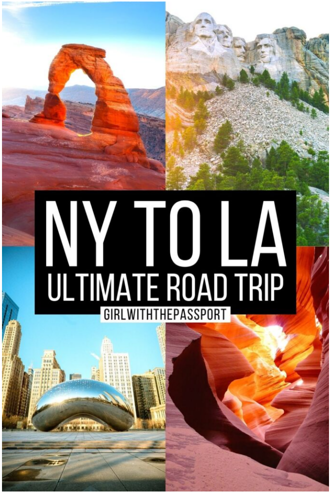


Class 26
April 21, 2023

Dynamic Programming II

Handouts

Characteristics of Dynamic Programming

Announcements

TODAY

MATHEMATICS SEMINAR

Dynamic and adaptive information accumulation and exchange during foraging

To effectively forage in natural environments, organisms must learn and adapt to changes in the availability of resources. Patch exploitation is a canonical foraging behavior, and the way in which animals account for

environmental change and uncertainty should be captured more accurately by mathematical models. We first address this issue in a model describing agents that statistically and sequentially infer patch resource quality using Bayesian updating, based on their resource encounter history. Uncertainty leads patch-exploiting foragers to overharvest (underharvest) patches with initially low (high) resource yields. Social interactions that synchronize the times that foragers depart patches improve group foraging efficiency. We also address the problem of groups responding to rapid patch-quality changes. Using the example of honeybee swarms, we find social interactions that allow bees can directly switch the opinion of nest-mates foraging at lower-yielding feeders can quickly increase the fraction of the swarm at the correct feeder. Our mathematical framework allows us to compare the effects of a suite of mechanisms by which bees social inhibit the expressed opinions of their neighbors.

Zachary P Kilpatrick, PhD
Associate Professor
University of Colorado Boulder
Department of Applied Mathematics
https://www.colorado.edu/amath/zpkilpat

