

MS-E2148 Dynamic optimization

Lecture 7

- ▶ DP-algorithm and deterministic problems with finite states
- ▶ Application: shortest path problem
- ▶ Application: critical path analysis
- ▶ Application: comparison of DNA sequences
- ▶ Material Bertsekas 2.1 and 2.2.1

Recap

- ▶ The principle of optimality: if $a-b-c$ is optimal route, then $b-c$ is optimal subroute
- ▶ Cost-to-go: what is the cost starting from stage k to the last stage N
- ▶ DP-algorithm: compute the cost-to-go values backwards from $N - 1$ to the initial stage

Deterministic, finite state space

- ▶ **Deterministic:** w_k get one value for each k
 - ▶ Problems that are truly deterministic
 - ▶ Problems that are stochastic but the noise can be assumed approximately constant
- ▶ **Finite state space:** S_k is *finite set* for all k
 - ▶ E.g. time table scheduling problems are finite
 - ▶ Are they finite in modeling altitude of 747 or the balance of bank account?
- ▶ At state x_k the control u_k means deterministic transition to state $f_k(x_k, u_k)$ with cost $g_k(x_k, u_k)$

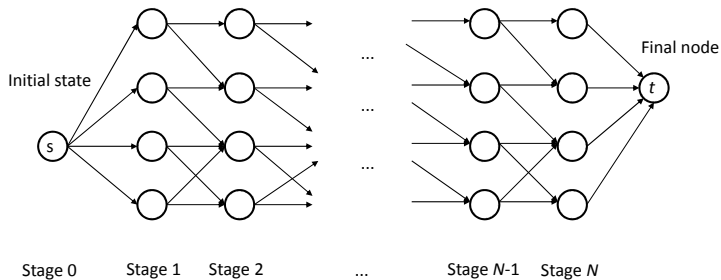
Deterministic, finite state space

- ▶ Special feature: feedback/closed-loop control does not give smaller cost (without stochastics, there is no benefit in observing the state)
- ▶ Minimizing cost over admissible controls $\{\mu_0, \dots, \mu_{N-1}\} \in \Pi$ gives the same cost as minimizing cost over control vectors $\{u_0, \dots, u_{N-1}\}$
- ▶ For some control law $\{\mu_0, \dots, \mu_{N-1}\}$ and initial state x_0 , the future states are perfectly predictable by the equation $x_{k+1} = f_k(x_k, \mu_k(x_k))$, $k = 0, 1, \dots, N-1$, and the corresponding controls can be determined: $u_k = \mu_k(x_k)$
- ▶ This gives computational advantages

Deterministic, finite state space

- ▶ These problems can be presented as *graphs*:
 - ▶ **Nodes** correspond states
 - ▶ **Arcs** correspond transition between consecutive states
 - ▶ The cost of moving from state to another is given by the **length of the arc**
- ▶ Initial state: initial node s that is attached to the states of stage 1
- ▶ Final state: artificial end node t that is attached to each final state x_N with arc that has cost $g_N(x_N)$
- ▶ Controls: paths that start from s and end to some x_N

Deterministic, finite state space



Shortest path problem

- ▶ Transition cost $g_k(x_k, u_k)$ along the arc from state x_k to x_{k+1} is the length of the arc
- ▶ Let us denote: a_{ij} is the length of the arc from node i to j ; if there is no arc then $a_{ij} = \infty$
- ▶ We want to find the **shortest path**, from every node i to the end node t
 \Rightarrow we search for a sequence of controls that minimize the total cost of getting to t from each node $1, 2, \dots, N$

Shortest path problem

- ▶ The solution exists if we require that the length of the *cycles* are non-negative
- ▶ Cycle: $(i, j_1), (j_1, j_2), \dots, (j_k, i)$
- ▶ The optimal path is at most N length: formulate the problem as N -length but allow degenerate transitions from node i to itself so that $a_{ii} = 0$

- ▶ We denote for all $i = 1, \dots, N$ and $k = 0, \dots, N - 1$:
 $J_k(i)$ = optimal cost with $N - k$ steps from node i to end t

