ldots+L/P = (L/P) \times (L/P+1)/2$ on processor P1.

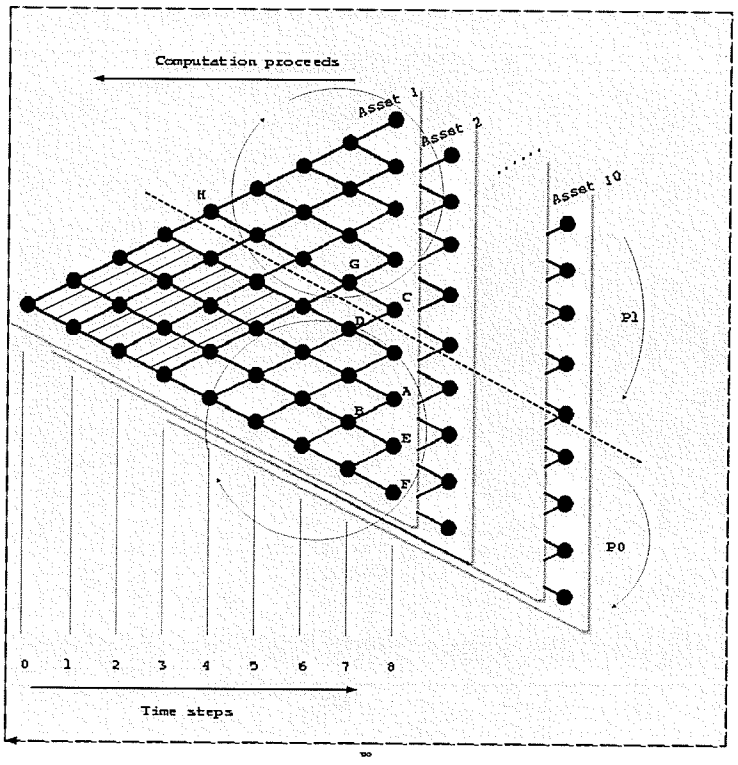


Figure 8.1: Computation of plain option with 8 levels and 2 processors

Let us now consider processor P0. The longest path from leaf nodes of this processor ends at level 0. We use the technique of triangles discussed above to calculate the total number of computation performed by processor P0.

We superimpose all the local nodes of processor P1 (represented as a triangle) to cover some of the processor P0 nodes (that is, the first node (counting from the bottom) at the level 4 in P0 can be superimposed on to the fifth node at the same level, which is in P1). This way, we can cover most of the processor P0 except for the square area (actually a rhombus) shaded. The number of nodes on each side of this square is equal to L/P and hence, the total number of nodes in this square is $(L/P)^2$. Therefore, processor P0 has $(L/P) \times (L/P+1)/2 + (L/P)^2$ total nodes for computations. Therefore, the number of total computations for plain option (TCP) for processor Pi can be obtained as the following:

$$TCP = (L/P) \times (L/P+1)/2 + (P-i-1) \times (L/P)^2, 0 < i < (P-1) \qquad (8.1)$$

For a plain option with 10 underlying assets, the total computation is 10 times of the total computations of a option with single asset.

Computations for Asian Options (TCA): Since each processor is dealing with the nodes on a path that reaches the node on this processor, it is difficult to give a general formula for the number of computations on each processor. However, a careful investigation on the computational structure of the tree reveals a simple pattern with which we can derive a generalized formula for total computation without loss of generality.

For Asian options, first of all, average value of the underlying asset need to be computed. This is additional cost over the plain options. In this section we present an analysis on the total number of average computations performed by each processor.

Recall that for computing the average asset price at a node, the information of all the nodes on the path from the root to this target node is needed[1]. Let us take Figure 8.2 as an example. It can be observed that the nodes on the upper and lower price boundaries

[1]In our experiments, without loss of generality, we calculate the underlying asset's average price by computing the mean from the maximum and minimum prefix sum. (refer Figure 4.4)

have only one path information to compute the average value. That is, reaching the first or last node at any given time level (nodes at any given level are counted from the bottom) can be achieved by only one path each. The other nodes that are not on the price boundaries have minimum of 2 paths to reach and we can use those two paths to compute their average (also refer Figure 4.4). Without loss of generality, we suppose that all the nodes in the tree would suffice to consider 2 paths information to compute the average value, maximum and minimum paths. A careful consideration reveals that this hypothesis simplifies the total computation time without affecting much the computed average values at each node. Also note that the number of boundary nodes are far less than the number of interior nodes and hence the assumption of each node having two paths including the boundary nodes approximates well for a large tree.

