Computational Techniques in Derivatives Pricing

Yuh-Dauh Lyuu
Computer Science & Information Engineering
National Taiwan University

March/22/2000 by LATEX

Outline

1. Computational complexity
2. Trade against the central bank
3. Derivatives pricing with combinatorics
4. The differential tree approach to model calibration
5. Monte Carlo pricing
6. Path-dependent options pricing
7. Looking into the future

References

- Yuh-Dauh Lyuu, *Financial Engineering and Computation: Principles, Mathematics, Algorithms*, Cambridge University Press, 2000.
- `www.csie.ntu.edu.tw/~lyuu/Capitals/capitals.htm`
- Other published and unpublished papers

When Professors Scholes and Merton and I
invested in warrants,
Professor Merton lost the most money.
And I lost the least.

—Fischer Black

Part 1:
Computational Complexity

It is unworthy of excellent men
to lose hours like slaves
in the labor of computation.
—Leibniz

Measures of Complexity

1. Time
   - **Tractable**: "solvable" in *polynomial* time such as $O(n)$ and $O(n^2)$
   - **Intractable**: otherwise
     - Candidates: Asian options & certain reset options
     - Approaches: analytical approximations, approximation algorithms, Monte Carlo simulation, etc.
2. Memory
   - Maybe an issue for long-dated fixed-income securities or path-dependent derivatives

Part 2:
Trade against the Central Bank

Competitive Analysis

- The trader wants to trade USD for JPY (say)
  - Applicable to any assets with relative prices
- n exchange rates will be revealed
- The trader acts on each exchange rate
- Converting JPY back to USD is not allowed (**buy-and-hold** only)
- Goal: maximize the total JPY amount on day n as compared against the adversary with complete foresight
  - This adversary trades *once*, at the highest rate
- Result is (almost) model-free (no distribution assumptions) and therefore more robust

Trader's Dilemma

- Convert too little and future exchange rates go down
- Convert too much and future exchange rates go up

Competitive Performance

- A trading algorithm $\mathcal{A}$ is c-*competitive* if for any rate sequence, it guarantees a JPY amount at least $1/c$ of the adversary's amount; i.e.,

$$E[\mathcal{A}] \geq \frac{\text{OPT}}{c}$$

  - OPT trades all its USD for JPY at the highest exchange rate, which is known to the adversary
  - $c \geq 1$; the lower the better

- The least c that $\mathcal{A}$ achieves is called its **competitive ratio**

The Model

- Geometric upper and lower bounds
  - If the current rate is r, the next is $\in [\, r/\theta, r\theta \,]$
  - $\theta \approx 1.07$ for the Taipei Stock Exchange
  - Results available for the general $[\, r/\alpha, r\beta \,]$ case
- Related to the popular *lognormal* process (*geometric Brownian motion*) used in finance [Hull 1999]

The Optimal Buy-and-Hold Trading Strategy

- The optimal strategy per USD:
  - Invest $\frac{\theta}{n\theta-(n-2)}$ dollar on the first and last days
  - Invest $\frac{\theta-1}{n\theta-(n-2)}$ dollar on the other days
- Achieves the optimal competitive ratio any algorithm can attain: $\frac{n\theta-(n-2)}{\theta+1}$ [Chen, Kao, Lyuu, Wong 1999]
- Beat the popular **dollar-averaging** strategy, whose competitive ratio is $\frac{n(1-\theta^{-1})}{1-\theta^{-n}}$
- Indirect support for the soundness of dollar-averaging strategy

Part 3:
Derivatives Pricing with Combinatorics

The shift toward options as
the center of gravity of finance [. . .]
—Merton H. Miller

Listed Futures and Futures Options, 1997–1998

Name	Monthend open interest	Trading volume	Contracts settled
Futures contracts			
Chicago Board of Trade (CBT)			
Dow Jones Industrial Index	14,494	3,505,262	31,293
Treasury bonds	838,403	114,945,293	55,595
Total CBT	2,602,372	218,204,974	556,213
Chicago Mercantile Exchange (CME) and IMM			
S&P 500 Index	372,542	30,698,445	369,072
3-month Eurodollar	2,961,562	107,386,746	1,556,484
Total CME/IMM	4,191,618	181,051,919	3,179,971
Total all exchanges	8,732,915	500,562,510	4,186,906
Futures options			
Chicago Board of Trade (CBT)			
Dow Jones Industrial Index	39,706	354,094	
Treasury bonds	959,597	37,947,756	
Total CBT	2,398,298	61,369,819	
Chicago Mercantile Exchange (CME) and IMM			
S&P 500 Index	274,655	5,049,771	
3-month Eurodollar	3,064,612	31,842,995	
Total CME/IMM	3,779,892	42,172,666	
Total all exchanges	8,073,479	124,107,563	

Calls and Puts

- $S_0, S_1, \ldots, S_n$ denote the prices of the underlying asset
- The call option has a terminal payoff given by

$$\max(S_n - X, 0)$$

- The put option has a terminal payoff given by

$$\max(X - S_n, 0)$$

- Variations
- Backward induction

Binomial Models

- Stock price can go from S to Su with probability p or Sd with probability $1-p$ in a period
  - The Cox-Ross-Rubinstein (CRR) version:

$$\begin{aligned} u &= e^{\sigma\sqrt{\Delta t}} \\ d &= 1/u \\ p &= (e^{r\Delta t} - d)/(u-d) \end{aligned}$$