Friday, April 21 from 12:15 - 1:15pm in WNS 100

Announcements

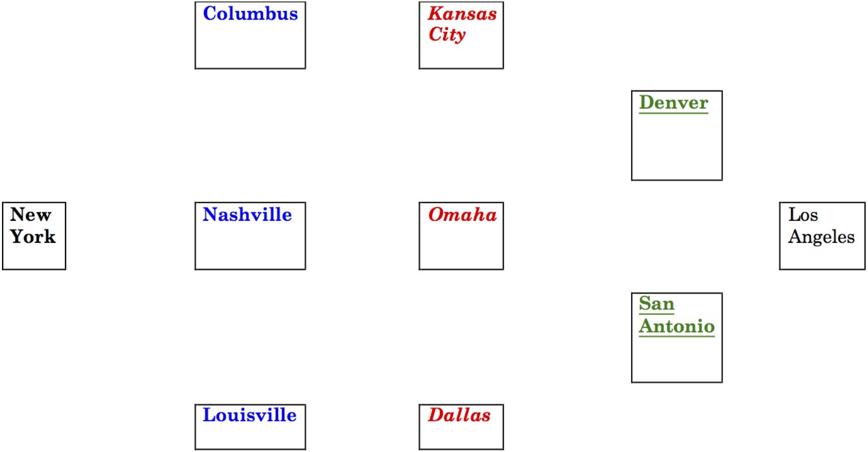
Exam 2: Monday Evening

7 PM – ?

Dynamic Programming

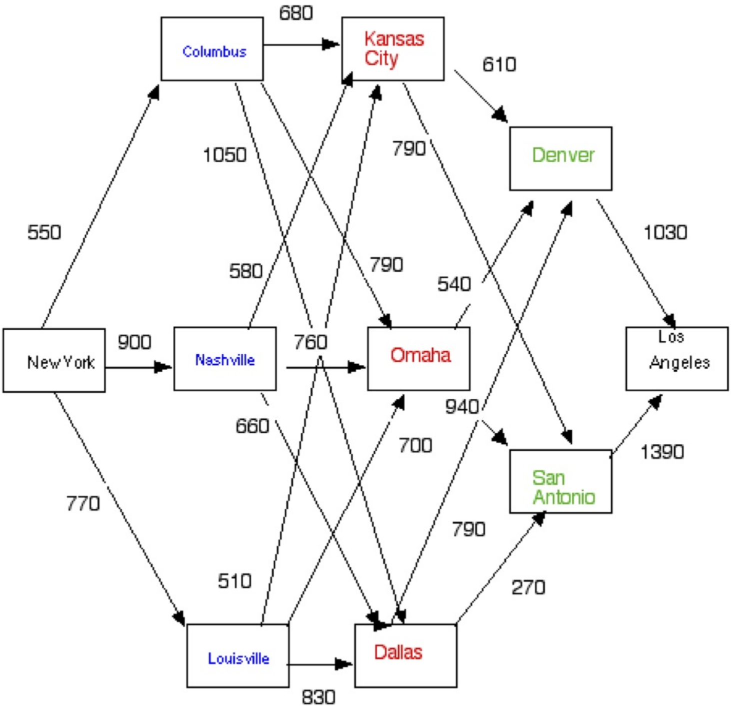
Dynamic programming is a useful technique that we can use to solve many optimization problems by breaking up large problems into a sequence of smaller, more tractable problems and then **Working Backward** from the end of the problem toward the beginning of the problem.

New York to Los Angeles Road Trip

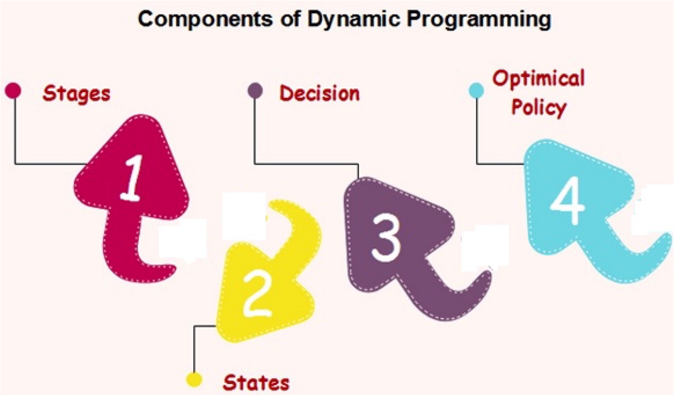


New York		Columbus	Nashville	Louisville
Columbus: 550		Kansas City: 680	Kansas City 580	Kansas City: 510
Nashville: 900		Omaha: 790	Omaha: 760	Omaha: 700
Louisville: 770		Dallas: 1050	Dallas: 660	Dallas: 830

Kansas City	Omaha	Dallas		Denver	San Antonio
Denver: 610	Denver: 540	Denver: 790		Los Angeles: 1030	Los Angeles: 1390
San Antonio: 790	San Antonio: 940	San Antonio: 270			



Columbus
680
Kansas City
610
Denver
1050
790
550
580
790
540
1030
New York
900
Nashville
760
Omaha
Los Angeles
940
660
700
1390
San Antonio
770
790
510
270
Louisville
Dallas
830


Components of Dynamic Programming
Stages
1
Decision
3
Optimical Policy
4
2
States

Characteristics of Dynamic Programming Problems

1. Problem can be divided into stages with a policy decision required at each stage

(NY to LA trip: which city should I stop at tonight?)

We'll number the stages $n = 1, 2, 3, ..$

2. Each stage has a number of states associated with the beginning of the stage.

STATES = possible conditions in which the system could be at that stage of the problem (use s to indicate states)
Stage 1: States = New York
Stage 2: States = Columbus, Nashville, Louisville
Stage 3: States = Kansas City, Omaha, Dallas
Stage 4: States = Denver, San Antonio

Number of states can be finite or infinite

3. The effect of the policy decision at each stage is to transform the current state to a state associated with the beginning of the next stage.

Image: column of nodes at a stage.
Arcs going from these nodes to nodes at the next stage.
Value on an arc = immediate contribution to the objective function from making that policy decision.

4. The solution procedure finds an optimal policy for the overall problem

What policy decision you should make at each stage for every possible state? x_n^* for each s

5. The Principle of Optimality: An optimal policy for the remaining stages is independent of the policy decisions adopted at previous stages.