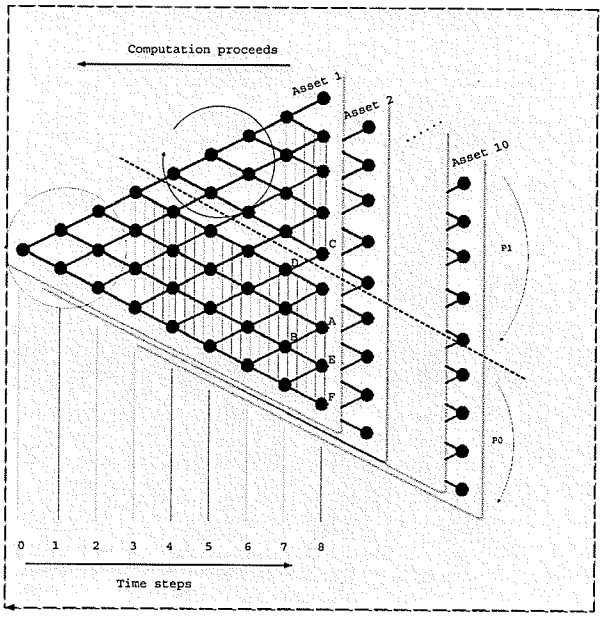


Figure 8.2: Computation of Asian option with 8 levels and 2 processors

From Figure 8.2, we can find that the computation of one processor can be divided into two parts. Let us take processor 0 in Figure 8.2 as an example. The first part is in the

Table 8.1: Computation of triangle area of P0 in Figure 8.2

level i	Nodes	Computations on each node	Total computation
0	1	$2*(L_0+1)$	$2*(L_0+1)$
1	2	$2*(L_1+1)$	$2*2*(L_1+1)$
2	3	$2*(L_2+1)$	$3*2*(L_2+1)$

circle area (presented as a triangle) and the second part is in the shaded area (presented as the rhombus). Hence, the total average computation is made up of 2 parts, computation on the triangle area (CT) and computation on the rhombus area (CR). First, we consider part 1, the triangle area. There are 3 levels in this triangle area and the computation information is shown in Table 8.1. Note that L_i represents the number of nodes need to be visited for the computation of the node on level i. Then the total computation of the triangle area on P0 is $2*(L_0+1)+2*2*(L_1+1)+3*2*(L_2+1) = \sum_{k=0}^{2}\sum_{j=1}^{k+1}(j*2*(k+1))$. Similarly, the total computation of the triangle area on P1 is $\sum_{k=4}^{6}\sum_{j=1}^{k-3}(j*2*(k+1))$. Then, we can derive a general formula for the computation of the triangle area (CT) on Processor Pi:

$$CT = \sum_{k=i*(L/P)}^{((i+1)*(L/P)-2)} \sum_{j=1}^{(k+1-i*(L/P))} (j*2*(k+1)), 0 < i < (P-1) \qquad (8.2)$$

Second, we consider part 2, the shaded rhombus area in Figure 8.2. We still take processor 0 in Figure 8.2 as an example. The number of nodes allocated to this processor initially is $L/P(=4)$ (leaves). There are $4(=L/P)$ nodes on each level from level 3 to level 8 in this shaded rhombus area and the computation on each node is $2*(L_i+1)$. Therefore, the computation on level i is $(L/P)*2*(L_i+1)$. Then, the total computation of this shaded rhombus for processor P0 is $\sum_{j=3}^{L}((L/P)*2*(j+1))$. Similarly, the total computation of this shade rhombus for P1 is $\sum_{j=7}^{L}((L/P)*2*(j+1))$. There is one more node on P1 (the top most one at level 8), because P1 is the last processor and gets one extra node when distributing the leaf nodes. We skip this node, without loss of

Table 8.2: Computation for 2 processors and 8 time steps.

