Optimization Methods in Finance **(EPFL, Fall 2010)**

Lecture 9: Dynamic Programming

17.11.2010

Lecturer: Friedrich Eisenbrand *Scribe: Yankai Shao*

Introduction

In real finance world, decisions must often be made in a sequential manner over time. Earlier decisions may affect the feasibility and performance of later decision. In such environments, myopic decisions that optimize only the immediate impact are usually suboptimal for the overall process. To find optimal strategies, one must consider current and future decisions simultaneously. These types of *multi-stage* desicion problems are the typical settings where one employs *dynamic programming*.

Dynamic programming models and methods are based on Bellman's *Principle of Optimality*, namely that for overall optimality in a sequential descision process, all the remaining decisions after reaching a particular state must be optimal with respect to that state.

In the following sections, we are going to introduce the basic set-up of Dynamic Programming based on the Bellman's equation with an asset allocation example and have a glance on the Knapsack problem.

Dynamic Programming

Common elements of DP problems include a *directed graph*, a graph in which each graph edge is replaced by a directed graph edge, a set of possible *states* in each stage, *transitions* from state in one stage to states in the next, *value functions* that measure the best possible objective values that can be achieved starting from each stage, and finally the *recursive relationships* between value functions of different states.

Let's set up the DP problem in the following way with an example at last,

- Directed Graph: $G = (V, A)$ where V is a finite set of *nodes/vertices* or possible *states* in each stage and $A \subseteq V \times V$ is the set of *arcs/edges* .
- Cost of each transitions: $c : A \to \mathbb{R}$ is a function of A, from which we get the value functions.
- Path: A sequence of vertices $v_0, v_1, v_2, \ldots, v_k$, s.t. $(v_{i-1}, v_i) \in A$ with $i = 1, \ldots, k$.
- Lengh/Weight of path: $c(P) = \sum_{i=1}^{k} c(v_{i-1}, v_i)$.
- Cycle: Path $v_0, v_1, \ldots, v_k$ with $v_0 = v_k$.

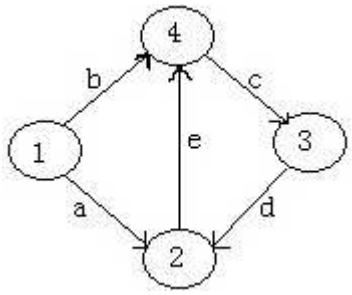


Figure 1: Example for a directed graph $G = (V,A)$. In this case one has $V = \{1,2,3,4\}$ and $A = \{(1,4),(2,4),(1,2),(3,2),(4,3)\}$. The arcs are labelled with their length $a,b,c,d,e \subset \mathbb{R}$.

Shortest Path Problem

In graph theory, the *shortest path problem* is the problem of finding a path between two vertices (or nodes) such that the sum of the weights of its constituent edges is minimized. An example is finding the quickest way to get from one location to another on a road map; in this case, the vertices represent locations and the edges represent segments of road and are weighted by the time needed to travel that segment.

Formally, given a directed graph $G = (V,A)$, the costs $c : A \to \mathbb{R}_+$ and the source node $s \in V$, our task is to determine the length of the shortest path from s to all other nodes. To solve this problem, we start with the Bellman's Equation, which is a necessary condition for dynamic programming problem. We closely follow the *Principle of Optimality* that an optimal policy has the property that whatever the initial state and initial decision are, the remaining decisions must constitute an optimal policy with regard to the state resulting from the first decision.

Bellman's Equations setup:

- Suppose G has no cycles of negative lengths.
- Let $\ell(v)$, for $v \in V$, be the length of the shortest path from s to v. Clearly $\ell(s) = 0$.
- For $(v,u) \in A$, one has $\ell(u) + c(u,v) \geq \ell(v)$.
- If $v \neq s$, there must be a node u on shortest path from s to v immediately preceding v. For this u, $\ell(u) + c(u,v) = \ell(v)$.
- The shortest path $\ell(v)$ satisfying the *Bellman's Equation* has

$$\begin{cases} x_s = 0 \\ x_v = \min\limits_{(u,v)\in A} (x_u + c(u,v)), \quad v \neq s \end{cases}$$

Theorem 1. *If G has no cycle of non-positive length and has a path from s to each other node v, then there exists a unique solution of Bellman Equation, where $x_u = \ell(u)$ is the shortest path for $u \in V$.*

Proof. Let x be a solution of the Bellman equation, we are going to show $x_u = \ell(u)$ is the shortest path. We retreat this directed graph from the last node to s and find the optimal.

Let $v \in V$, $v \neq s$, we get $x_v = x_u + c(u,v)$ for some $(u,v) \in A$. Repeat this process until back to s (no cycle of length 0). This constructs a path $P = (s = v_0, v_1, v_2, \ldots, v_k = v)$, s.t. $c(P) + x_s = x_v = c(P) \Rightarrow x(v) \geq \ell(v)$. Suppose $x \neq \ell$, then there exists a node v with $x_v > \ell_v$ s.t. there exists a node u preceding v on shortest path from s to v has $x_u = \ell(u)$. Bellman's equation implies $\ell(v) = \ell(u) + c(u,v) = x_u + c(u,v) \geq x_v \Rightarrow x_v = \ell(v)$. □

Is there an efficient algorithm to solve the shortest path problem? Two very efficient solutions are the Bellman-Ford algorithm if a graph contains only non-negative cycles and the directed acyclic graph with no directed cycles.