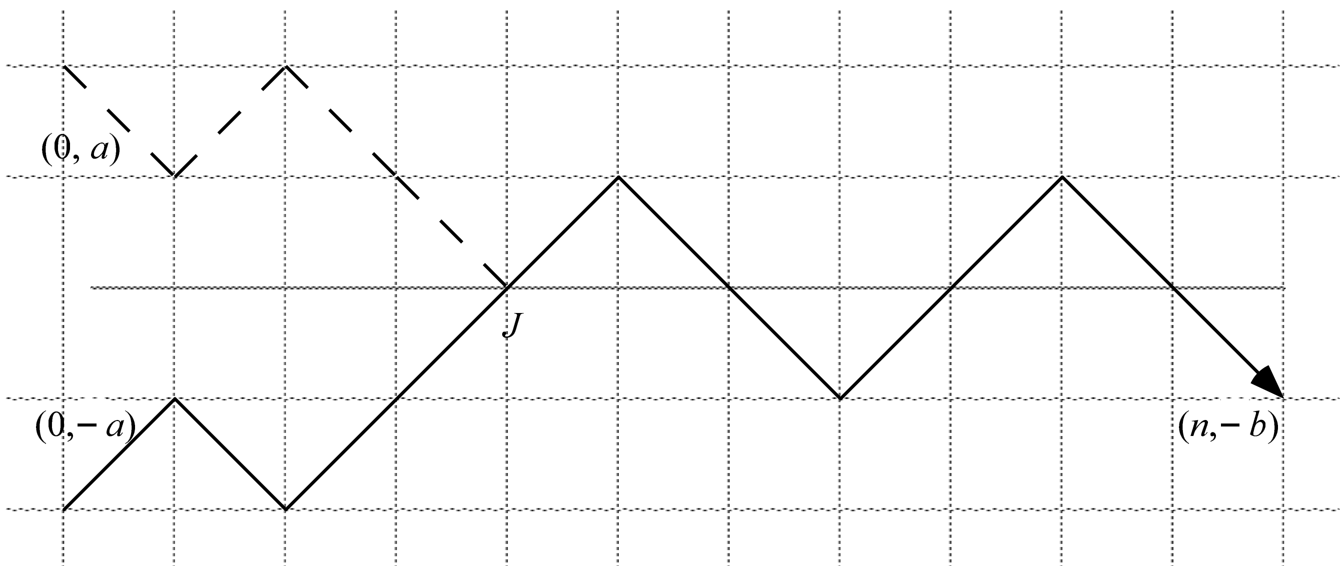
  - The Jarrow-Rudd (JR) version:

$$\begin{aligned} u &= e^{(r-\sigma^2/2)\,\Delta t + \sigma\sqrt{\Delta t}} \\ d &= e^{(r-\sigma^2/2)\,\Delta t - \sigma\sqrt{\Delta t}} \\ p &= 1/2 \end{aligned}$$

Barrier Option Pricing

- Standard backward induction takes time $O(n^2)$
- Solving the Black-Scholes differential equation takes $O(n^2)$ time
- Combinatorics cuts the time to $O(n)$
  - Shortcoming: cannot handle American options
- **A rule of thumb**: pricing European options is faster than pricing American options by an order of magnitude
  - Mathematically true?

The Reflection Principle for Binomial Random Walk



The Reflection Principle

- Imagine a particle at position $(0, -\boldsymbol{a})$ on the integral lattice that is to reach $(n, -\boldsymbol{b})$, where $\boldsymbol{a}, \boldsymbol{b} \geq 0$
- How many paths touch the x-axis?
- Answer:

$$\binom{n}{\frac{n+\boldsymbol{a}+\boldsymbol{b}}{2}} \quad \text{for even } n + \boldsymbol{a} + \boldsymbol{b}$$

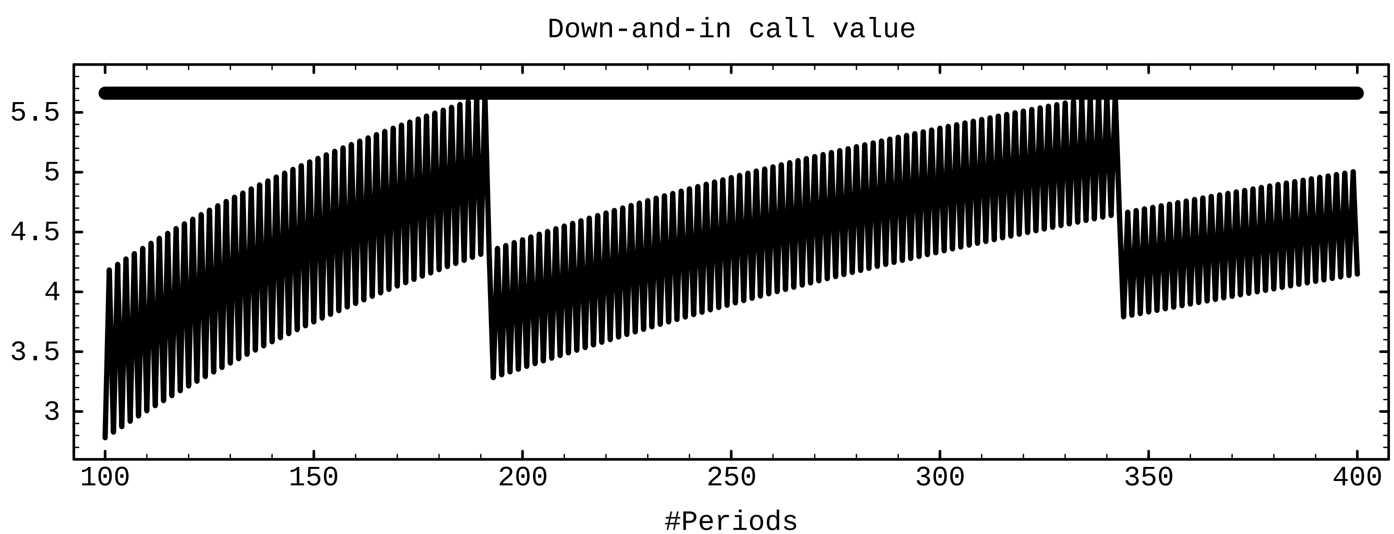
Single-Barrier Options

- We focus on the down-and-in call with barrier $H < X$
  - Knocked in if the barrier is touched
  - Assume $H < S$ without loss of generality
- Let

$$a \equiv \left\lceil \frac{\ln(X/S)}{2\sigma\sqrt{\Delta t}} + \frac{n}{2} \right\rceil \text{ and } h \equiv \left\lfloor \frac{\ln(H/S)}{2\sigma\sqrt{\Delta t}} + \frac{n}{2} \right\rfloor$$

  - $\tilde{H} \equiv S u^h d^{n-h}$ is the new barrier
  - $\tilde{X} \equiv S u^a d^{n-a}$ is the new strike price
- May introduce fluctuations as well

Convergence of the Binomial Model



The analytical value is 5.6605

The Combinatorial Formula

- Each path from S to the terminal price $Su^j d^{n-j}$ has probability $p^j (1-p)^{n-j}$ of occurring
- There are $\binom{n}{j}$ paths, and $\binom{n}{n-2h+j}$ of them hit $\tilde{H}$
- So the terminal price $Su^j d^{n-j}$ is reached by a path that hits the barrier with probability $\binom{n}{n-2h+j} p^j (1-p)^{n-j}$
- The option value equals

$$e^{-r\tau} \sum_{j=a}^{2h} \binom{n}{n-2h+j} p^j (1-p)^{n-j} \left(Su^j d^{n-j} - X \right)$$

