ε.δε.μ²
ΕΠΙΣΤΗΜΗ ΔΕΔΟΜΕΝΩΝ & ΜΗΧΑΝΙΚΗ ΜΑΘΗΣΗ

STOCHASTIC PROCESSES & OPTIMIZATION IN MACHINE LEARNING

Reinforcement Learning - Dynamic Programming:

1. Markov Decision Processes
2. Bellman's Optimality Criterion
3. Policy Iteration Algorithm
4. Value Iteration Algorithm

Prof. Vasilis Maglaris

maglaris@netmode.ntua.gr

www.netmode.ntua.gr

Room 002, New ECE Building

Tuesday April 8, 2025

STOCHASTIC PROCESSES & OPTIMIZATION IN MACHINE LEARNING

Reinforcement Learning - Markov Decision Processes

Supervised Learning - Teacher

Machine Learning (ML) configuration tuning in training phase assisted by ***external supervisor*** (teacher), aware of the desired output for all ***labeled*** examples in a ***pre-existing*** training dataset, tested for generalization when the system is fed by new test data

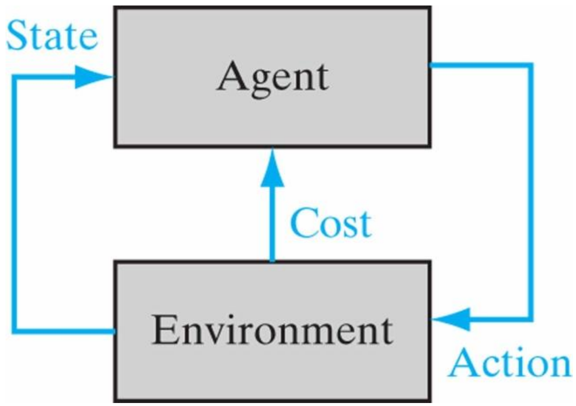
Unsupervised Learning

Self-tuning of ML configuration, based on properties of a ***pre-existing unlabeled*** training examples, tested for generalization when the system is fed by new test data

Reinforcement Learning (RL)

(*Andrew Barto* & *Richard Sutton*, Turing Awards 5/3/2025)

- The ***actions*** of an ***agent*** in a horizon of K steps may control the evolution of the ***states*** of the ***environment*** with cost/reward in current step and anticipated in future state trajectories
- RL involves ***policy planning*** of ***states*** and ***actions*** of the ***agent*** towards medium-long term goals via ***interactive*** learning scenarios
- RL theoretical models: ***Dynamic Programming*** (***DP***), ***Markov Decision Processes*** (***MDP***)
- The ***training dataset*** may be dynamically (***on-line***) specified/updated to reflect decisions of the ***agent*** on the state evolution (***no pre-existence*** of a training dataset is required)



- Training examples in ***Supervised*** & ***Unsupervised Learning*** are usually modeled as independent random variables/vectors in sufficiently large training subsets of the sample space
- In ***Reinforcement Learning*** system configuration is usually based on scenarios of dynamic evolution of ***Markov*** environment states, that depend on control actions of an ***Agent*** associated with a certain ***cost***/***reward***

STOCHASTIC PROCESSES & OPTIMIZATION IN MACHINE LEARNING

Reinforcement Learning - Markov Decision Processes (1/2)

Markov Decision Processes (MDP) Model: State, Action, Cost, Policy

- Finite Sample Space $\mathcal{X}$ of discrete environment ***states*** in steps $n = 0,1,2,\dots,K$

  The ***Random Variable*** $X_n \in \mathcal{X}$ assumes discrete values $X_n = i,\ 1 \le i \le N$

- Finite Sample Space $\mathcal{A}_i$ of discrete ***actions*** of the ***agent*** if the environment state is $X_n = i$

  The ***Random Variable*** $A_n \in \mathcal{A}_i$ of action at step n assumes values a_{ik} when $X_n = i$

- Environment State Transitions: ***Markov*** $p_{ij}(a)$ from i to j with the ***agent*** enforcing action a in transition steps $n = 0,1,2,\dots,K$