Shortest path problem

- ▶ The cost from i to t is then $J_0(i)$
- ▶ **DP algorithm for the shortest path problem:**
the optimal cost from i to t in $N - k$ steps is

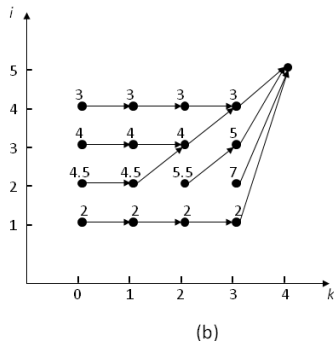
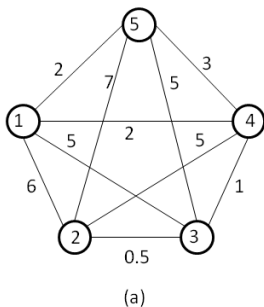
$$J_k(i) = \min_{j=1, \dots, N} [a_{ij} + J_{k+1}(j)], \quad k = 0, 1, \dots, N - 2,$$

where $J_{N-1}(i) = a_{it}$, $i = 1, 2, \dots, N$.

- ▶ Note that the above $J_{k+1}(j)$ = optimal cost from j to t in $N - k - 1$ steps

Shortest path problem: example

- ▶ The figure (a) is a graph where the costs a_{ij} are denoted on the arcs; figure (b) gives the cost-to-go $J_k(i)$ from each node i and stage k generated by DP
- ▶ Node 5 is the end node; $N = 5$, $k = 0, 1, 2, 3, 4$



Shortest path problem: example

- ▶ The cost-to-go values are computed with DP in the following way in figure (b)
 - ▶ $J_2(2) = \min\{5 + 3, 0.5 + 5, 6 + 2, \dots\} = 5.5$ (2 steps)
 - ▶ $J_0(4) = \min\{0 + 0 + 0 + 3, 0 + 0 + 1 + 5, \dots\} = 3$ (4 steps)
- ▶ Zero means staying in the node: $a_{ii} = 0$
- ▶ The optimal paths can be seen from the figure (b):

1 \rightarrow 5,
2 \rightarrow 3 \rightarrow 4 \rightarrow 5,
3 \rightarrow 4 \rightarrow 5,
4 \rightarrow 5

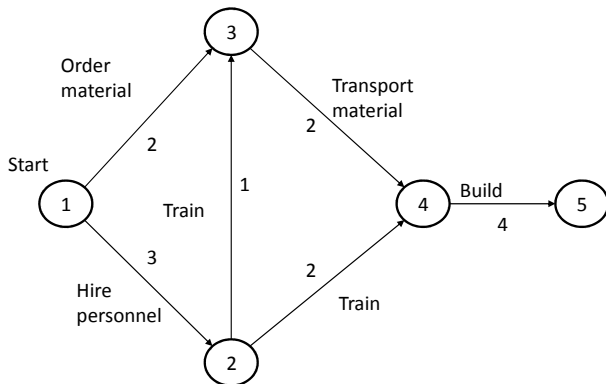
Application: critical path analysis – general questions

- ▶ Project has various activities and some of them need to be finished before starting the next ones
- ▶ How to schedule the project?
- ▶ What are the *critical activities*, which cause the whole project to delay if they are delayed?
- ▶ How long does it take to complete the project (in minimal time)?

Application: critical path analysis

- ▶ Node in the graph represents the ending of a stage
- ▶ Arc (i, j) describes activity that starts when stage i is completed; arc length is the length of the activity $t_{ij} > 0$
- ▶ Stage j is done when all activities that end at node j (arcs (i, j)) are completed
- ▶ Nodes 1 and N are the start and end nodes
- ▶ Graph does not contain cycles

