Dynamic Programming
Outline of Introduction

1. Basic concept: Implicit Enumeration
   - Motivational Example
2. Key Assumptions
   - Independence (Separability) and Monotonicity
3. Mathematics
   - Recurrence Formulas
4. Example
5. Types of Problems DP can solve
6. Summary

Basic Solution Strategy

- Enumeration is basic concept
  - This means evaluating “all” the possibilities
  - Checking “all” possibilities, we must find best
- Means that DP can optimize over
  - Non-Convex Feasible Regions
  - Discontinuous, Integer Functions
  - Classes of Problems that LP cannot handle
- HOWEVER
  - CURSE OF DIMENSIONALITY
Curse of Dimensionality

- Number of Possible Designs very large
- Example
  - consider simple issue: the development of 2 sites, for 4 sizes of operations over 3 periods
  - Number of Combinations in 1 period = \(4^2 = 16\)
  - Possibilities over 3 periods = \(16^3 = 4096\)
- In general, size of design is exponential
  = \([\text{(Size)}^{\text{locations}}]\text{ periods}\)
  - Complete enumeration is impractical
- DP uses “implicit” enumeration

Implicit Enumeration

- IE considers all possibilities in principle
- Exploits features of problem to
  - Identify classes of dominated possibilities
  - Reject these classes
  - Vastly reduce dimensionality of enumeration
- Size of numeration for DP
  - Order of \([\text{(Size)}^{\text{Locations}}]\text{ Periods}\)
  - Multiplicative size, not exponential
  - This analysis computationally practical
- Assumptions are key
Tree Metaphor

- Number of actual possibilities grows like branches on a tree
  - At start of analysis (trunk of tree)
  - There are many first choices (branches)
  - Then second choices (branches on branches)
  - Etc

- Dynamic Programming Process
  - Identifies less productive branches
  - Cuts them off (prunes the branches)

- Image common in DP, Decision Analysis

Implicit Enumeration
Simple Example (1)

- Consider shipment from Seattle to DC
- Many routes, with costs for links as in diagram
- To Omaha, 3 routes shown
- From Omaha, 3 possible routes => 9 routes via Omaha
Implicit Enumeration
Simple Example (2)

- We could cost out all 9 routes
- Alternatively, we find best cost to Omaha (350 via Boise)
- Salt Lake (400), Phoenix (450) routes dominated, “pruned”
- So we drop combinations including those segments

Seattle 100 Boise 500 Fargo Detroit
200 250
Salt Lake 200 Omaha Memphis DC
300 150
Phoenix 200 Houston Atlanta

Implicit Enumeration
Simple Example (3)

- We thus consider 3 routes to Omaha, plus 3 routes after
- Total (via Omaha) is 6 = 3 x 2, not 9 = 3^2
- Savings not dramatic in this simple example, of course
- Example illustrates idea

Seattle 100 Boise 500 Fargo Detroit
200 250
Salt Lake 200 Omaha Memphis DC
300 150
Phoenix 200 Houston Atlanta
Solution depends on Decomposition

- Must be able to “decompose” objective function into functions of individual $X_i$:
  \[ G(X) = [g_1X_1, \ldots, g_NX_N] \]
  - Example: cost of Seattle to DC trip can be decomposed into cost of 4 segments of which Seattle to Boise, Salt Lake or Phoenix is first

- Necessary conditions for decomposition
  - Separability
  - Monotonicity

Necessary Conditions

- Separability
  - Objective Function is separable if $g_iX_i$ are independent of $g_JX_J$ for all $J$ not equal to $I$

- Monotonicity
  - Obj. Fnc. is monotonic if improvements in each $g_iX_i$ lead to improvements in Obj. Fnc.
  - Formally, given $G(X) = [g_iX_i, G'(X)]$, monotonic if, for all $g_iX'_i > g_iX''_i$,
    \[ [g_iX'_i, G'(X)] \geq [g_iX''_i, G'(X)] \]
  - Additive functions always monotonic
  - Multiplicative fncts only if $g_iX_i$ non-negative, real
Useful Concepts and Definitions

- Return function
  - Each $g_iX_i$ is a “return function”
  - It defines contribution (“the return”) to Obj. Fcn.
- Each $g_iX_i$ associated with a “stage” in the problem
  - Example: 1st. Stage is from Seattle to Boise, etc
  - Thus $g_1X_1$ are costs from Seattle to Boise, etc
- Each $g_iX_i$ takes on different “states”, that is, possible situations at a stage
  - Example: There are 3 states for 1st stage, Boise, Salt Lake and Phoenix

Stages and States of Simple Example

- “Stages” are associated with each move along trip
- Stage 1 consists of Boise, Salt Lake and Phoenix, Stage 2 has Fargo, Omaha and Houston; etc.
- “States” are possibilities in each Stage: Boise, Salt Lake, etc...

