Primary objectives:

- Shortest paths and Bellman-Ford algorithm
- Capital budgeting
- Knapsack
- Pricing of American Options

PART 8
DYNAMIC PROGRAMMING

Shortest paths in directed graphs

Directed graph

- Tuple $G = (V, A)$, where V is finite set of nodes and $A \subseteq V \times V$ is set of arcs (also called edges); arc from u to v is denoted by uv
- $c : A \longrightarrow \mathbb{R}$ weights on arcs
- Path is sequence $P = v_0, v_1, \ldots, v_k$ of nodes such that $v_{i-1}v_i \in A$ for $i = 1, \ldots, k$.
- Length of path $c(P) = \sum_{i=1}^{k} c(v_{i-1}v_i)$
- Path $v_0, v_1, \ldots, v_k$ with $v_0 = v_k$ is called cycle

Shortest path problem

Let $s \in V$. Determine for each $v \in V$ a shortest path from s to v.

Bellman's Equations

- Suppose G does not contain cycle of negative length
- Let $\ell(v)$ be length of shortest path from s to v (possibly ∞)
- Clearly $\ell(s) = 0$ (no neg. cycles!)
- For $uv \in A$ one has $\ell(v) \leqslant \ell(u) + c(uv)$
- If $v \neq s$, there must be a node u on shortest path from s to v immediately preceding v.
  For this u one has
  $$\ell(v) = \ell(u) + c(uv)$$
  (principle of optimality)
- The shortest path distances $\ell(v)$ satisfy Bellman's equations

$$x_s = 0,$$
$$x_v = \min\{x_u + c(uv) : uv \in A\}, \quad v \in V.$$

Sufficiency

Theorem:

If G does not contain cycles of length $\leqslant 0$ and if one has a path from s to each other node v, then there is unique solution to Bellman's equations, where $x_u = \ell(u)$ for each $u \in V$.

Proof

- Let x be solution to Bellman's equations
- Let $v \in V$, $v \neq s$. There exists $uv \in A$ with $x_v = x_u + c(uv)$. Likewise, there exists $w \in V$ with $x_u = x_w + c(wu)$.
- If, repeating, this process, we come back to v again, we have found a cycle of length 0 which is excluded
- Consequently, we arrive at s and we have constructed a path $P = (s = v_0, v_1, \ldots, v_k = v)$ with $x_v = c(P) + x_s = c(P)$. This shows $x_v \geqslant \ell(v)$.
- Suppose $x \neq \ell$. Then there exists node v with $x_v > \ell(v)$ such that the node u preceding v on shortest path from s to v has $x_u = \ell(u)$.
- Bellman's equations imply $x_v \leqslant x_u + c(uv) = \ell(u) + c(uv) = \ell(v)$

Acyclic Graphs

- $G = (V, A)$ is acyclic if G does not contain cycles
- In this case, G has an ordering "$<$" of nodes, such that $uv \in A$ implies $u < v$ (depth first search)
- Assume nodes are set $\{1, \ldots, n\}$.

Bellman's equations acyclic graphs

$$x_1 = 0$$
$$x_j = \min_{k<j}(x_k + c_{kj}) \quad j = 2, 3, \ldots, n.$$

- Easily solved by substitution
- x_2 is determined from x_1, x_3 determined from x_1 and x_2 etc.
- Running time: $O(|A| + |V|)$

Bellman-Ford algorithm

Bellman-Ford algorithm

- x_v^j length of shortest path from s to v using at most j arcs
- Initialization: $x_s^0 = 0$, $x_v^0 = \infty$ for $v \neq s$
- **for** $i = 1, 2, \ldots, n$
  for $uv \in A$
  $x_v^i = \min\{x_v^{i-1}, x_u^{i-1} + c(uv)\}$

Running time

- Iterations outer loop: $|V|$
- Each arc is considered exactly once in inner loop: $O(|A|)$
- Complexity: $O(|V| \cdot |A|)$.

Negative cycles

- If network does not contain negative cycles and v is reachable from s, then a shortest path uses at most $n-1$ arcs.
- Bellman-Ford is correct if no negative cycles are present.