Application: critical path analysis



Application: critical path analysis

- ▶ $p = \{(1, j_1), (j_1, j_2), \dots, (j_k, i)\}$ is path from start to node i
- ▶ D_p is the length of the path: $D_p = t_{1j_1} + t_{j_1j_2} + \dots + t_{j_k i}$
 - ▶ E.g.: $D_{1 \rightarrow 2 \rightarrow 3} = 3 + 1 = 4$, $D_{1 \rightarrow 3} = 2$
- ▶ Completion of stage i takes

$$T_i = \max_p D_p$$

- ▶ E.g.: completion of stage 3 takes $\max\{4, 2\} = 4$
- ▶ To find T_i we need to find the *longest path* from 1 to i
 - ▶ Does the longest path exist? Yes since there are no cycles

Application: critical path analysis

- ▶ The length of the project is the longest path from 1 to N ; this is called the *critical* path
- ▶ Delay in any of the activities on the critical path will delay the whole project with that amount

- ▶ Note: it is a shortest path problem if we multiply t_{ij} by -1

Application: critical path analysis

- ▶ Let S_1 be a group of stages (nodes) that do not depend on completion of any other stage
- ▶ Let S_k , when $k = 1, 2, \dots$, be a set $S_k = \{i \mid \text{all paths from 1 to } i \text{ have } \leq k \text{ arcs}\}$, where $S_0 = \{1\}$.
- ▶ The sets S_k are states of the DP, and DP-algorithm is $T_1 = 0$ and

$$T_i = \max_{(j,i)} [t_{ji} + T_j], \quad \forall i \in S_k, j \in S_{k-1}, i \notin S_{k-1} \quad (1)$$

- ▶ This is "*forward DP-algorithm*" that starts from 1 and ends at N ; computation does not change from the regular DP

Application: critical path analysis

- ▶ For the problem in the figure

$$\begin{aligned}S_0 &= \{1\}, & S_1 &= \{1, 2\}, & S_2 &= \{1, 2, 3\} \\S_3 &= \{1, 2, 3, 4\}, & S_4 &= \{1, 2, 3, 4, 5\}\end{aligned}$$

- ▶ Using the algorithm (1) we get

$$T_1 = 0, T_2 = 3, T_3 = 4, T_4 = 6, T_5 = 10.$$

- ▶ Critical path is $1 \rightarrow 2 \rightarrow 3 \rightarrow 4 \rightarrow 5$

Application: comparison of DNA sequences

- ▶ The task is to compare two DNA sequences and find how similar they are
 - ▶ Sequence consists of nucleotides that are named by their bases **G**(uanine), **A**(denine), **C**(ytosine), **T**(hymine)
 - ▶ Sequence may contain (*gaps*)
-
- ▶ Widely used Needleman-Wunsch algorithm is based on DP-algorithm in bioinformatics

Application: comparison of DNA sequences

- ▶ Let us compare two sequences:

(1) G A A T T C A G T T A

(2) G G A T C G A

- ▶ They can be aligned in the following way:

(1) G A A T T C A G T T A

(2) G G A - T C - G - - A

- ▶ Sequence (2) has gaps
- ▶ Sequence (2) may develop to sequence (1) e.g. by mutation when one base G is transformed to A

Application: comparison of DNA sequences

- ▶ To find out similarity of the sequences, we weight different alignments:

$S_{i,j} = 1$ if base in sequence 1 at place j matches the base of sequence 2 at place i ; otherwise $S_{i,j} = 0$;

$w = 0$ is a penalty for having a gap

- ▶ The alignment on previous slide gives:

$$S_{G,G} + S_{G,A} + S_{A,A} + w + S_{T,T} + S_{C,C} + w + S_{G,G} + \\ 2w + S_{A,A} = 1 + 0 + 1 + 0 + 1 + 1 + 0 + 1 + 0 + 1$$

Application: comparison of DNA sequences

- ▶ The task is to find the alignment with highest score
- ▶ We design an alignment matrix whose elements are:

$$M_{i,j} = \max \left\{ \begin{array}{ll} M_{i-1,j-1} + S_{i,j}, & \text{(match/no match)} \\ M_{i,j-1} + w, & \text{(gap in sequence 2)} \\ M_{i-1,j} + w, & \text{(gap in sequence 1)} \end{array} \right\}$$