  – Can be summed in $O(n)$ steps

Compared with the Trinomial Model (in milliseconds)

n	Combinatorial method		Trinomial tree algorithm a.k.a. Ritchken (1995)	
	Value	Time	Value	Time
21	5.507548	0.30		
84	5.597597	0.90	5.634936	35.0
191	5.635415	2.00	5.655082	185.0
342	5.655812	3.60	5.658590	590.0
533	5.652253	5.60	5.659692	1440.0
768	5.654609	8.00	5.660137	3080.0
1047	5.658622	11.10	5.660338	5700.0
1368	5.659711	15.00	5.660432	9500.0
1731	5.659416	19.40	5.660474	15400.0
2138	5.660511	24.70	5.660491	23400.0
2587	5.660592	30.20	5.660493	34800.0
3078	5.660099	36.70	5.660488	48800.0
3613	5.660498	43.70	5.660478	67500.0
4190	5.660388	44.10	5.660466	92000.0
4809	5.659955	51.60	5.660454	130000.0

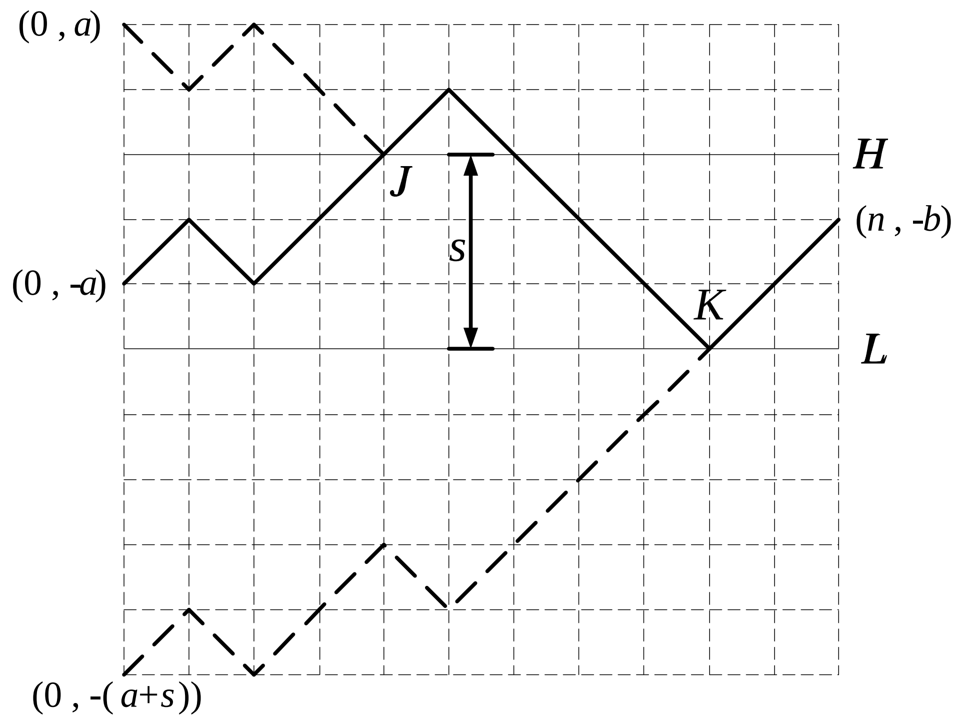
Analytical value 5.6605; 100 MHz Intel Pentium processor and 32 MB of DRAM, running Windows NT 4.0

When the Current Stock Price Is Near the Barrier

- Some claimed it makes the binomial model impractical:
  - n will have to be very large to tackle fluctuations
  - But then the n^2 bound becomes too high
- No problem if we use an $O(n)$-time algorithm

Barrier at 95.0			Barrier at 99.5			Barrier at 99.9		
n	Value	Time	n	Value	Time	n	Value	Time
	⋮		795	7.47761	8.0	19979	8.11304	253.0
2743	2.56095	31.1	3184	7.47626	38.0	79920	8.11297	1013.0
3040	2.56065	35.5	7163	7.47682	88.0	179819	8.11300	2200.0
3351	2.56098	40.1	12736	7.47661	166.0	319680	8.11299	4100.0
3678	2.56055	43.8	19899	7.47676	253.0	499499	8.11299	6300.0
4021	2.56152	48.1	28656	7.47667	368.0	719280	8.11299	8500.0
4378	2.56095	53.0	39003	7.47674	500.0	979019	8.11299	11800.0
	2.5615			7.4767			8.1130	

The Reflection Principle—Iterated



Must hit both barriers (an L-hit preceded by an H-hit)

Reflect the path first at J and then at K

Double Barrier Options

- Double barrier options contain two barriers L and H with $L < H$
- Consider options that come into existence if and only if *either* barrier is hit (knock-in type)
- The number of paths in which a hit of the H-line ($x = 0$) appears before a hit of the L-line ($x = -s$) is

$$\binom{n}{\frac{n+\boldsymbol{a}-\boldsymbol{b}+2s}{2}} \quad \text{for even } n + \boldsymbol{a} - \boldsymbol{b}$$

The Combinatorial Pricing Formula

- L^+ denotes a sequence of Ls, and H^+ a sequence of Hs
- Let A_i denote the set of paths that hit the barriers with a hit sequence containing $\overbrace{H^+ L^+ H^+ \cdots}^{i}$, $i \geq 2$
- Let B_i denote the set of paths that hit the barriers with a sequence containing $\overbrace{L^+ H^+ L^+ \cdots}^{i}$, $i \geq 2$
- The number of paths that hit either barrier equals

$$\boldsymbol{N}(\boldsymbol{a}, \boldsymbol{b}, s) = \sum_{i=1}^{n} (-1)^{i-1} (|\, A_i \,| + |\, B_i \,|)$$

- The running time is $O(n)$

The Combinatorial Pricing Formula (continued)

$$|A_i| = \begin{cases} \dbinom{n}{\frac{n+\boldsymbol{a}+\boldsymbol{b}+(i-1)\,s}{2}} & \text{for odd } i \\ \dbinom{n}{\frac{n+\boldsymbol{a}-\boldsymbol{b}+is}{2}} & \text{for even } i \end{cases}$$

$$|B_i| = \begin{cases} \dbinom{n}{\frac{n-\boldsymbol{a}-\boldsymbol{b}+(i+1)\,s}{2}} & \text{for odd } i \\ \dbinom{n}{\frac{n-\boldsymbol{a}+\boldsymbol{b}+is}{2}} & \text{for even } i \end{cases}$$