Exercise

Design a polynomial-time algorithm which detects whether $G=(V,A)$ together with arc-weights c has a negative cycle.

Dynamic programming

A capital budgeting problem

4 million to be invested in projects in three different regions
At most one project can be run in each region

Project	Region 1 cost/profit	Region 2 cost/profit	Region 3 cost/profit
1	0/0	0/0	0/0
2	1/2	1/3	1/2
3	2/4	3/9	2/5
4	4/10	-	-

Enumeration

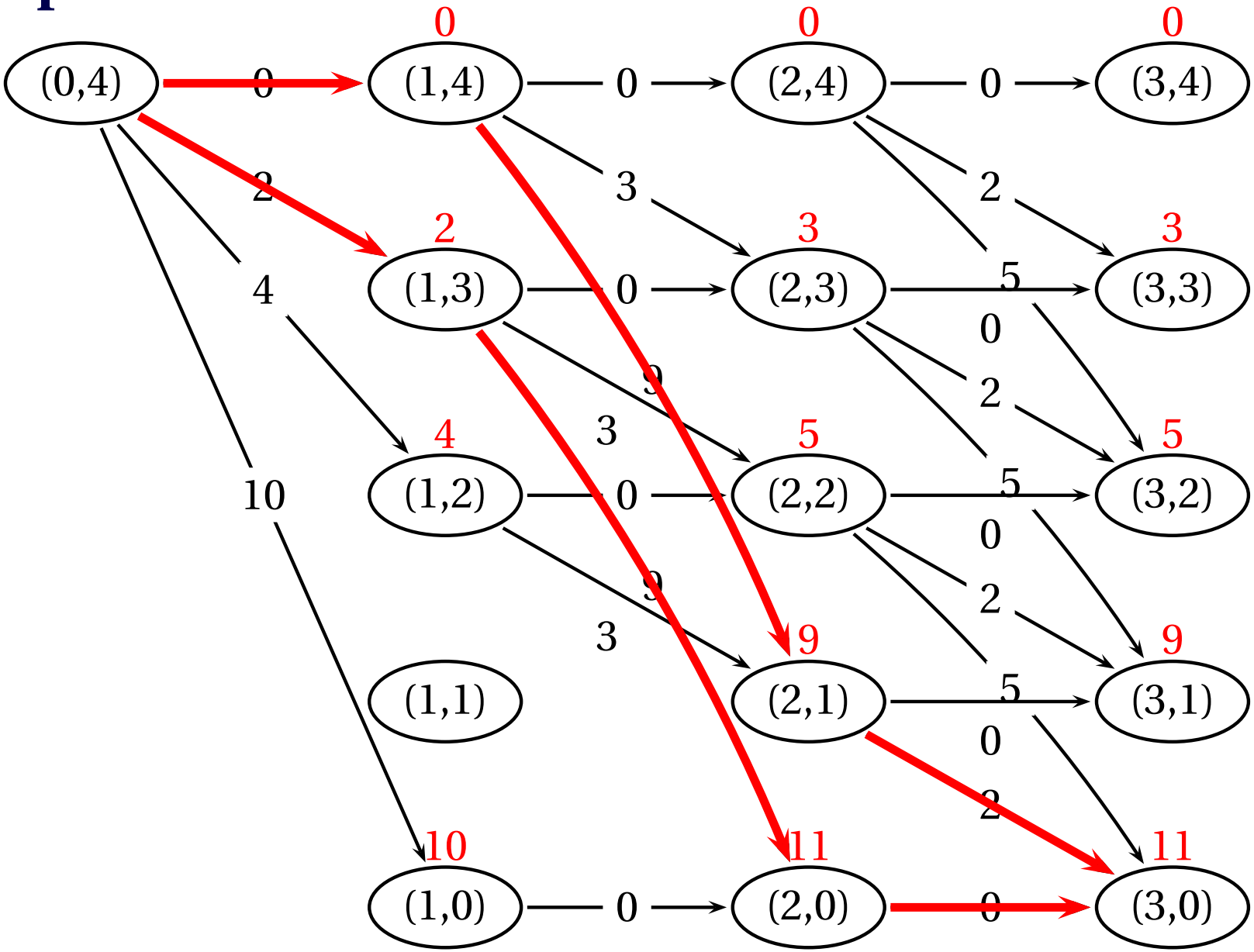
$4 \times 3 \times 3$ possibilities

Dynamic program

States

- (i,j) represents optimal choice for first $i \in \{0,1,2,3\}$ regions having still $j \in \{0,1,2,3,4\}$ mio. left to invest.
- Construct graph with nodes being states and arcs from stages (i,j) to $(i+1,j')$ having cost p if the investment of $j-j'$ reflects an implementation of a project of cost $j-j'$ and profit p in region $i+1$
- Longest path in this directed acyclic graph is optimal strategy
- Number of states: 4×5
- If we had $x \in \mathbb{N}$ regions with at least two possible projects per region and an amount of $y \in \mathbb{N}$ to invest, where each project would have integral costs, one would have $(x+1) \cdot (y+1)$ states, whereas enumeration would require at least 2^x decisions to be compared.

Example cont.



The Knapsack Problem

Knapsack

- n items of weight $w_i \in \mathbb{N}$ and profit $p_i \in \mathbb{N}$, $i = 1, \ldots, n$
- Knapsack of capacity $K \in \mathbb{N}$
- Task: Compute subset of items which still fit into knapsack and maximizes the profit

Dynamic program for Knapsack