- ▶ This is recursion of "*forward-DP*", the text in the bracket gives the "control" how the sequence (2) is built
- ▶ Also assume that $i, j = 1, 2, \dots$ and $M_{0,0} = M_{i,0} = M_{0,j} = 0$

Application: comparison of DNA sequences

► $M_{1,1} = \max\{0 + 1, 0 + 0, 0 + 0\} = 1$

	G	A	A	T	T	C	A	G	T	T	A
G	1										
G											
A											
T											
C											
G											
A											

Application: comparison of DNA sequences

- ▶ $M_{1,2} = \max\{M_{0,1} + S_{1,2}, M_{1,1} + w, M_{0,2} + w\} = 1 + 0$
- ▶ $M_{1,3} = \max\{M_{0,2} + S_{1,3}, M_{1,2} + w, M_{0,3} + w\} = 1 + 0$
- ▶ $M_{2,1} = \max\{M_{1,0} + S_{2,1}, M_{2,0} + w, M_{1,1} + w\} = 0 + 1$

	G	A	A	T	T	C	A	G	T	T	A
G	1	1	1	1	1	1	1	1	1	1	1
G	1										
A	1										
T	1										
C	1										
G	1										
A	1										

Application: comparison of DNA sequences

- ▶ $M_{2,2} = \max\{M_{1,1} + S_{2,2}, M_{2,1} + w, M_{1,2} + w\} = 1 + 0$
- ▶ $M_{3,2} = \max\{M_{2,1} + S_{3,2}, M_{3,1} + w, M_{2,2} + w\} = 1 + 1$

	G	A	A	T	T	C	A	G	T	T	A
G	1	1	1	1	1	1	1	1	1	1	1
G	1	1									
A	1	2									
T	1	2									
C	1	2									
G	1	2									
A	1	2									

Application: comparison of DNA sequences

	G	A	A	T	T	C	A	G	T	T	A
G	1	1	1	1	1	1	1	1	1	1	1
G	1	1	1	1	1	1	1	2	2	2	2
A	1	2	2	2	2	2	2	2	2	2	3
T	1	2	2	3	3	3	3	3	3	3	3
C	1	2	2	3	3	4	4	4	4	4	4
G	1	2	2	3	3	4	4	5	5	5	5
A	1	2	3	3	3	4	5	5	5	5	6

- ▶ Maximum score is 6

Application: comparison of DNA sequences

- ▶ Now we find the alignment that gives score 6 by *traceback*-algorithm, where we move from the element of maximum score to the element $M_{1,1}$
- ▶ **Traceback:** find the predecessor of i, j :
 - ▶ If $M(i, j) = M(i, j - 1)$, seq. (2) has gap at place i
 - ▶ If $M(i, j) = M(i - 1, j)$, seq. (1) has gap at place j
 - ▶ If $M(i, j) > M(i - 1, j), M(i, j - 1), M(i - 1, j - 1)$, then element j of sequence (1) matches with element i of sequence (2)
 - ▶ If several of the above conditions hold at same time, we can move in any of the directions, i.e., there are several alignments that gives the score 6
- ▶ The example has many possible alignments

Application: comparison of DNA sequences

- ▶ Alignment and scoring matrix
- ▶ One possible alignment shown (notice that the sequences could be aligned in other ways, and we still get the score 6)

(1) G A A T T C A G T T A

(2) G G A - T C - G - - A

	G	A	A	T	T	C	A	G	T	T	A
G	1	1	1	1	1	1	1	1	1	1	1
G	1	1	1	1	1	1	1	2	2	2	2
A	1	2	2	2	2	2	2	2	2	2	3
T	1	2	2	3	3	3	3	3	3	3	3
C	1	2	2	3	3	4	4	4	4	4	4
G	1	2	2	3	3	4	4	5	5	5	5
A	1	2	3	3	3	4	5	5	5	5	6

Summary

- ▶ Deterministic problem
- ▶ Finite state space
- ▶ Shortest path problem
- ▶ DP-algorithm in shortest path problem
- ▶ Critical path analysis
- ▶ Comparison of DNA sequences