  $p_{ij}(a) = P(X_{n+1} = j | X_n = i, A_n = a),\ p_{ij}(a) \ge 0,\ \sum_j p_{ij}(a) = 1$

- The ***observed cost*** of a state transition $(X_n = i) \to (X_{n+1} = j)$ with agent ***action*** a_{ik} is

  $g(i, a_{ik}, j)$ or, anticipated n ***steps ahead,*** $\gamma^n g(i, a_{ik}, j)$ with a ***discount factor*** $0 \le \gamma < 1$

  ✓ If $\gamma = 0$ the ***agent*** is not concerned for the longer-term impact of current action (***myopic***)

  ✓ As $\gamma \to 1$ the ***agent*** actions are determined by their impact on the environment evolution

- A ***policy*** $\pi = \{\mu_0, \mu_1, \dots, \mu_n, \dots, \mu_{K-1}\}$ consists of functions μ_n that map ***states*** $X_n = i$ at step n into ***agent*** action $A_n = a$

  $\mu_n(i) \in \mathcal{A}_i$ for all states $i \in \mathcal{X}$ (π ***admissible policies***)

> If $\mu_n(i) = \mu(i)$ for all steps n, policy $\pi = \{\mu, \mu, \dots\}$ is ***stationary*** and transitions $p_{ij}(a)$ identify a stationary ***Markov Chain*** $(X_n = i) \to (X_{n+1} = j)$

STOCHASTIC PROCESSES & OPTIMIZATION IN MACHINE LEARNING

Reinforcement Learning - Markov Decision Processes (2/2)

Policy Optimization

The total cost is estimated over possible ***trajectories*** in finite steps K (***Finite-Horizon***) of repeated sample ***episodes*** (or as $K \to \infty$ in ***Infinite-Horizon*** scenarios) summing the observed costs of ***Markov Transitions*** $X_n \to X_{n+1}$ under ***action*** $\mu_n(X_n)$:

$$g(X_n, \mu_n(X_n), X_{n+1})$$

The ***Total Discounted Expected Cost-to-Go*** for finite-horizon K and policy $\pi = \{\mu_0, \mu_1, \dots, \mu_{K-1}\}$ from an ***initial*** state $X_0 = i$ and with ***discount factor*** γ is:

$$J^{\pi}(i) = \mathrm{E}\left[\sum_{n=0}^{K} \gamma^n g(X_n, \mu_n(X_n), X_{n+1}) | X_0 = i\right]$$

where the expectation refers to ***Markov Chain*** trajectory frequencies from $X_0 = i$ in K steps

An ***optimal policy*** π minimizes $J^{\pi}(i)$: $J^*(i) \triangleq \min_{\pi} J^{\pi}(i)$

The optimal policy is ***greedy*** in the sense that the ***agent*** minimizes the ***Expected Cost-to-Go*** $J^{\pi}(i)$ from initial state $X_0 = i$ without considering better alternatives in the future as the environment proceeds to a trajectory identified by π

> If the policy space is confined to ***stationary*** decisions, $\pi = \{\mu, \mu, \dots\}$ independent of the transition step n, then $J^{\pi}(i) \triangleq J^{\mu}(i)$ and the problem is to search for the function $\mu(X_n)$ that minimizes $J^{\mu}(i) = J^*(i)$ for all initial states $X_0 = i$

Note: Optimization objectives other than ***Total Discounted Expected Cost-to-Go*** include the ***Expected Average Cost*** per step in ***Infinite Horizon*** with no discount (Sheldon Ross, "**Applied Probability Models with Optimization**", Dover, 1992)

STOCHASTIC PROCESSES & OPTIMIZATION IN MACHINE LEARNING

Principle of Optimality (*Bellman 1957*) – Finite Horizon Problem

Let an ***MDP*** with ***transition costs*** $g_n(X_n, \mu_n(X_n), X_{n+1}) \triangleq \gamma^n g(X_n, \mu_n(X_n), X_{n+1})$, at step $n < K$ and terminal cost $g_K(X_K)$. The ***Expected Cost-to-Go*** in K step expected trajectories $\{X_0, X_1, \dots, X_K\}$ is:

$$J_0(X_0) = \mathrm{E}\left[\left\{g_K(X_K) + \sum_{n=0}^{K-1} g_n(X_n, \mu_k(X_n), X_{n+1})\right\} \mid X_0\right]$$