	P0	P1
TCP	26	10
TCA	388	276

generality, and get a general formula for the total computation on the shaded rhombus (CR) of processor Pi:

$$CR = \sum_{j=((i+1)*(L/P)-1)}^{L} ((L/P)*2*(j+1)), 0 < i < (P-1) \tag{8.3}$$

According to equations(8.2) and (8.3), we get the total computations of Asian options (TCA) on processor Pi as the following:

$$\begin{aligned} TCA &= \sum_{k=i*(L/P)}^{((i+1)*(L/P)-2)} \sum_{j=1}^{(k+1-i*(L/P))} (j*2*(k+1)) \\ &+ \sum_{j=((i+1)*(L/P)-1)}^{L} ((L/P)*2*(j+1)) \\ &+ TCP, 0 < i < (P-1) \end{aligned} \tag{8.4}$$

Now we compare the computation of Asian option with the computation of plain option in [24] which studied the parallel algorithm of American-style plain options. We use the same binomial tree with $L = 8$ and 2 processors. As presented in Table 8.2, the total computation for plain options (TCP) is 26 for P0 while the total computation for Asian options (TCA) is 388 for P0. Also, the total computation for plain options for P1 is 10 while for Asian options is 276. Therefore, we can say that the computation cost of Asian options is larger than that of plain options by an order of magnitude.

If we compute the computations of American-style Asian options with 10 underlying assets, the computations should be 10 times of the single asset Asian option.

8.1.2 Total Communications

By communication, we mean that a single send and receive operation constitute complete communication. Therefore, we can only compute the amount of sending or receiving in the following and then find the total amount of communication.

Let us take an example binomial tree with 8 levels and 4 processors. Recall that at each level of the tree, the higher numbered processor sends a data value to the immediate lower numbered processor. Therefore, processor P0 only receives some node values from P1. Processor P3 sends $(L/P + 1)$ data values to processor P2; processor P2 receives that many data values and in turn sends $(L/P + 1) + (L/P)$ date values to processor P1, etc. Therefore, the total number of communication (TC) is:

$$TC = \sum_{i=0}^{P-1}(1 + (L/P) * (P - i)) = (P - 1)(1 + L/2) \tag{8.5}$$

which is the same as that in [24]. This is because when we are sending data, we use function MPI_Pack and MPI_UnPack, which can pack all the sending data together and unpack them after received. Therefore, as long as the size of the tree is same, the number of the communication will not change.

If we compute the communication of American-style Asian options with 10 underlying assets, the communication should be 10 times as the single asset Asian options.

8.2 Experimental Results

For the implementation, C programming language and MPI are used on a distributed memory architecture. The algorithms were tested on one, two, four, eight, and sixteen processors. These workstation have Pentium II processors with 350MHz of CPU speed and 512KB cache. The memory in these machines is 256MB. MPI (Message Passing Interface) is a library specification for message-passing, proposed as a standard by a broadly based committee of vendors, implementors, and users. The complete details on MPI is available on *http://www.lam-mpi.org.* In our experiments, the pricing problem is

Table 8.3: Input Parameters for American-style Asian Put

Start Price S=25.0
Strike Price K=30.0
Option's life time T=1.0 (1 year)
Interest rate r=12%
Volatility σ=12%

parameterized as shown in Table 8.3.

8.2.1 Experimental results for Non-recombining Binomial Tree Method

First, we implemented the sequential algorithm for American-style Asian option on the non-recombining tree. Figure 8.3 shows the performance results of the non-recombining

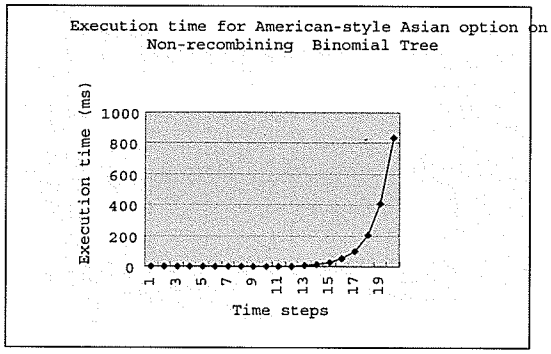


Figure 8.3: Execution time of algorithm on non-recombining binomial tree structure

algorithm for up to 20 levels. The maximum level in Figure 8.3 is 20 because the structure of the non-recombining tree causes a memory problem. It can be observed that the number of the leave nodes in the non-recombining tree increases exponentially. For example, there are 2 nodes on level 1, 4 nodes on level 2, 8 nodes on level 3 and consequently 2^i

nodes on level i. This exponential increase causes a memory limitation since on level 20, there are 2^{20} leave nodes, see [24] for detail. That is, we cannot decrease the step size beyond certain level without having to overcome the memory problems. To avoid this situation, we consider recombining trees. Therefore, we will focus on discussing the experimental results of the recombining algorithm.