![Diagram of a network with cities and distances between them. The distances are: Seattle to Boise 100, Boise to Fargo 500, Fargo to Detroit 200, Boise to Salt Lake 200, Salt Lake to Omaha 400, Omaha to Memphis 150, Omaha to DC 200, Phoenix to Houston 200, Phoenix to Atlanta 300.]
Solution Strategy

- Two Steps
  - Partial optimization at each stage
  - Repetition of process for all stages
- Result of Optimization at each stage is the “cumulative return function”
  - \( f_s(K) \) denotes effect of being in state \( K \), having passed through previous \( S \) stages
  - Example: \( f_2(\text{Omaha}) = 350 \)
  - Defined in terms of best over previous stages and return function for this stage:
    \[
    f_s(K) = [g_i X_i, f_{S-1}(K)]
    \]

Mathematics: Recurrence formulas

- Transition from one stage to next is via a “recurrence formula” (or equivalent analysis)
- For Example: Consider the Maximization investments in independent projects.
  - Each project is a “stage”
  - Amount of Investment in each is its “state”
  - Obj. Fcn. Is Additive: Value = \( \Sigma \) (value each project)
  - Recurrence formula is: \( f_i(K) = \text{Max}[g_i X_i + f_{i-1}(K - X_i)] \)
  - that is: optimum for investing \( K \) over “\( I \)” stages is the maximum of combinations of investing all the levels of \( X_i \) in stage “\( I \)” and \( (K - X_i) \) in previous stages
Application to Investment

- 3 Projects, 4 Possible Investment levels (0, 1, 2, 3)
- Return functions below. Note Non-convex feasible region
- Objective: Maximize value of investing 3 units

Dynamic Programming Analysis (1)

- At 1st stage the cumulative return function identically equals return for $X_1$
- At 2nd stage, best way to spend:
  - 0: is 0 on both 1st and 2nd stage
  - 1: either 1 on 1st and 0 on 2nd stage (= 2) or 0 on 1st and 1 on 2nd stage (=1) BEST
  - 2: 2 on 1st, and 0 on 2nd stage (= 4)
    - 1 on 1st, and 1 on 2nd stage (= 3)
    - 0 on 1st, and 2 on 2nd stage (= 5) BEST
  - 3: 4 Choices, Best allocation is (1,2) ==> 7
Dynamic Programming Analysis (2)

- At 3rd stage we look at another 4 choices and find that optimum allocation is (0,2,1) ==> 8

Contrast Dynamic Programming and Marginal Analysis Solutions

- Marginal Analysis approach to example would
- Invest 1st unit in Project 3 (value = 3)
- Next 2 units in Project 1 (value = 2 each). Total = 7 < 8 by DP
Classes of Problems suitable for Dynamic Programming

- “Dynamic” Problems
  -- aircraft flight paths to maximize speed, altitude
  -- movement across territory (example used)
- Investment Maximizations
  -- Nothing Dynamic. Key is separability of projects
- Schedule, Inventory (Management over time)
- Reliability -- Multiplicative example, see text
- Options analysis!
  -- More on this later in course

Issues in Formulations Problems for Dynamic Programming

- No standard (“canonical”) form
  -- contrast to LP
- Careful formulations required (see text)
- DP assumes discrete states -- thus easily handles integers, discontinuity
  -- Contrast with LP
- DP handles constraints in formulation. In any case, many fewer than LP
- DP doesn’t provide Shadow prices, O. costs
  -- Contrast with LP
## Dynamic Programming Summary

- Handles Issues LP cannot deal with easily
- Solution by implicit enumeration
- Approach requires separability, monotonicity
- Careful formulation needed
- Useful for wide range of issues
- -- in particular for options analyses!