An ***optimal policy*** $\pi^* = \{\mu_0^*, \mu_1^*, \mu_2^*, \dots, \mu_{K-1}^*\}$ leads the environment in n steps, $n < K$, to possible ***state sub-trajectories*** $\{X_0, X_1, \dots, X_n\}$. The ***Expected Cost-to-Go*** for the ***tail sub-trajectory*** $\{X_{n+1}, X_{n+2}, \dots, X_K\}$ is:

$$J_n(X_n) = \mathrm{E}\left[\left\{g_K(X_K) + \sum_{k=n}^{K-1} g_k(X_k, \mu_k(X_k), X_{k+1})\right\} \mid X_n\right]$$

Then the ***truncated*** policy $\{\mu_n^*, \mu_{n+1}^*, \dots, \mu_{K-1}^*\}$ is optimal for the tail-process (subproblem) $\{X_{n+1}, X_{n+2}, \dots, X_K\}$ with initial state X_n (***Principle of Optimality***)

Justification: if the ***truncated*** policy were not optimal, then the overall optimal policy π^* would lead the environment up to state X_n. The agent could subsequently change the policy for the remainder steps $\{n+1, n+2, \dots, K\}$, thus yield lower total trajectory cost than anticipated using π^*

STOCHASTIC PROCESSES & OPTIMIZATION IN MACHINE LEARNING

Dynamic Programming (*Bellman 1957*) – Finite Horizon Problem

The ***Bellman Principle of Optimality*** leads to ***Dynamic Programming*** formulation for determining an ***Optimal Policy*** $\pi^* = \{\mu_0^*, \mu_1^*, \mu_2^*, \dots, \mu_{K-1}^*\}$ in three stages by ***reversing*** the state transition order: $K \to (K-1) \to (K-2) \to \cdots \to 1 \to 0$

- Determine the optimal policy μ_{K-1}^* for the ***final*** step $X_{K-1} \to X_K$ for all possible X_K
- For the ***tail subproblem*** $X_{K-2} \to X_{K-1} \to X_K$ deterime μ_{K-2}^* without changing μ_{K-1}^*
- Repeat until reaching X_0. μ_0^* , completing the search for the overall optimal policy π^*

Dynamic Programming Algorithm

1. Start with $J_K(X_K) = g_K(X_K)$ for all terminal states X_K
2. For $n = \{K-1, K-2, \dots, 1, 0\}$ evaluate recursively the tail ***Expected Cost-to-Go*** $J_n(X_n)$ for all intermediate states X_n and optimal policies $\mu_n(X_n)$ of the tail subproblems using the ***recursive formula*** of ***greedy*** decisions:

$$J_n(X_n) = \min_{\mu_n(X_n)} \mathrm{E}[g_n(X_n, \mu_n(X_n), X_{n+1}) + J_{n+1}(X_{n+1})]$$

The average in the formula refers to all possible states X_{n+1}

3. Final determination of $J_0(X_0)$ for all initial states X_0 and actions μ_0^* that complement identification of optimal policies $\pi^* = \{\mu_0^*, \mu_1^*, \dots, \mu_{K-1}^*\}$ that satisfy the ***recursive formula***
4. For ***stationary*** policies $\pi = \{\mu, \mu, \dots\}$ the ***recursive formula*** is simplified by letting $\mu_n = \mu$

STOCHASTIC PROCESSES & OPTIMIZATION IN MACHINE LEARNING

Optimality Equations - Infinite Horizon Problem, Stationary Policy

Let an ***MDP*** of fimite states $X_n \in \{1,2,\dots,N\}$, ***stationary policies*** π, ***discount*** γ, ***transition costs*** $g_n(X_n, \mu(X_n), X_{n+1}) \triangleq \gamma^n g(X_n, \mu(X_n), X_{n+1})$ and starting from ***initial state*** X_0