The Combinatorial Pricing Formula (continued)

- Define

$$h \equiv \left\lceil \frac{\ln(H/S)}{2\sigma\sqrt{\Delta t}} + \frac{n}{2} \right\rceil \quad l \equiv \left\lfloor \frac{\ln(L/S)}{2\sigma\sqrt{\Delta t}} + \frac{n}{2} \right\rfloor$$

  – The barriers are replaced by the barriers $\tilde{H} \equiv Su^h d^{n-h}$ and $\tilde{L} \equiv Su^l d^{n-l}$

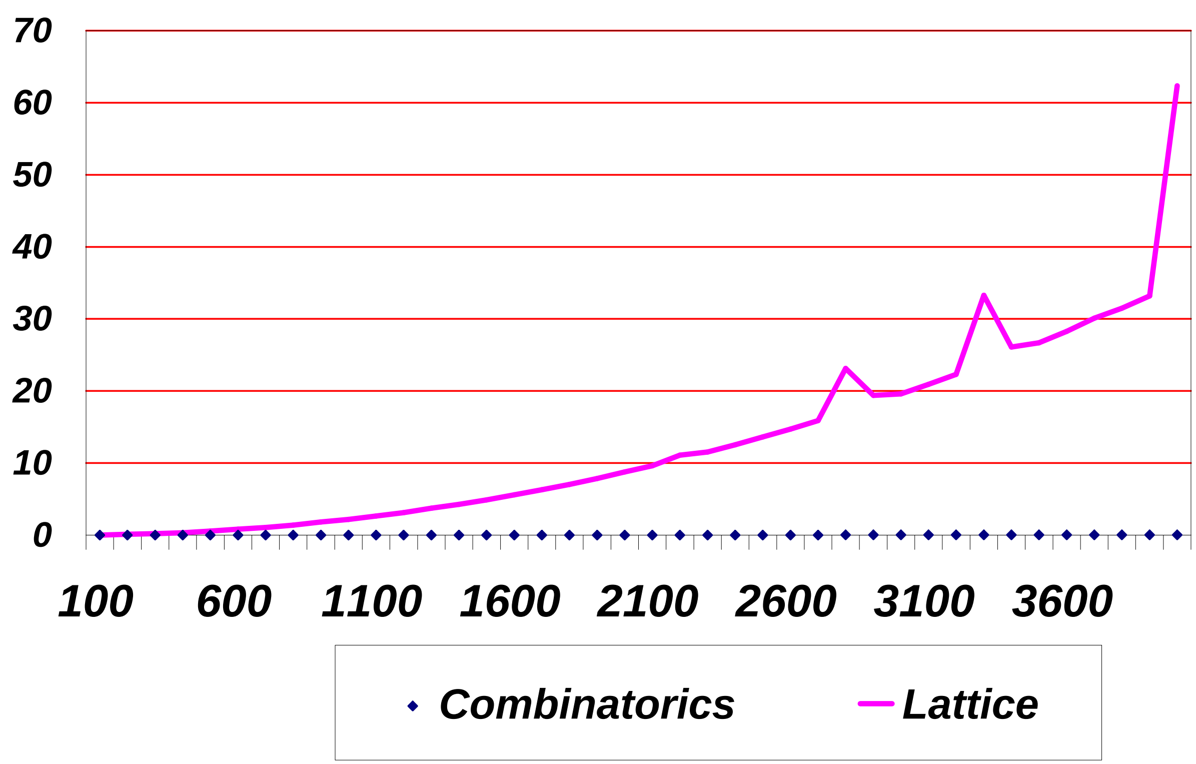
- The terminal nodes between $\tilde{L}$ and $\tilde{H}$ (inclusive) together contribute

$$e^{-r\tau} \sum_{j=a}^{h} \boldsymbol{N}(2h-n, 2h-2j, 2(h-l))\, p^j (1-p)^{n-j} (Su^j d^{n-j} - X)$$

  to the option value

- The terminal nodes outside the above-mentioned range constitute a standard call; add this to the above

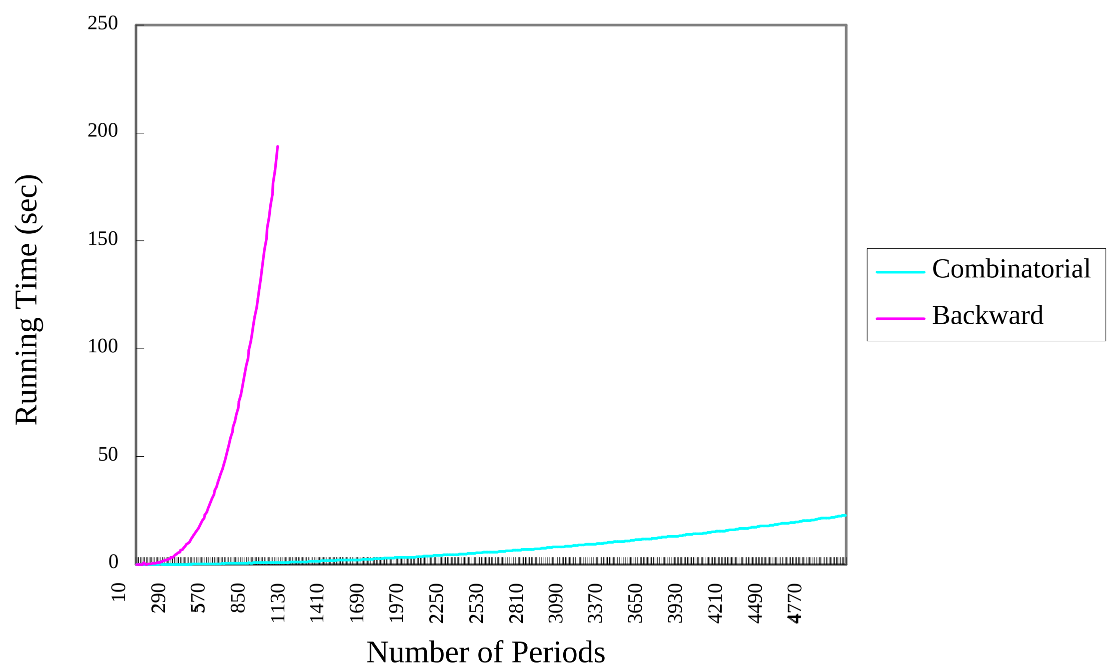
Comparison with Backward Induction (in seconds)