Markovian property: it matters not how you got here, only where you are, to determine the next step.
Consequence: Suppose shortest route from NY to LA passes through Kansas City. Then the portion of that route from KC to LA is the shortest path from KC to LA.

6. Solution procedure begins by finding optimal policy for the last stage.

7. A recursive relationship that identifies the optimal policy for stage n given the optimal policy for $n+1$ is known.

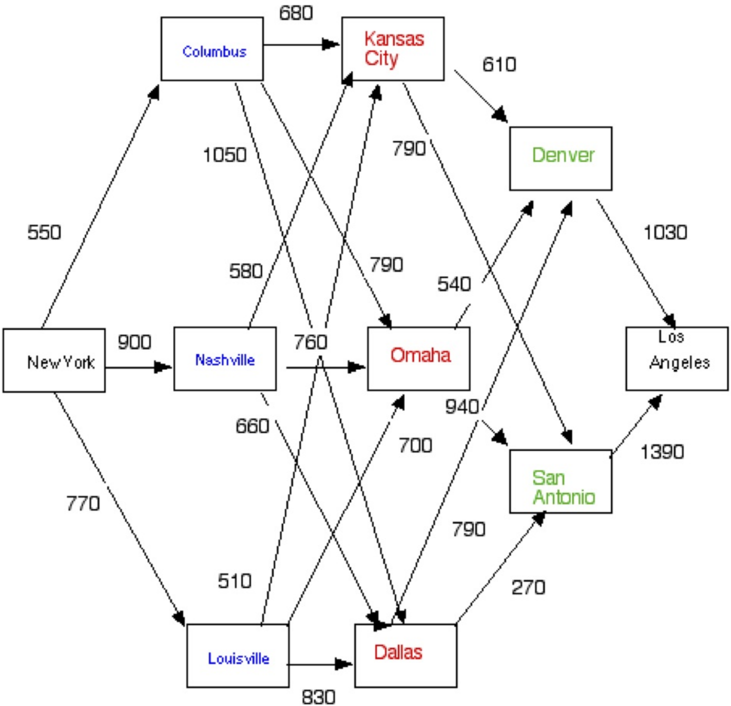
$$f_n^*(s) = min_x[C_{sx_n} + f_{n+1}^*(x_n)$$

N = number of stages
n = label for current stage $(n = 1, 2, ..., N)$
s_n = current state for stage n
x_n = decision variable for stage n
x_n^* = optimal value for x_n given s_n
$f_n(s_n, x_n)$ = contribution of stages $n, n+1, ..., N$ to objective function if the system starts in state s_n at stage n, we make decision x_n and we make optimal decisions at all future stages.

8. Use the recursive relationship to work backward stage by stage.

Construct a table at each stage:

x_n / s_n	$f_n(s_n, x_n)$	$f_n^*(s_n)$	x_n^*



Columbus
680
Kansas City
610
Denver
1050
790
550
580
790
540
1030
New York
900
Nashville
760
Omaha
Los Angeles
660
940
700
1390
San Antonio
770
790
510
270
Louisville
Dallas
830

Illustration

Stage 4

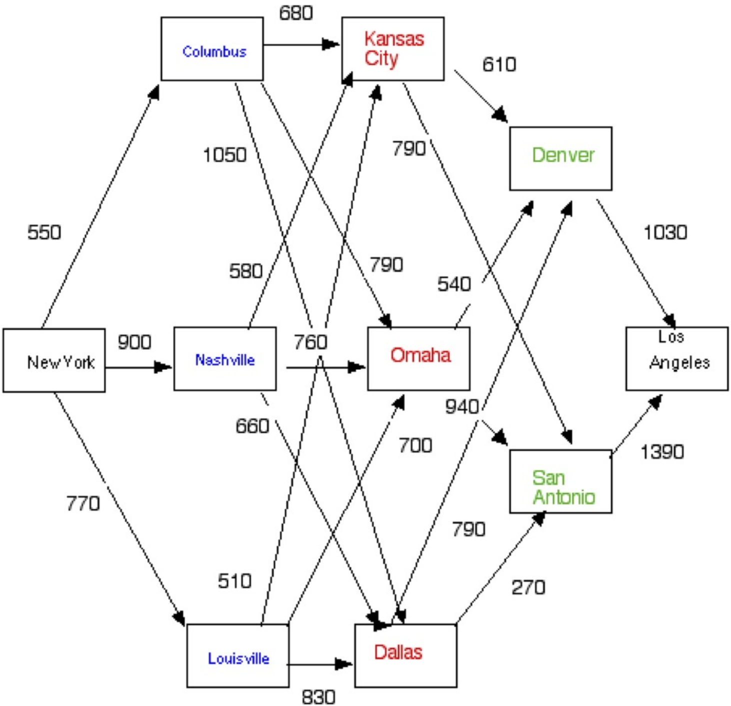
$n = 4$	s	$f_4^*(s)$	x_4^*
	Denver	1030	Los Angeles
	San Antonio	1390	Los Angeles