8.2.2 Experimental results for Recombining Binomial Tree Method

In this chapter, first, we will discuss the sequential experimental results of the algorithm that employs the technique VTT. Next, we will discuss the experimental results of the parallel algorithms on 2, 4, 8 and 16 processors. Finally, we will present the experimental results of the comparison between our algorithm and some similar work done previously [24].

Sequential Algorithm with VTT

Virtual Tree Transformation (VTT) is discussed in Chapter 6 which may improve the performance. In order to see how it works, we implemented two sequential algorithms for American-style Asian option with 10 assets, in which one employs VTT technique while the other does not employ the technique. The experimental result is shown in Figure 8.4.

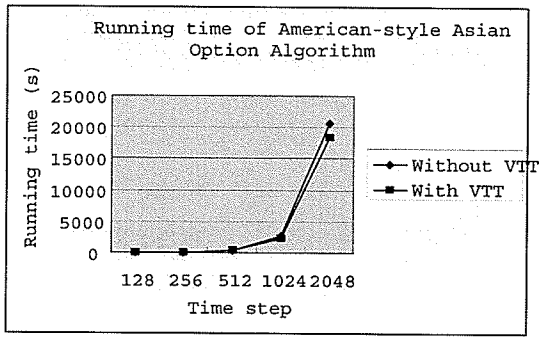


Figure 8.4: Execution time of sequential algorithms with and without VTT

It can be observed that since time step 1024 the execution time of the algorithm that employs VTT is less than the algorithm without employing VTT. The improvement is not very obvious when the time step is less than 1024. This could be explained that the workload is not big enough so we cannot see big improvement. However, from the trends of these two lines, we can infer that with the increase of the time step, the algorithm employing VTT will produce better performance than the algorithm without employing VTT.

Parallel Algorithm with up to 10 assets

In this chapter, we are going to discuss the experimental results for parallel algorithms with up to 10 assets employing technique VTT.

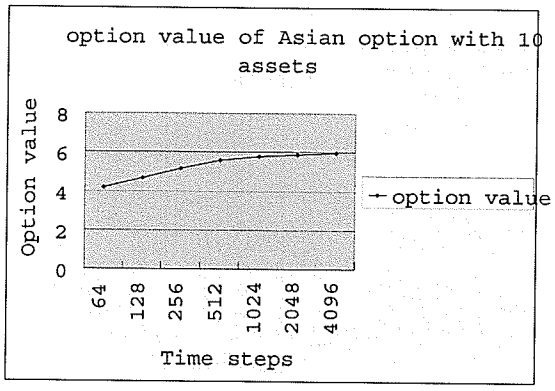


Figure 8.5: Option value of American-style Asian option with 10 underlying assets

Figure 8.5 is the experimental result of the potential maximum option value of the American-style Asian option with 10 underlying assets. By potential maximum option value, we mean the maximum option value in the whole life of the option since American-style options have the possibilities that can be exercised at any time during the whole life of the option. It is observed that after a time period, the option value tends to reach a certain value, which means this algorithm converges. A theoretical treatment of convergence of the option price computed under the binomial tree to the true option

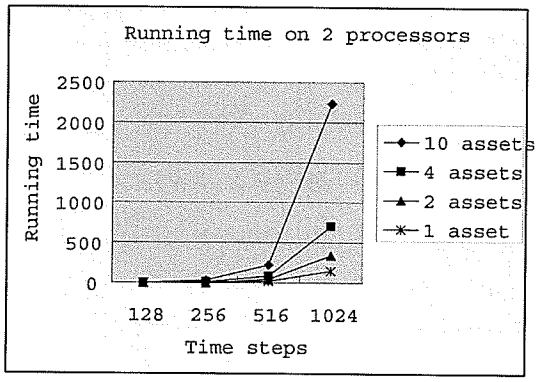


Figure 8.6: Execution time on two processors with asset number 1,2,4,10

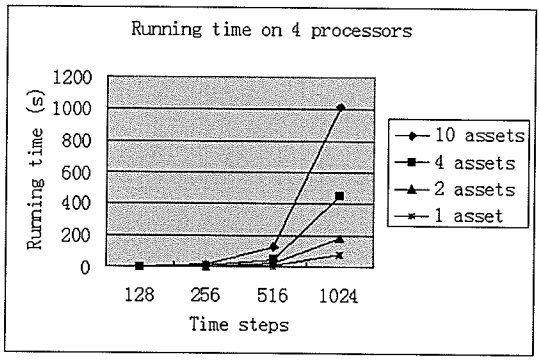


Figure 8.7: Execution time on four processors with asset number 1,2,4,10

price can be found in [12].