Lookback Option

Payoff is $\max(S_n - \min_i S_i, 0)$

Figure <1> Comparison of combinatorial and backward methods in running time



Pricing Geometric Asian Options

- $S_0, S_1, \ldots, S_n$ denote the prices of the underlying asset
- The Asian call has a terminal payoff given by

$$\max((S_0 S_1 \cdots S_n)^{\frac{1}{n+1}} - X, 0)$$

- Can be priced in time $O(n^4)$ using backward induction
- Needed in some approximation algorithms and control variates approach for pricing *arithmetic* Asian options

The Combinatorial Approach

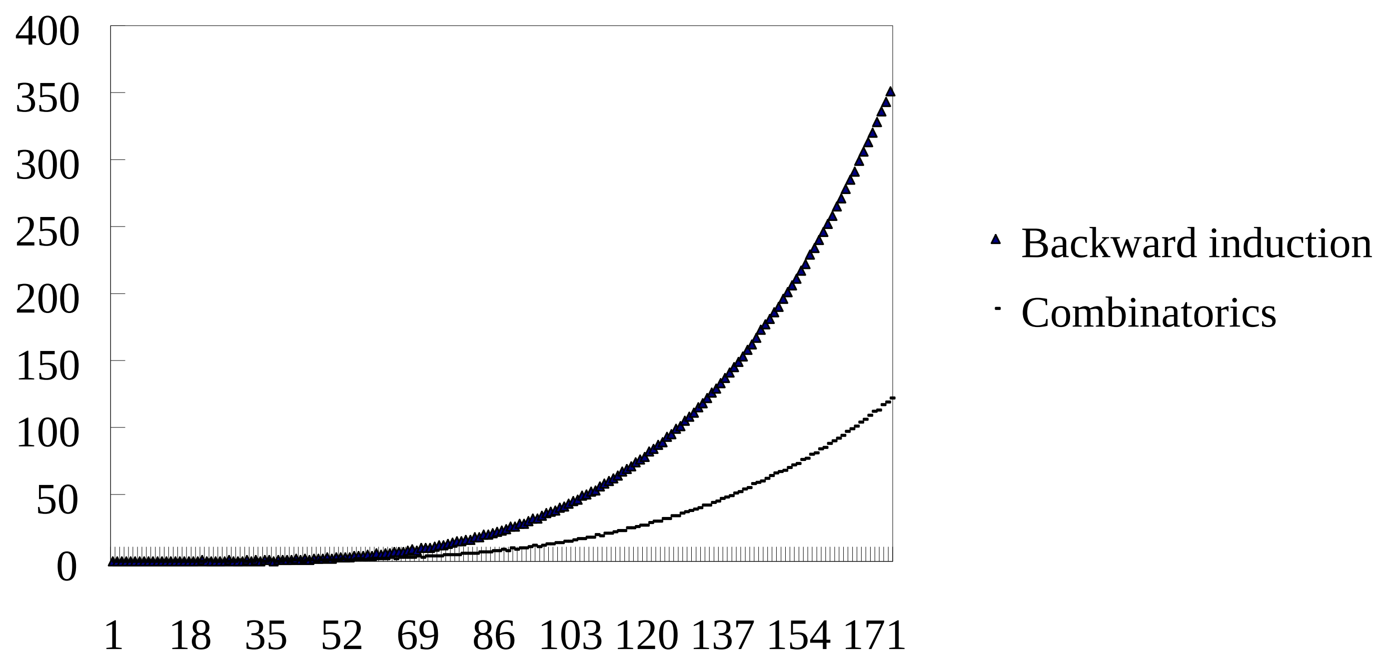
- Use the Jarrow-Rudd binomial model
  - Each move has identical probability 1/2
- Computable in time $O(n^3)$ (recall the rule of thumb)
- Define $\boldsymbol{q(0)}, \boldsymbol{q(1)}, \ldots$ with

$$(1+x)(1+x^2)(1+x^3)\cdots(1+x^n) = \sum_{m=0}^{n(n+1)/2} \boldsymbol{q(m)}\, x^m$$

- Value is then

$$e^{-r\tau} \sum_{m=0}^{n(n+1)/2} 2^{-n} \boldsymbol{q(m)} \max(S(u^m d^{n(n+1)/2-m})^{\frac{1}{n+1}} - X, 0)$$

Comparison with Backward Induction (in seconds)



Part 4:
The Differential Tree Approach to Model Calibration

The fox often ran to the hole
by which they had come in,
to find out if his body was still thin enough
to slip through it.
—*Grimm's Fairy Tales*

Outstanding U.S. Debt Market Securities (bln)

Year	Municipal	Treasury	Agency MBSs	U.S. corporate	Fed agencies	Money market	Asset — backed
1985	859.5	1,360.2	372.1	719.8	293.9	847.0	2.4
1986	920.4	1,564.3	534.4	952.6	307.4	877.0	3.3
1987	1,010.4	1,724.7	672.1	1,061.9	341.4	979.8	5.1
1988	1,082.3	1,821.3	749.9	1,181.2	381.5	1,108.5	6.8
1989	1,135.2	1,945.4	876.3	1,277.1	411.8	1,192.3	59.5
1990	1,184.4	2,195.8	1,024.4	1,333.7	434.7	1,156.8	102.2
1991	1,272.2	2,471.6	1,160.5	1,440.0	442.8	1,054.3	133.6
1992	1,302.8	2,754.1	1,273.5	1,542.7	484.0	994.2	156.9
1993	1,377.5	2,989.5	1,349.6	1,662.1	570.7	971.8	179.0
1994	1,341.7	3,126.0	1,441.9	1,746.6	738.9	1,034.7	205.0
1995	1,293.5	3,307.2	1,570.4	1,912.6	844.6	1,177.2	297.9
1996	1,296.0	3,459.0	1,715.0	2,055.9	925.8	1,393.8	390.5
1997	1,367.5	3,456.8	1,825.8	2,213.6	1,022.6	1,692.8	518.1
1998	1,464.3	3,355.5	2,018.4	2,462.0	1,296.5	1,978.0	632.7