- $W_i(p)$ least possible weight that has to be accumulated in order to have total profit p using only items from $\{1,\ldots,i\}$
- $W_{i+1}(p) = \min\{W_i(p), W_i(p-p_{i+1}) + w_{i+1}\}$
- States: $\{(i,p)\colon\ i \in \{1,\ldots,n\},\ p \in \{0,\ldots,n\cdot p_{\max}\}\}$, where $p_{\max} = \max\{p_i\colon\ i = 1,\ldots,n\}$.
- There is an edge from (i,p) to $(i+1,p)$ of weight 0 and there is an edge from (i,p) to $(i+1,p+p_{i+1})$ of weight w_{i+1}.
- Compute shortest path distances from $(0,0)$ to all nodes in directed graph
- Number of nodes: $p_{\max}\cdot n^2$
- Number of arcs: $2\cdot p_{\max}\cdot n^2$

Theorem

The knapsack problem can be solved in time $O(n^2\cdot p_{\max})$.

Alternative dynamic program

- $P_i(w)$: Maximum profit which can be accumulated choosing items out of $\{1,\ldots,i\}$ having total weight w.
- $P_{i+1}(w) = \max\{P_i(w), P_i(w-w_{i+1}) + p_{i+1}\}$
- States: $\{(i,w) : i \in \{1,\ldots,n\}, w \in \{0,\ldots,K\}\}$
- There is an edge from (i,w) to $(i+1,w)$ of weight 0 and an edge from (i,w) to $(i+1, w+w_{i+1})$ of weight p_{i+1}.
- Number of states: $n \cdot K$
- Number of arcs: $2 \cdot n \cdot K$
- Find longest paths from $(0,0)$ to other states
- Running time: $O(n \cdot K)$

PART 8.1
OPTION PRICING

American call options

- Gives the holder right to purchase underlying security for prescribed amount strike price
- Valid until certain expiration date

Pricing derivative security

- S_0 current price of underlying security
- Two possible outcomes up and down at time 1:

$$S_1^u = S_0 \cdot u$$
$$S_1^d = S_0 \cdot d$$

- How to price the derivative security?