Figures 8.6 to 8.9 are the experimental results on the execution time for various number of processors with various asset numbers from time step 128 to 1024. We can see from these four figures that when the number of processors is decided, the execution time increases greatly with the increase of time steps. The execution time for 10 assets increases much more than the execution time of smaller number of assets. This is due to the huge increase of computation for 10 assets with the increase of time steps as explained earlier. Figure 8.10 is the result on the execution time of various number of assets with 1024 time steps. It is observed that with the increase in number of processors,

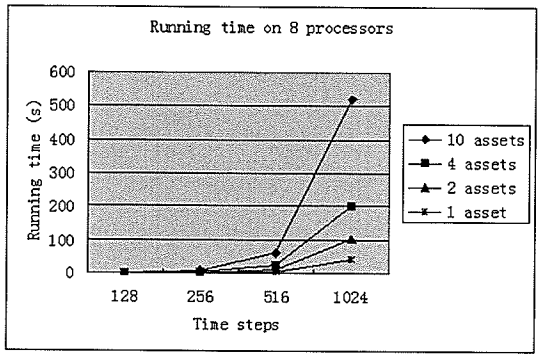


Figure 8.8: Execution time on eight processors with asset number 1,2,4,10

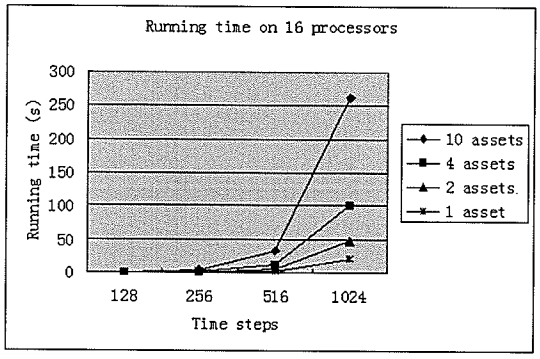


Figure 8.9: Execution time on sixteen processors with asset number 1,2,4,10

the execution time of options with ten assets decreases much more sharply than options with smaller number of assets. This could be explained that for large options (that is, options with many underlying assets) the larger problem size offers a better problem for parallel computation.

In order to do the comparison, we also implemented the parallel algorithm for American-style plain options with up to 10 assets. Figure 8.11 is the experimental result of the parallel algorithm for plain options. Figure 8.12 is the execution time compared between American-style Asian option and American-style plain option. From Figure 8.12, it can be figured out easily that compared with the execution time of American-style Asian

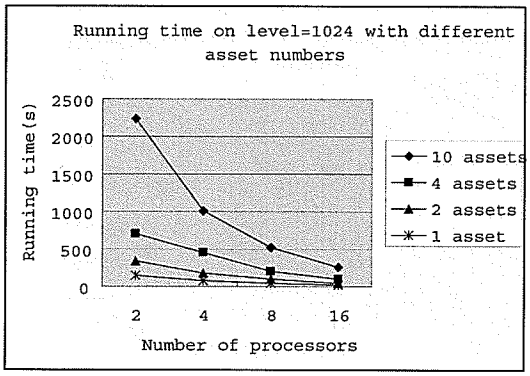


Figure 8.10: Execution time on level 1024 with asset number 1,2,4,10

option, the execution time of American-style plain option is really less, which can be correlated with the analytical results we derived in Chapter 8.1.

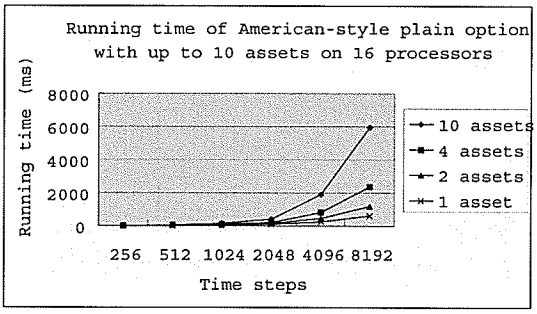


Figure 8.11: Execution time of parallel algorithm for American-style plain option with 10 underlying assets on 16 processors

Figure 8.13 is the experimental result on the speedup for varying number of processors with 10 underlying assets. It can be observed from Figure 8.13 that we have a very good speedup from our experiments. For example, with 2 processors the execution time is 2235.97 seconds and with 16 processors it is 261.73 seconds. The speedup ($S = \frac{T_S}{T_P}$, where T_S is the execution time with sequential algorithm and T_P is the execution time with parallel algorithm) is 1.8 with 2 processors and 15.38 with 16 processors at time step