Find a stationary policy π minimizing the ***Expected Cost*** in ***Infinite Horizon*** $n \to \infty$ trajectories

- The ***recursive dynamic programming*** formula is re-formulated by reversing the trajectory evolution, starting from ***initial*** states X_0 over a ***finite horizon*** $\mathrm{n} \le K$:
  $J_{n+1}(X_0) = \min_{\mu} \mathrm{E}[(g(X_0, \mu(X_0), X_1) + \gamma J_n(X_1))|X_0]$ with initial condition $J_0(X) = 0, \forall X$
- Over ***infinite horizon*** and $X_0 = i$ the optimal policy π yields costs $J^*(i) = \lim_{K\to\infty} J_K(i), \forall i \Rightarrow$
  $$J^*(i) = \min_{\pi} \mathrm{E}[(g(i, \mu(i), X_1) + \gamma J^*(X_1))|X_0 = i]$$
- Define $c\big(i, \mu(i)\big)$ the ***Immediate Expected Cost*** of environment state $X_0 = i$ and action $\mu(i)$:
  $$c\big(i, \mu(i)\big) \triangleq \mathrm{E}[g(i, \mu(i), X_1 = j)|X_0 = i] = \sum_{j=1}^{N} p_{ij}(\mu(i)) g(i, \mu(i), j)$$
  The average term refers to possible states X_1 resulting from X_0 in one-step transitions
- The optimal μ yields one-step transition cost $\mathrm{E}[J^*(X_1)|X_0 = i] = \sum_{j=1}^{N} p_{ij}(\mu) J^*(j)$

We obtain N equations referred to as ***Bellman's Optimality Equations***:

$$J^*(i) = \min_{\mu}\left(c\big(i, \mu(i)\big) + \gamma \sum_{j=1}^{N} p_{ij}(\mu(i)) J^*(j)\right), \qquad i = 1,2,\dots,N$$

These N equations determine $J^*(i)$ via ***Policy Iteration*** or ***Value Iteration algorithms***

Caution: We assume knowledge of $p_{ij}(a)$ in what is referred to as ***Model-based learning***

STOCHASTIC PROCESSES & OPTIMIZATION IN MACHINE LEARNING

Model-based Learning: Policy Iteration (1/2)

Q-factors

- A stationary policy $\pi = \{\mu, \mu, \dots\}$ leads to ***costs-to-go*** $J^{\mu}(i), \forall i \in \mathcal{X}$ (the ***environment state*** space) with ***agent action*** $a = \mu(i) \in \mathcal{A}_i$
- At every step and for all (i, a) pairs and tail-policies $\pi = \{\mu, \mu, \dots\}$ define the ***Q-factors*** $Q^{\mu}(i, a)$ as a comparison metric of alternative direct ***agent*** actions $a \in \mathcal{A}_i$ that would lead the ***environment*** from present state i to state j with expected ***costs-to-go*** $J^{\mu}(j), \forall j \in \mathcal{X}$

$$Q^{\mu}(i, a) \triangleq c(i, a) + \gamma \sum_{j=1}^{N} p_{ij}(a)\, J^{\mu}(j)$$

- A stationary policy $\pi = \{\mu, \mu, \dots\}$ satisfies the ***greedy conditions*** regarding the expected ***costs-to-go*** $J^{\mu}(j)$ for the remaining transitions, if the ***agent*** at every step and $\forall i \in \mathcal{X}$ selects $a = \mu(i)$ so that

$$Q^{\mu}(i, \mu(i)) = \min_{a \in \mathcal{A}_i} Q^{\mu}(i, a), \forall i \in \mathcal{X}$$

- A policy $\pi^* = \{\mu^*, \mu^*, \dots\}$ is optimal for all steps if it satisfies the ***greedy conditions*** τof dynamic programming:

$$Q^{\mu^*}(i, \mu^*(i)) = \min_{a \in \mathcal{A}_i} Q^{\mu^*}(i, a)$$

Note: In a dual formulation cost ***minimization*** is translated as reward ***maximization***, ***costs*** $c(i, a)$ are defined as ***rewards*** $r(i, a)$, the ***costs-to-go*** $J^{\mu}(i)$ are referred to as ***Value Functions*** $V^{\mu}(i)$, and the ***Q-factors*** are:

$$Q^{\mu}(i, a) \triangleq r(i, a) + \gamma \textstyle\sum_{j=1}^{N} p_{ij}(a)\, V^{\mu}(j) \text{ and } Q^{\mu^*}(i, \mu^*(i)) = \max_{a \in \mathcal{A}_i} Q^{\mu^*}(i, a)$$