Replication

- Consider portfolio of $\triangle$ shares of the underlying and B cash
- Up-state: $\triangle \cdot S_0 \cdot u + B \cdot R$, ($R$ is risk-less interest rate)
- Down-state: $\triangle \cdot S_0 \cdot d + B \cdot R$
- For what values of $\triangle$ and B will portfolio have same payoff C_1^u and C_1^d of derivate?

$$\begin{aligned} \triangle \cdot S_0 \cdot u + B \cdot R &= C_1^u \\ \triangle \cdot S_0 \cdot d + B \cdot R &= C_1^d \end{aligned}$$

- One obtains

$$\triangle = \frac{C_1^u - C_1^d}{S_0(u-d)}$$

$$B = \frac{uC_1^d - dC_1^u}{R(u-d)}$$

- Since portfolio is worth $S_0\triangle + B$ today, this should also be price for derivate security

$$\begin{aligned} C_0 &= \frac{C_1^u - C_1^d}{u-d} + \frac{uC_1^d - dC_1^u}{R(u-d)} \\ &= \frac{1}{R}\left[\frac{R-d}{u-d}C_1^u + \frac{u-R}{u-d}C_1^d\right] \end{aligned}$$

Risk-neutral probabilities

$p_u = \frac{R-d}{u-d}$, $p_d = \frac{u-R}{u-d}$

Remark

There is arbitrage opportunity if $u > R > d$ is not satisfied. Consequently $p_u, p_d > 0$ and since $p_u + p_d = 1$ one can interpret p_u and p_d as probabilities.

n time steps

- Basic period length (day or week . . .)
- Price of asset in period being S there is the "up" event (price uS) with probability p and "down" event (price dS) with probability $1-p$
- Starting from price S_0 in period 0, in period k it is $u^j d^{k-j} S_0$ if there are j up- and $k-j$ down moves.
- Probability is $\binom{k}{j} p^j (1-p)^{k-j}$ (binomial distribution).

n time steps

Determining u, d and p

- S_k: Price in periods $k = 0, \ldots, n$
- Assume to know : Mean value μ and volatility σ of $\ln(S_n/S_0)$
- Let $\Delta = 1/n$ be length between two consecutive periods
- Mean and volatility of $\ln(S_1/S_0)$ are $\mu\Delta$ and $\sigma\sqrt{\Delta}$ respectively
- Direct computation yields $\mu\Delta = p \cdot \ln u + (1-p)\ln d$ and $\sigma^2\Delta = p \cdot (1-p)(\ln u - \ln d)^2$.
- Set $d = 1/u$ then equations simplify

$$\begin{aligned} \mu \cdot \Delta &= (2p-1)\ln u \\ \sigma^2 \cdot \Delta &= 4p(1-p)(\ln u)^2 \end{aligned}$$

- Squaring first and adding it to second equation yields

$$(\ln u)^2 = \sigma^2\Delta + (\mu\Delta)^2.$$

Determining u, d and p cont.

- $$\begin{aligned} u &= e^{\sqrt{\sigma^2\triangle+(\mu\triangle)^2}} \\ d &= e^{-\sqrt{\sigma^2\triangle+(\mu\triangle)^2}} \\ p &= \frac{1}{2}\left(1+\frac{1}{\sqrt{1+(\sigma^2/\mu^2)\triangle}}\right) \end{aligned}$$

- For small $\triangle$ this is approximately

$$\begin{aligned} u &= e^{\sigma\sqrt{\triangle}} \\ d &= e^{-\sigma\sqrt{\triangle}} \end{aligned}$$

Example

- 52 periods
- S_0 known and random price S_{52} with mean and standard deviation of $\ln(S_{52}/S_0)$ being 10% and 30% respectively. Since $\triangle = 1/52$ "is small" one has $u = e^{0.3/\sqrt{52}} = 1.0425$ and $d = 0.9592$.

Dynamic program to determine value of option

- Suppose strike price is c
- Work backwards from time N to time 0
- Nodes at time N are terminal nodes
- Option value $v(N,j)$ of terminal nodes at height j is

$$\max\{u^j d^{N-j} S_0 - c, 0\}$$

- Compute $v(k,j)$ from $v(k+1,j)$ and $v(k+1,j+1)$ using formula with risk-neutral probabilities $p_u = \frac{R-d}{u-d}$, $p_d = \frac{u-R}{u-d}$:

$$v(k,j) = \max\left\{\frac{1}{R}(p_u v(k+1,j+1) + p_d v(k+1,j)), u^j d^{k-j} S_0 - c\right\} \tag{21}$$