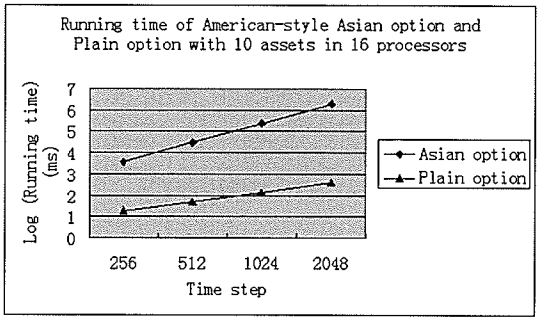
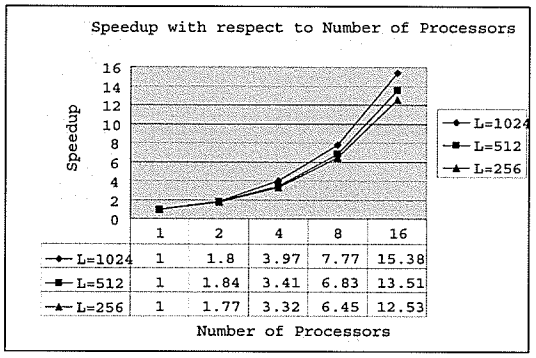


Figure 8.12: Execution time compared between American-style Asian option & plain option with 10 underlying assets on 16 processors



	1	2	4	8	16
L=1024	1	1.8	3.97	7.77	15.38
L=512	1	1.84	3.41	6.83	13.51
L=256	1	1.77	3.32	6.45	12.53

Figure 8.13: Speedup on level 1024 for option with 10 assets

1024. The number of communications in the parallel algorithm remained same compared to the plain option [24]. However, as we have explained in the analytical results section, the computation complexity of the Asian option is much higher than the vanilla option. This gives rise to good work load for all the processors and hence the speedup obtained is much better than with the vanilla option. In other words, our analytical observation of higher computation complexity of Asian option could also be explained through speedup chart.

8.2.3 Algorithm Performance Comparison

A previous work studied the parallel algorithm of American-style plain options [24]. In order to see if our coding of the parallel algorithm for the American-style Asian options is optimized, we modified the parallel algorithm of American-style plain options in [24] into parallel algorithm for American-style Asian options (hereafter called *AP algorithm*), and compare with our own parallel algorithm for American-style Asian options (hereafter called *AA algorithm*).

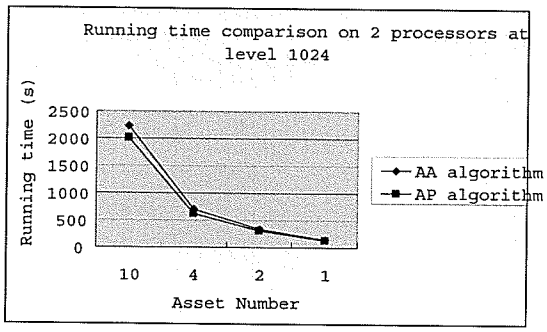


Figure 8.14: Execution time comparison on 2 processors at level 1024

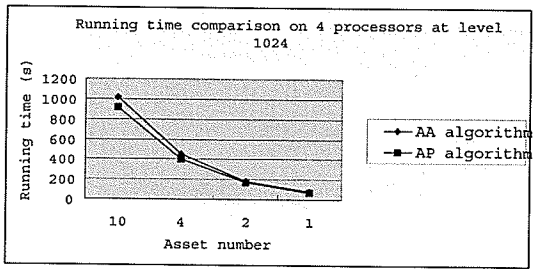


Figure 8.15: Execution time comparison on 4 processors at level 1024

Figure 8.14 to Figure 8.18 is the comparison of the execution time between the AA algorithm and the AP algorithm at the level 1024. It can be observed that the AP algorithm performed a little bit better than AA algorithm. This could be explained