Bellman-Ford Algorithm

Bellman–Ford iterates through all the edges, and does this $|V| - 1$ times, where $|V|$ is the number of vertices in the graph. The repetitions allow minimum distances to accurately propagate throughout the graph, since, in the absence of negative cycles, the shortest path can only visit each node at most once. It runs in $O(|V| \cdot |E|)$ time, where $|V|$ and $|E|$ are the number of vertices and edges respectively.

Assume G has no negative cycle, $V = \{1, 2, \ldots, n\}$, x_i^j is the length of a shortest path from $s = 1$ to i using at most j arcs, $\forall j = 1, \ldots, n-1$. Our task is to find the length of the shortest path from source node s to other nodes.

- Initialization: $x_1^j = 0$, $x_i^j = \infty$, $\forall j = 1, \ldots, n-1, 1 < i \leq n$
- For $j = 1$ to $n-1$, do $x_i^j = \min\limits_{(k,i) \in A} \{x_i^{j-1}, x_k^{j-1} + c(k,i)\}$

Note that the above algorithm has a running time of $O(|V| \cdot |E|)$.

Acyclic Graphs

$G = (V, A)$ is acyclic if G does not contain directed cycles. That is, it is formed by a collection of vertices and directed edges, each edge connecting one vertix to another, such that there is no way to start at some vertix v and follow a sequence of edges that eventually loops back to v again. In this case, arcs induce an ordering $u < v \Leftrightarrow (u,v) \in A$. Assume nodes are set $\{1, 2, \ldots, n\}$. The Bellman's equation acyclic graph is constructed as following,

- $x_1 = 0$, $x_j = \min\limits_{k<j}(x_k + c(k,j))$, which means x_2 is determined from x_1, x_3 is determined from x_2 and x_1, etc.

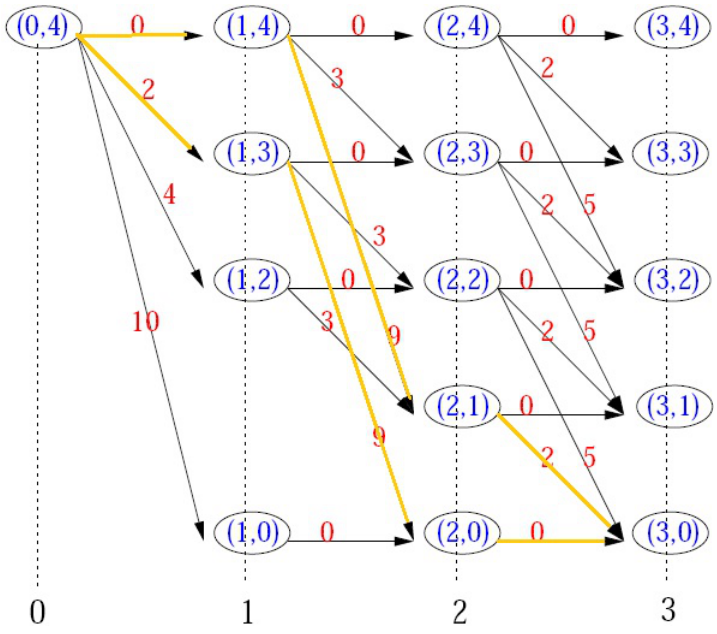
The running time is bounded by $O(|A|+|V|)$.

Dynamic Capital Budgeting

Suppose you have 4 million to be invested in three different regions. At most one project can be run in each region. Question: how much should we invest into different regions and projects in order to maximize our profits? In the table below, we can find the corresponding cost/profit for different projects in different regions.

	Region 1	Region 2	Region 3
Project	cost/profit	cost/profit	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	-	-

In the table, (i, j) represents the optimal choice for the first i regions, $i \in \{0,1,2,3\}$, still having $j \in \{0,1,2,3,4\}$ million to invest. We construct graph with nodes being states and arcs from stages (i, j) to $(i+1, j-c)$, where c is the cost with a profit of p in region $i+1$. Our aim here is to find the longest path in this directed acyclic graph.



From the graph above, we can conclude that the optimal path is $\{(0,4),(1,4),(2,1),(3,0)\}$ or $\{(0,4),(1,3),(2,0),(3,0)\}$ (the yellow paths), both with total length 11.

Knapsack Problem

A traveller has a knapsack that she plans to take along for an expedition. Each item she would like to take with her in the knapsack has a given size and a value associated with the benefit the traveler receives by carrying that item. Given that the knapsack has a fixed and finite capacity, how many of each of these items should she put in the knapsack to maximize the total value of the items in the knapsack? This is the well-known and well-studied integer programm called the *knapsack problem*. It has the special property that it only has a single constraint apart from the non-negative integrality condition on the variables.

Suppose we have n items of weight $w_i \in \mathbb{N}$ and profit $p_i \in \mathbb{N}$, $i = 1,...,N$. The knapsack of capacity $K \in \mathbb{N}$. The task is to pick asset of items of maximum profit under the condition that the subset of items still fit into knapsack.

$$\begin{aligned} \max & \sum_{i=1}^{n} p_i \cdot x_i \\ & \sum_{i=1}^{n} w_i \cdot x_i = K \\ & x_i \in \{0,1\} \quad \forall i = 1,...,n \end{aligned}$$

This problem is different from LP with a new integer constraint, which is called *Integer Programming* (will be covered in the next lecture).

References

[1] Gerard CORNUEJOLS & Reha TÜTÜNCÜ. Optimization Methods in Finance, *Cambridge University Press*, Cambridge (USA), 2007.