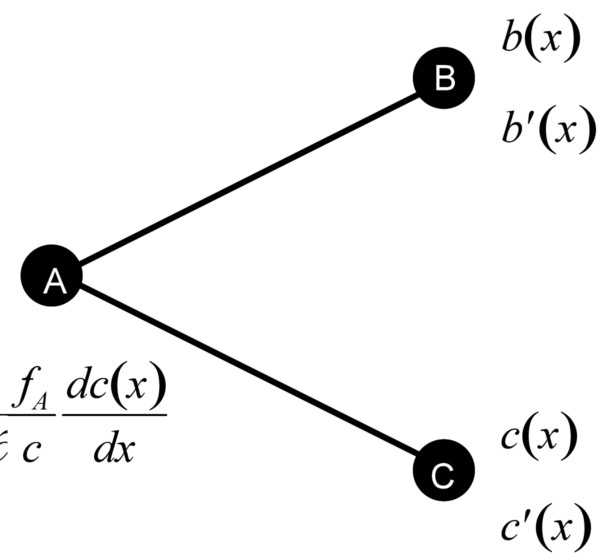
Calibration and Pricing

- **Pricing** is basically function evaluation: $P(x, y, \dots)$ given $x, y, \dots$
- **Calibration** fundamentally is root finding: solve $P(x, y, \dots) = p$ for $x, y, \dots$
  - Implied volatility, interest rate tree calibration, spread, option-adjusted spread, etc.
- Fast foot finding usually requires derivatives: $\frac{\partial P(x,y,\dots)}{\partial x}, \frac{\partial P(x,y,\dots)}{\partial y}, \dots$
- How to find those derivatives efficiently?

The Differential Tree Idea

- Given a backward induction tree for pricing
- Computation at A is driven by inputs from B and C
- Chain rule

$$f_A\big(a(x), b(x), c(x)\big)$$



$$\frac{df_A}{dx} = \frac{\partial f_A}{\partial a}\frac{da(x)}{dx} + \frac{\partial f_A}{\partial b}\frac{db(x)}{dx} + \frac{\partial f_A}{\partial c}\frac{dc(x)}{dx}$$

Calibrating the Black-Derman-Toy Model (BDT)

Number of years	Average number of iterations	Number of years	Average number of iterations
100	3.474747	1100	2.926297
200	3.236181	1200	2.917431
300	3.157192	1300	2.923788
400	3.085213	1400	2.922802
500	3.020040	1500	2.893262
600	2.973289	1600	2.870544
700	2.951359	1700	2.847557
800	2.929912	1800	2.831573
900	2.923248	1900	2.817272
1000	2.919920	2000	2.806903

The zero-coupon bond yield is described by $0.06 + 0.05 \ln t$

Time partition is one period per year

Efficiency in Calibrating BDT (in seconds)

Number of years	Running time	Number of years	Running time	Number of years	Running time
3000	398.880	39000	8562.640	75000	26182.080
6000	1697.680	42000	9579.780	78000	28138.140
9000	2539.040	45000	10785.850	81000	30230.260
12000	2803.890	48000	11905.290	84000	32317.050
15000	3149.330	51000	13199.470	87000	34487.320
18000	3549.100	54000	14411.790	90000	36795.430
21000	3990.050	57000	15932.370	120000	63767.690
24000	4470.320	60000	17360.670	150000	98339.710
27000	5211.830	63000	19037.910	180000	140484.180
30000	5944.330	66000	20751.100	210000	190557.420
33000	6639.480	69000	22435.050	240000	249138.210
36000	7611.630	72000	24292.740	270000	313480.390

75MHz Sun SPARCstation 20, one period per year

Efficiency in Calculating Spread (in seconds)

Number of partitions	Running time	Number of iterations	Number of partitions	Running time	Number of iterations
500	7.850	5	10500	3503.410	5
1500	71.650	5	11500	4169.570	5
2500	198.770	5	12500	4912.680	5
3500	387.460	5	13500	5714.440	5
4500	641.400	5	14500	6589.360	5
5500	951.800	5	15500	7548.760	5
6500	1327.900	5	16500	8502.950	5
7500	1761.110	5	17500	9523.900	5
8500	2269.750	5	18500	10617.370	5
9500	2834.170	5	....	....	....

75MHz Sun SPARCstation 20

Efficiency in Calculating Implied Volatility (in seconds)

American call			American put		
Number of partitions	Running time	Number of iterations	Number of partitions	Running time	Number of iterations
100	0.008210	2	100	0.013845	3
200	0.033310	2	200	0.036335	3
300	0.072940	2	300	0.120455	3
400	0.129180	2	400	0.214100	3
500	0.201850	2	500	0.333950	3
600	0.290480	2	600	0.323260	2
700	0.394090	2	700	0.435720	2
800	0.522040	2	800	0.569605	2