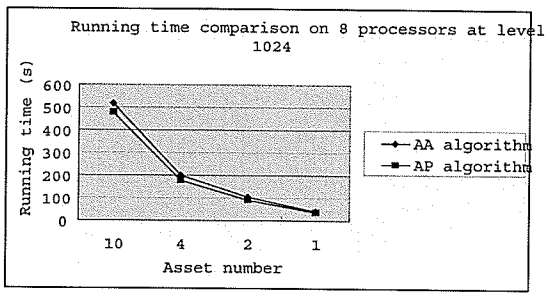


Figure 8.16: Execution time comparison on 8 processors at level 1024

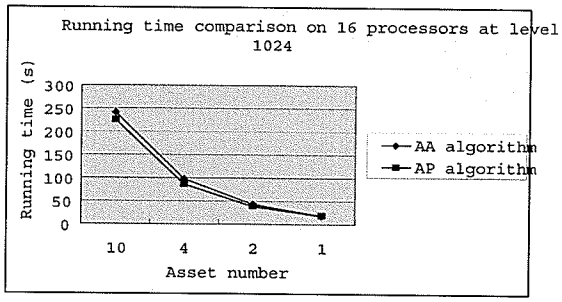


Figure 8.17: Execution time comparison on 16 processors at level 1024

that the AP algorithm was optimized with proper MPI constructs and hence the AA algorithm requires some optimization, which is left as a future work.

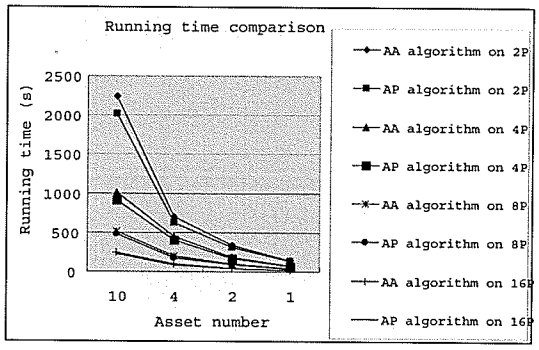


Figure 8.18: Execution time comparison at level 1024

Chapter 9

Conclusions

The Asian option is a kind of path-dependent derivatives and popularly traded in many exchanges worldwide. How to price Asian options efficiently and accurately has been a critical problem for both academics and practitioners alike.

This thesis addresses the Multi-dimensional American-style Asian option pricing problem with the parallel algorithm development to meet the efficiency and accuracy requirements. First, a mathematical transformation, Virtual Tree Transformation (VTT) is developed, which reflects the dependent relationship among multi-assets. The technique VTT can eliminate the construction of tree, thereby improving the overall performance. Second, by employing VTT a parallel algorithm is developed. We exploit the inherent parallelism in the binomial lattice method and discuss the strategy we adopt in implementing the algorithm in a distributed computing environment. We have presented a detailed analysis of timing complexity of our parallel algorithm and validated it with experimental results. We compared the computation of our algorithm for American-style Asian options with that for American-style plain options and showed that the computation requirement of Asian options is much more than that of the plain options, which is correlated with our analytical results in Chapter 8.1. This could explain that even with a small number of time steps, the execution time still decreases with the increase of number of processors, which means the computation is more predominant than the

communication. The speedups attained algorithm illustrate clearly that large portfolios of long-dated options can be rapidly priced. Part of these conclusions are reported in [15].

Chapter 10

Future Work

There are some future work which can make this research on Asian option pricing problem more complete.

First, implementing the algorithm on shared memory architecture is a challenge which may avoid the communication latency in the distributed environment. In shared memory architecture the communication latency is less pronounced depending on the design of the algorithm. Therefore, we can expect better performance. OpenMP could be adopted for the shared memory implementation. In addition, a new parallel programming environment mpC also could be employed and the comparison could be made among the performances in these different environments.

Another possible future work is about the maximum and minimum paths. Recall that we suppose the average value is the mean value of the maximum and minimum paths of every node in the binomial tree. But in fact, a more realistic situation is that the average value comes from the mean value of all or part of all the paths which reach the target node from the root of the tree. This will increase the computational intensity of the problem.

Recall that in Chapter 8.2.3, we found out that our algorithm could be improved for better performance if we make some coding optimization. Hence, optimization could be another possible future work as well.

Bibliography

[1] J. Barraquand and T. Pudet. Pricing of American path-dependent contingent claims. *Mathematical Finance*, 6:17–51, 1996.

[2] E. Benhamou. Fast Fourier transform for discrete Asian options. *Journal of computational finance*, 6(1):49–68, 2002.

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

[4] P. Boyle. Options: A Monte Carlo approach. *Journal of Financial Economics*, 4:323–338, May 1977.