  This reflects the following rationale: Either wait until tomorrow (time $k+1$) or exercise your option today.
- Output $v(0,0)$

Example

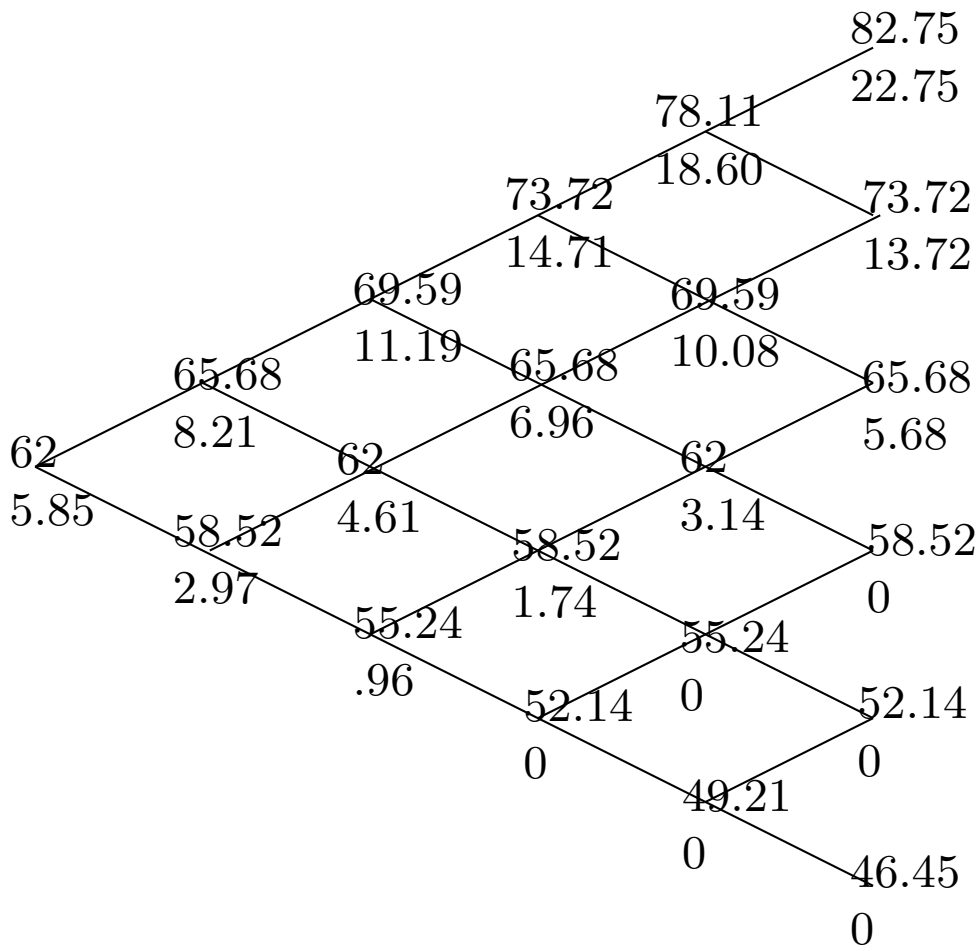
Stock with

- Yearly volatility of logarithm $\sigma = .2$
- Current price is 62
- What is price of American call option with expiration date in 5 months from now?
- Strike price c is 60
- Rate of interest is 10% compounded monthly

Compute values

- $\triangle = 1/12$, $u = 1.05943$, $d = .94390$, $R = 1 + 0.1/12 = 1.00833$
- $p_u = .55770$
- Fill table entries at the end of period. Example (upper right): $S_0 \cdot u^5 - c = 22.75$

Example cont.



Each node is an outcome of a sequence of "ups" and "downs". The label on top of each node is the value of the underlying, the value on the bottom is the cost of the option. Entries are filled from left-to-right using formula (21).

Primary objectives:

- Shortest paths and Bellman-Ford algorithm ✔
- Capital budgeting ✔
- Knapsack ✔
- Pricing of American Options ✔