Intel 166MHz Pentium, running Microsoft Windows 95

Part 5:
Monte Carlo Pricing

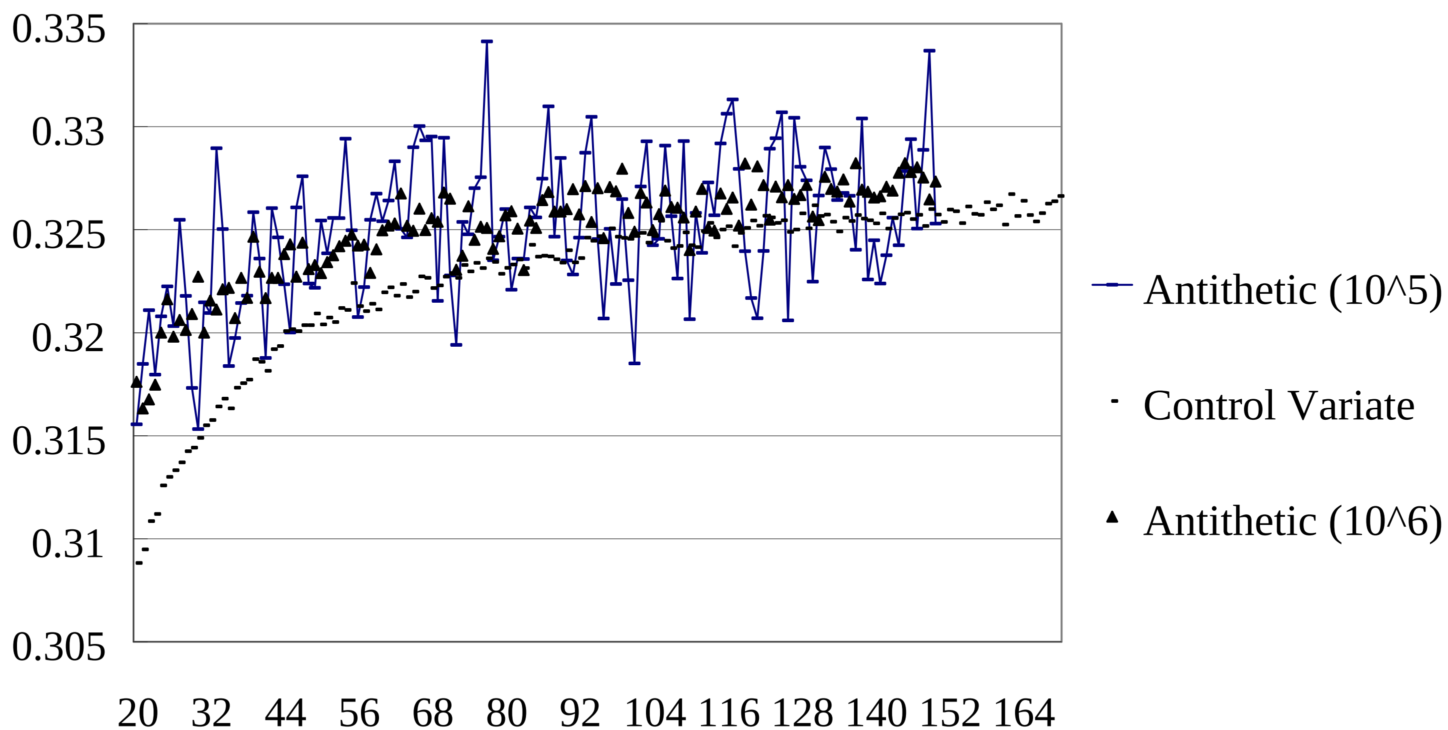
Ideas and Facts

- Simulation of the underlying asset price
- Average the replications
- Bound is only probabilistic (no guarantee)
- Maybe the only viable method for complex securities
  - Mortgage-backed securities and multivariate options
- Promising applicability to *American-style* options
  - Efficiency remains an issue
- Quasi-Monte Carlo: jury still out

Variance Reduction Schemes

- Crude Monte Carlo converges relatively slowly, at a rate of $O(1/\sqrt{N})$
- Variance reduction (efficiency improving) schemes are often necessary to improve convergence
  - Antithetic, control variates, conditioning
- For many path-dependent options, control variates seem to have the lowest variance

Variance Reduction Schemes for Asian Options



Part 6:
Path-Dependent Options Pricing

Issues

- Some path-dependent derivatives are easy to price
  - Barrier-type options, (simple) reset options, geometric Asian options, etc.
- Other path-dependent derivatives seem hard to price
  - *Arithmetic* Asian options, e.g.
- Theory says there are derivatives which are *provably* hard to price
  - No natural options have been identified as such yet
- Analytical approximations, approximation algorithms, Monte Carlo simulation, etc.

Asian Option Defined

- $S_0, S_1, \ldots, S_n$ denote the prices of the underlying asset
  - Arithmetic Asian call's terminal payoff:

$$\max(\frac{1}{n+1}\sum_{i=0}^{n} S_i - X, 0)$$

  - Arithmetic Asian put's terminal payoff:

$$\max(X - \frac{1}{n+1}\sum_{i=0}^{n} S_i, 0)$$

- Want to calculate the expected payoff

Issues

- The binomial model converges to the analytic value
- Due to the non-combining of the tree, the time is in the order of $O(2^n)$
  - It *seems* almost every path has to be explored
- Monte Carlo: no control over the error and limited mostly to European options
- Quasi-Monte Carlo is not well understood
- Analytical approximations fail under some circumstances
- That leaves us exact and approximation algorithms

Approximation and Exact Algorithms

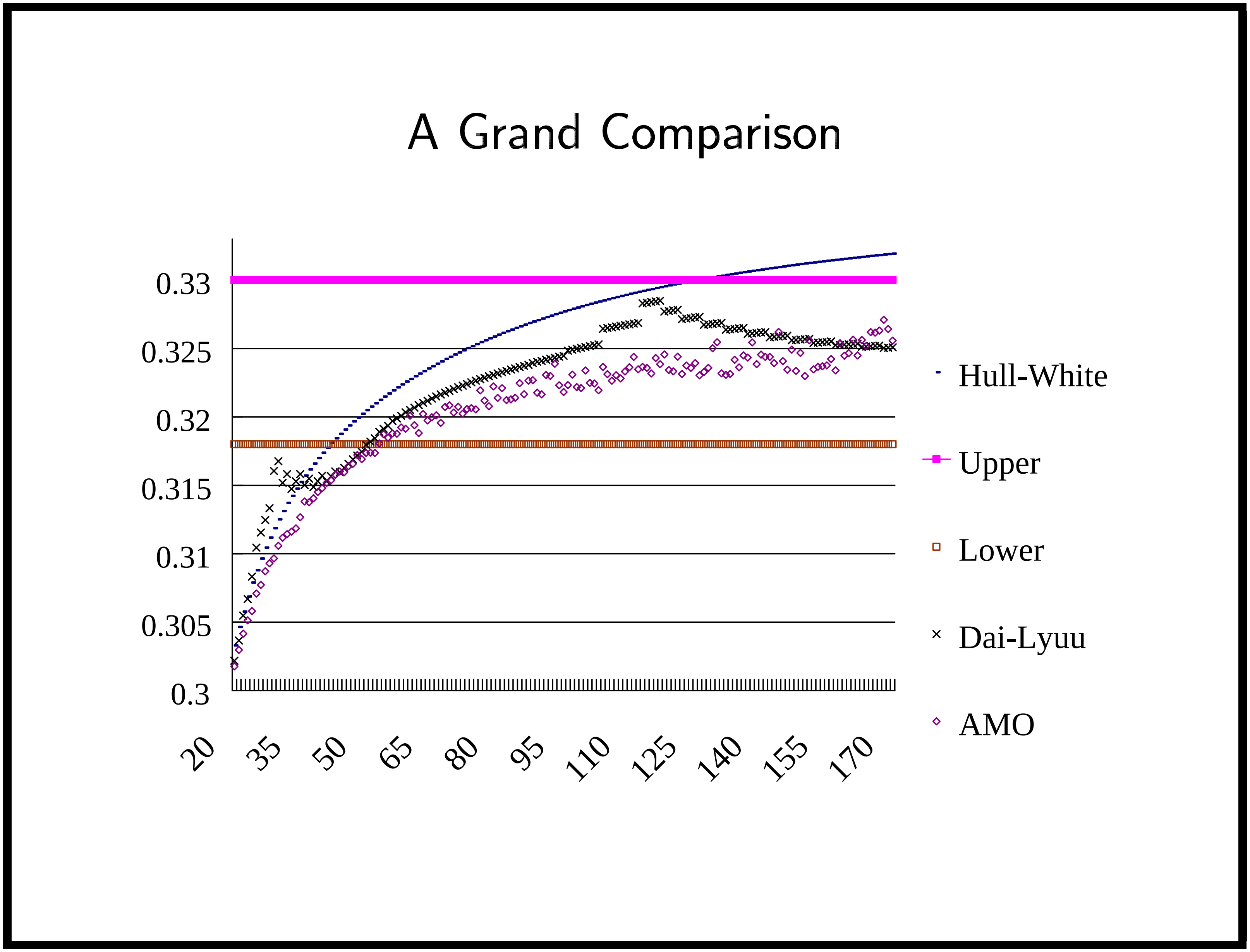
- The popular Hull-White algorithm of 1993
  - Interpolation on the price tree (see [Hull 1999])
  - Overpricing
- A recent $O(kn^2)$-time algorithm (AMO) can deviate from the $O(2^n)$ binomial tree algorithm by at most $O(nX/k)$ [Aingworth, Motwani, and Oldham 2000]
  - Similar to Hull-White, but analyzable
- An unpublished result lowers the error bound to $O(\sqrt{n \ln n}\, X/k)$ [Huang and Lyuu 2000]
- A converging general-purpose quasi-polynomial-time algorithm [Dai and Lyuu 1999]