[5] P. Boyle, M. Broadie, and P. Glasserman. Monte Carlo methods for security pricing. *Journal of Economic Dynamics and Control*, 21(1267-1321), 1997.

[6] M. Broadie and P. Glasserman. Estimating security price derivatives using simulation. *Management Science*, 42:269–285, 1996.

[7] M. Broadie, P. Glasserman, and S. Kou. Connecting discrete and continuous path-dependent options. *Finance and Stochastics*, 3:55–82, 1999.

[8] P. Chalasani, S. Jha, and A. Varikooty. Accurate approximations for European-style Asian options. *Journal of Computational Finance*, 1998.

[9] J. Cox, S. Ross, and M. Rubinstein. Option pricing: A simplified approach. *Journal of Financial Economics*, 7:229–263, 1979.

[10] M. Craddock, D. Heath, and E. Platen. Numerical inversion of laplace transforms: A survey of techniques with applications to derivative pricing. *Journal of Computational Finance*, 4:57–81, 2000.

[11] T. Dai and Y. Lyuu. An exact subexponential-time lattice algorithm for Asian options. In *Proceedings of the fifteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 710–717, New Orleans, Louisiana, 2004.

[12] D. Duffie. *Dynamic Asset Pricing Theory.* Princeton University Press, Princeton, NJ., 1996.

[13] M. Fu, D. Dilip, and T. Wang. Pricing continuous Asian options: a comparison of Monte Carlo and laplace transform inversion methods. *Journal of Computational Finance*, 2(2):49–74, 1998.

[14] H. Geman and M. Yor. Bessel processes, Asian options, and perpetuities. In *Mathematical Finance*, volume 3, pages 349–375, 1993.

[15] K. Huang and R. Thulasiram. Parallel algorithm for pricing American Asian options with multi-dimensional assets. In *Proceedings of the 19th Annual Symposium on High Performance Computing Systems and Applications (HPCS 2005)*, Guelph, Ontario, Canada, May 15-18, 2005.

[16] J. Hull. *Options, Futures, and Other Derivative Securities.* Prentice Hall, Upper Saddle River, NJ, 5th edition, 2002.

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

[18] F. Longstaff and E. Schwartz. Valuing American options by simulation: A simple least squares approach. *The Review of Financial Studies*, 14(1):113–147, 2001.

[19] M. Mascagni and H. Chi. Optimal quasi-Monte Carlo valuation of derivative securities. In *Computational Finance and its Applications*, pages 177–185. WIT Press, 2004.

[20] M. Milevsky and S. Posner. Asian options, the sum of lognormals, and the reciprocal gamma distribution. *Journal of Financial and Quantitative Analysis*, 33(3):409–422, 1998.

[21] L. Rogers and Z. Shi. The value of an Asian option. *Journal of Applied Probability*, 32:1077–1088, 1995.

[22] W. Shaw. *Modelling Financial Derivatives with Mathematica.* Cambridge University Press, Cambridge U.K., 1998.

[23] A. Srinivasan. Parallel and distributed computing issues in pricing financial derivatives through quasi Monte Carlo. In *In Proc. (CD-ROM) of the International Parallel and Distributed Processing Symposium (IPDPS)*, April 2002.

[24] R. Thulasiram and D. Bondarenko. Performance evaluation of parallel algorithm for pricing multidimensional financial derivatives. In *IEEE Computer Society Proceedings of the Fourth International Workshop on High Performance Scientific and Engineering Computing with Applications*, pages 306–313, Vancouver, BC, Canada., August 2002. (International Journal of Computational Science and Engineering - to appear).

[25] R. Thulasiram, L. Litov, H. Nojumi, C. Downing, and G. Gao. Multithreaded algorithms for pricing a class of complex options. In *Proceedings (CD-ROM) of the IEEE/ACM International Parallel and Distributed Processing Symposium (IPDPS)*, San Francisco, CA, USA, April 2001.

[26] S. Turnbull and L. Wakeman. A quick algorithm for pricing European average options. In *Financial and Quantitative analysis*, volume 26, pages 377–389, 1991.

[27] T. Vorst. *The Handbook of Exotic Options.* IRWIN Professional Publishing, Chicago, 1996.

[28] P. Wilmott, J. Dewynne, and S. Howison. *Option Pricing: Mathematical Models and Computation.* Oxford Financial Press, 1993.

[29] J. Zhang. Method for pricing and hedging continuously sampled average rate options. *Journal of Computational Finance*, 5(1):59–79, 2001.