STOCHASTIC PROCESSES & OPTIMIZATION IN MACHINE LEARNING

Model-based Learning: Policy Iteration (2/2)

Actor - Critic Architecture

(***A.G. Barto***, ***R.S. Sutton*** & ***C.W. Anderson***, "Neuronlike adaptive elements that can solve difficult learning control problems," IEEE Transactions on Systems, Man, and Cybernetics, vol. SMC-13, Sept. – Oct. 1983)

Iterations $n = 0,1,2,\dots$ of steps until convergence: $\mu_{n+1}(i) = \mu_n(i)$, $J^{\mu_{n+1}}(i) = J^{\mu_n}(i)$, $\forall i$

Step 1. **Policy Evaluation** (the ***critic*** evaluates the ***agent actions***):

Based on current policy $\pi_n = \{\mu_n, \mu_n, \dots\}$ evaluate ***costs-to-go***:

$$J^{\mu_n}(i) = c\big(i, \mu_n(i)\big) + \gamma \sum_{j=1}^{N} p_{ij}(\mu_n(i))\, J^{\mu_n}(j) \text{ for } \forall i$$

For $\forall i$ and $\forall a \in \mathcal{A}_i$ evaluate ***Q-factors***: $Q^{\mu_n}(i,a) = c(i,a) + \gamma \sum_{j=1}^{N} p_{ij}(a)\, J^{\mu_n}(j)$

Step 2. **Policy Improvement** (the ***actor*** guides the ***agent decisions***):

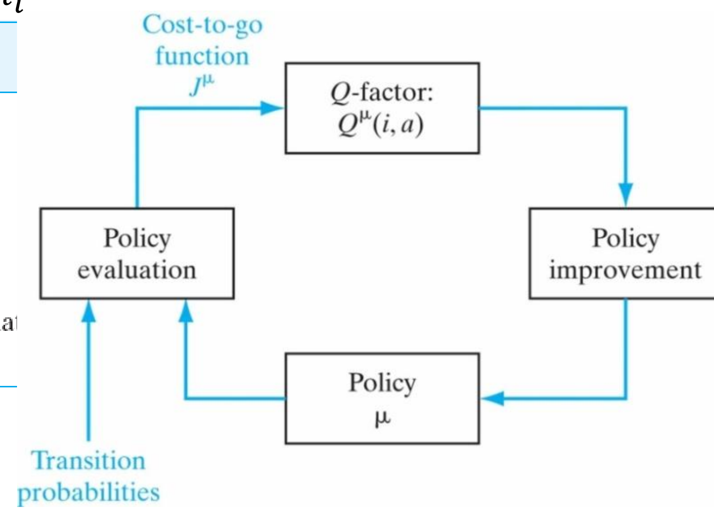
Policy π_n is updated to π_{n+1} by updating $\mu_{n+1}(i) = \arg\min_{a \in \mathcal{A}_i} Q^{\mu_n}(i,a)$ for $i = 1,2,\dots,N$

TABLE 12.1 Summary of the Policy Iteration Algorithm

1. Start with an arbitrary initial policy μ_0.
2. For $n = 0, 1, 2, \dots$, compute $J^{\mu_n}(i)$ and $Q^{\mu_n}(i, a)$ for all states $i \in \mathscr{X}$ and actions $a \in \mathscr{A}_i$.
3. For each state i, compute

$$\mu_{n+1}(i) = \arg\min_{a \in \mathscr{A}_i} Q^{\mu_n}(i, a)$$

4. Repeat steps 2 and 3 until μ_{n+1} is not an improvement on μ_n, at which point the algorithm terminat with μ_n as the desired policy.