Basic Ideas of the Dai-Lyuu (DL) Algorithm

- A trinomial tree that guarantees all the asset prices to be finite-precision binary numbers
- Convergence to the continuous-time model
- Backward induction is carried out exactly
  - Contrast this with Hull-White
- The extent of the exponential explosion is dramatically reduced
  - DL can be executed comfortably at $n = 141$
  - Note that $2^{141} \approx 3 \times 10^{42}$

More Details of the Dai-Lyuu Algorithm

- Option value is homogeneous of degree one in the stock price
- Multiply the stock price and the exercise price by 2^m to make sure every asset price on the tree is another integer
- Since a sum of integers is an integer, the state variable at each node, the running subtotal $\sum_{i=0}^{k} S_i$, is an integer
  - This key property relieves backward induction of approximations (such as interpolation in Hull-White)
- There are memory optimization issues


A Grand Comparison
0.33
0.325
0.32
0.315
0.31
0.305
0.3
20
35
50
65
80
95
110
125
140
155
170
Hull-White
Upper
Lower
Dai-Lyuu
AMO

Comparison with Monte Carlo and Hull-White

Period	Monte Carlo		Hull-White	AMO	Dai-Lyuu	
	Lower	Upper			Value	Time
42	0.32069128	0.32463872	0.318055	0.315367	0.315663	1
53	0.32222236	0.32617764	0.321244	0.318063	0.318910	2
64	0.32320948	0.32717652	0.323531	0.319740	0.321187	3
75	0.32252936	0.32648464	0.325318	0.320545	0.322644	7
86	0.32444756	0.32842244	0.326740	0.321654	0.323726	13
97	0.32507268	0.32905932	0.327897	0.323094	0.324915	23
108	0.32408644	0.32804956	0.328836	0.322812	0.326661	39
119	0.32621672	0.33020728	0.329614	0.323427	0.327743	61
130	0.32365844	0.32762156	0.330263	0.325458	0.326839	96
141	0.32463656	0.32861144	0.330767	0.324390	0.326170	145

The initial underlying asset value is 50, the exercise price is 60, the risk free rate is 10% per year, the volatility is 0.3 per year, and the option has a life of 0.5 year

More Comparisons

Maturity (Years)	Method	Exercise Price=40	Exercise Price=45	Exercise Price=50	Exercise Price=55	Exercise Price=60
0.5	HW	10.755	6.363	3.012	1.108	0.317
	MC	10.759	6.359	2.998	1.112	0.324
	DL	10.754	6.356	2.997	1.104	0.317*
	Levy	10.765	6.386	3.024	1.105	0.313
1.0	HW	11.545	7.616	4.522	2.420	1.176
	MC	11.544	7.606	4.515	2.401	1.185
	DL	11.547	7.616	4.517	2.412	1.170*
	Levy	11.576	7.662	4.557	2.431	1.172
1.5	HW	12.285	8.670	5.743	3.585	2.124
	MC	12.289	8.671	5.734	3.577	2.135
	DL	12.284	8.674	5.750	3.585	2.118
	Levy	12.337	8.738	5.801	3.619	2.133
2.0	HW	12.953	9.582	6.792	4.633	3.057
	MC	12.943	9.569	6.786	4.639	3.055
	DL	12.944	9.577	6.786	4.625	3.045
	Levy	13.024	9.671	6.874	4.691	3.087

The initial underlying asset value is 50, the risk free rate is 10% per year, and the volatility is 0.3 per year.

HW denotes the Hull-White algorithm. Monte Carlo simulations (MC) are based on 100,000 trials. DL is the Dai-Lyuu method with the number of periods equal to 30. Levy denotes Levy's approach.

Extreme-Case Comparisons with Many Methods

r, σ, T, S	GE	Shaw	Euler	PW	TW	MC	DL
0.05,0.5,1,1.9	0.195	0.193	0.194	0.194	0.195	0.196	0.193
0.05,0.5,1,2.0	0.248	0.246	0.247	0.247	0.250	0.249	0.246
0.05,0.5,1,2.1	0.308	0.306	0.307	0.307	0.311	0.309	0.306
0.02,0.1,1,2.0	0.058	0.520	0.056	.0624	.0568	.0565	0.0558
0.18,0.3,1,2.0	0.227	0.217	0.219	0.219	0.220	0.220	0.219
0.125,0.25,2,2.0	0.172	0.172	0.172	0.172	0.173	0.172	0.172
0.05,0.5,2,2.0	0.351	0.350	0.352	0.352	0.359	0.348	0.351

The exercise price is 2.0, S is the initial price of the underlying asset, GE is the Geman-Eydeland method, PW is the Post-Widder method, and TW is the Turnbull-Wakeman method.

Part 7:
Looking into the Future

Do We Really Have To Compute It?

- Think of the option values as the range of a function
- If the surface of the function is reasonably *smooth*, we may invest few nights' work in approximating the surface
- Afterwards, we only need to interpolate from the surface
- Issues
  - Will this work for complex options?
  - How many data points are needed?

Conclusions

- Derivatives pricing draws ideas from many fields
- Efficient algorithms allow more strategies to be explored
- Much work remains to be done
- Guarded optimism: inherent complexity is probably not a problem