Stage 3

$n = 3$

$$f_3(s, x_3) = C_{sx_3} + f_4^*(x_3)$$

x_2 / s	Denver	San Antonio	$f_3^*(s)$	x_3^*
Kansas City	610+1030	790+1390		
Omaha	549+1030	940+1390		
Dallas	790+1030	270+1390		



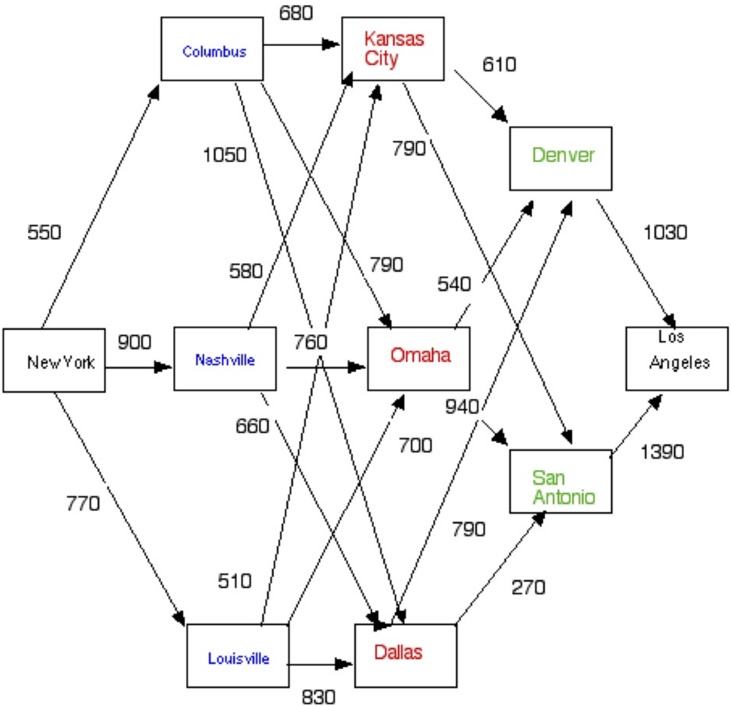
Columbus
Kansas City
Denver
New York
Nashville
Omaha
Los Angeles
San Antonio
Louisville
Dallas
680
610
1050
790
550
580
790
540
1030
900
760
660
940
700
1390
770
790
510
270
830

$$f_3(s, x_3) = C_{sx_3} + f_4^*(x_3)$$

s \ x_2	Denver	San Antonio	$f_3^*(s)$	x_3^*
Kansas City	1640	2180	1640	Denver
Omaha	1570	2330	1570	Denver
Dallas	1820	1660	1660	San Antonio

Stage 2: $n = 2$

	Kansas City	Omaha	Dallas	$f_2^*(x_2)$	x_2^*
Columbus	$680 + 1640$	$790 + 1570$	$1050 + 1660$		
Nashville	$580 + 1640$	$760 + 1570$	$660 + 1660$		
Louisville	$510 + 1640$	$700 + 1570$	$830 + 1660$		


680
Columbus
Kansas City
610
Denver
1050
790
550
580
790
540
1030
900
760
New York
Nashville
Omaha
Los Angeles
940
660
700
1390
San Antonio
770
790
510
270
Louisville
Dallas
830

$$f_2(s, x_2) = C_{sx_2} + f_3^*(x_2)$$

	Kansas City	Omaha	Dallas	$f_2^*(x_2)$	x_2^*
Columbus	2320	2360	2710	2320	Kansas City
Nashville	2220	2320	2320	2220	Kansas City
Louisville	2150	2270	2490	2150	Kansas City

