Discrete Optimization

Quentin Louveaux

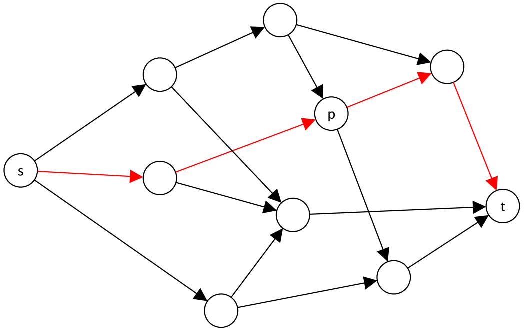
ULg - Institut Montefiore

2016

Dynamic Programming

- Dynamic Programming (DP) is an algorithm for combinatorial problems having the “optimal substructure property.”
- It solves a sequence of subproblems where each subproblem capitalizes on the solution of the previous subproblems
- This is different from divide and conquer (e.g. branch-and-bound) which is a top-down approach that splits the problem in disjoint subproblems, and compares their solutions.
- In DP, subproblems overlap : each subproblem contains the previous subproblems.

Example : shortest path.



- If an optimal path from s to t goes through p
- then it contains the optimal path from s to p and the optimal path from p to t.

Theorem of optimality (Bellman)

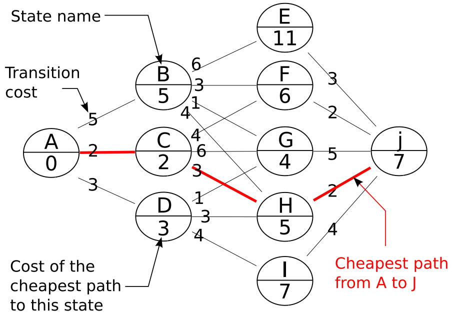
An optimal policy must contain only optimal sub-policies or

a policy is optimal if, at a stated stage, whatever the preceding decisions have been, the decisions still to be taken constitute an optimal policy when the result of the previous decisions is included.

Numerical example : shortest path.

Example :
Optimal path from A to J : ACHJ.

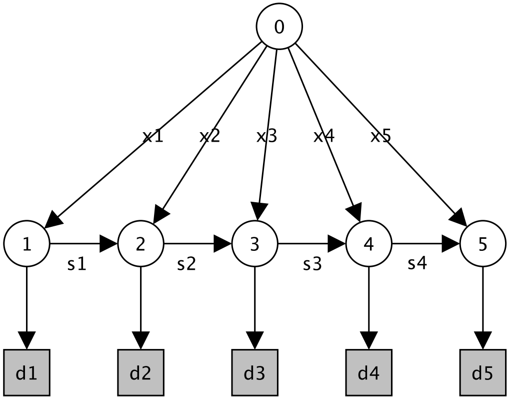
- ACH is the optimal path from A to H.
- CHJ is the optimal path from C to J.



Guidelines for applying DP

- View the choice of a feasible solution as a sequence of decisions occuring in stages, and so that the total cost is the sum of the costs of individual solutions.
- Define the state as a summary of all relevant past decisions.
- Determine which state transitions are possible, i.e. which decision can / cannot be taken in each state (hence the cost of a transition is the cost of the corresponding decision).
- Write a recursion on the optimal cost from the origin state to a destination state. Solving this recursion gives you the optimal cost.
- If needed, keep information on the optimal path from the origin state to a destination state (to ease reconstruction of the solution).

Example : uncapacitated lot sizing



- MIP formulation
- Analysis of the solutions
- DP recursion equation
- Numerical example

Example : Binary Knapsack

- MIP formulation.
- DP recursion equation.
- Complexity.
- Numerical example.

Example : Integer Knapsack

- MIP formulation
- DP recursion equation.
- Complexity.
- Numerical example.

Many diverse applications of DP

- Multiply several matrices while performing the fewest total scalar multiplications.
- Find the longest common subsequence of two sequences (Protein sequences alignment).
- Construct binary search trees that are optimal, given a known distribution of keys to be looked up.
- The Viterbi algorithm (cf. Information Theory course) is a DP algorithm.
- Central role in the Reinforcement Learning paradigm (cf. Applied Inductive Learning course).