Stage 1

$$f_1(s, x_1) = C_{sx_1} + f_1^*(x_1)$$

s \ x_1	Columbus	Nashville	Louisville	$f_1^*(s)$	x_1^*
New York	550 + 2320 = 2870	900 + 2220 = 3120	770 +2150 = 2920	2870	Columbus

Optimal Route

New York $\rightarrow$ Columbus $\rightarrow$ Kansas City $\rightarrow$ Denver $\rightarrow$ Los Angeles

Characteristics of Dynamic Programming Problems

Problem can be divided into stages with a policy decision required at each stage.

- Each stage has a number of states associated with the beginning of the stage.
- The effect of the policy decision at each stage is to transform the current state to a state associated with the beginning of the next stage.
- The solution procedure finds an optimal policy for the overall problem.
- *The Principle of Optimality:* An optimal policy for the remaining stages is independent of the policy decisions adopted at previous stages.
- Solution procedure begins by finding optimal policy for the last stage.
- A recursive relationship that identifies the optimal policy for stage n given the optimal policy for n+1 is known.

$$f_n^*(s) = min[C_{sx_n} + f_{n+1}^*(x_n)]$$

- A recursive relationship that identifies the optimal policy for stage n given the optimal policy for n+1 is known.

$$f_n^*(s) = min[C_{sx_n} + f_{n+1}^*(x_n)$$

N = number of stages
n = label for current stage ($n = 1, 2, ..., N$)
s_n = current state for stage n
x_n = decision variable for stage n
x_n^* = optimal value for x_n given s_n
$f_n(s_n, x_n)$ = contribution of stages $n, n+1, ..., N$ to objective function if the system starts in state s_n at stage n, we make decision x_n and we make optimal decisions at all future stages.

- Use the recursive relationship to work backward stage by stage.
Construct a table at each stage:

s_n \ x_n	$f_n(s_n, x_n)$	$f_n^*(s_n)$	x_n^*

The Video Distribution Problem

redbox
DVD
RENTALS
redbox

Video Rentals

You own 3 Redbox kiosks located in three different stores and have 5 copies of the new *Good Boys* film. How should you distribute the videos among the stores?

Video Available	Store 1 Rentals	Store 2 Rentals	Store 3 Rentals
0	0	0	0
1	5	6	4
2	9	11	9
3	14	15	13
4	17	19	18
5	21	22	20

Stages = Stores

States = Number of Videos We Have

$$n = 3$$

s	$f_3^*(s)$	x_3^*
0	0	0
1	4	1
2	9	2
3	13	3
4	18	4
5	20	5

$$n = 2 : f_2(s, x_2) = C_{sx_2} + f_3^*(s - x_2)$$

$s \backslash x_2$	0	1	2	3	4	5	$f_3^*(s)$	x_2^*
0	0 + 0	——	——	——	——	——	0	0
1	0 + 4	6 + 0	——	——	——	——	6	1
2	0 + 9	6 + 4	11 + 0	——	——	——	11	2
3	0 + 13	6 + 9	11 + 4	15 + 0	——	——	15	1, 2, 3
4	0 + 18	6 + 13	11 + 9	15 + 4	19 + 0	——	20	2
5	0 + 20	6 + 18	11 +13	15 + 9	19 + 4	22 + 0	24	1,2,3

$$f_2^*(s) = \max_{x_2}[C_{sx_2} + f_3^*(s - x_2)]$$

$$n = 1 : f_1(s, x_1) = C_{sx_1} + f_2^*(s - x_1)$$

x_1 / s	0	1	2	3	4	5	$f_1^*(s)$	x_1^*
5	0 + 24	5 + 20	9 + 15	14 + 11	17 + 6	21 + 0	25	1 or 3

	Solution 1	*Solution 2*
Store 1	1	3
Store 2	2	2
Store 3	2	0
Rentals	**5+11+9=25**	**14+11+0=25**

Why Redbox Has No Clerks