$$\arg\min_x f(x)$$

The value of independent variable x for which $f(x)$ reaches a minimum

The algorithm converges to an optimal policy in finite steps n due to finite state-space N and finite action space

STOCHASTIC PROCESSES & OPTIMIZATION IN MACHINE LEARNING

Model-based Learning: Value Iteration Algorithm

Estimation of Cost-to-Go via Successive Approximations $J_n(i) \rightarrow J_{n+1}(i)$

- ***Start*** with arbitrary initial values for $J_0(i), \forall i$
- ***Iterate*** $n \rightarrow n+1$ until ***acceptable convergence*** (In theory $n \rightarrow \infty$) via ***Bellman's*** equations
- ***Final*** evaluation of (sub)optimal ***Costs-to-Go***:

$$J^*(i) = \lim_{n\rightarrow\infty} J_n(i),\ Q^*(i,a) = c(i,a) + \gamma \textstyle\sum_{j=1}^{N} p_{ij}(a) J^*(j)$$

and ***determination*** of ***optimal policy***: $\mu^*(i) = \arg\min_{a \in \mathcal{A}_i} Q^*(i,a)$ για $i = 1,2,\ldots,N$

TABLE 12.2 Summary of the Value Iteration Algorithm

1. Start with arbitrary initial value $J_0(i)$ for state $i = 1, 2, \ldots, N$.
2. For $n = 0, 1, 2, \ldots$, compute

$$J_{n+1}(i) = \min_{a \in \mathcal{A}_i} \left\{ c(i,a) + \gamma \sum_{j=1}^{N} p_{ij}(a) J_n(j), \right\}, \qquad \begin{array}{l} a \in \mathcal{A}_i \\ i = 1, 2, \ldots, N \end{array}$$

Continue this computation until

$$|J_{n+1}(i) - J_n(i)| < \epsilon \qquad \text{for each state } i$$

where ϵ is a prescribed tolerance parameter. It is presumed that ϵ is sufficiently small for $J_n(i)$ to be close enough to the optimal cost-to-go function $J^*(i)$. We may then set

$$J_n(i) = J^*(i) \qquad \text{for all states } i$$

3. Compute the Q-factor

$$Q^*(i,a) = c(i,a) + \gamma \sum_{j=1}^{N} p_{ij}(a) J^*(j) \qquad \begin{array}{l} \text{for } a \in \mathcal{A}_i \text{ and} \\ i = 1, 2, \ldots, N \end{array}$$

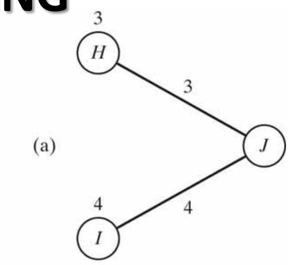
Hence, determine the optimal policy as a greedy policy for $J^*(i)$:

$$\mu^*(i) = \arg\min_{a \in \mathcal{A}_i} Q^*(i,a)$$

- The ***Value Iteration*** algorithm, if it converges in an acceptable run-time, avoids evaluations of ***Q-factor*** and policy updates at every step unlike ***Policy Iteration***
- Assumes (as ***Policy Iteration***) a priori knowledge of $p_{ij}(a)$ (***Model-based Learning***)
- Alternatively ***Model-free Learning*** methods search for optimal policies without prior knowledge of $p_{ij}(a)$, e.g. via ***Monte Carlo*** trajectory simulations, algorithms for ***Q-Learning*** estimation…

STOCHASTIC PROCESSES & OPTIMIZATION IN MACHINE LEARNING

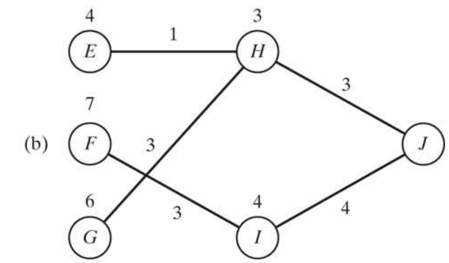
Dynamic Programming Example: The Stagecoach Problem



Determine the **least cost** path (route) from ***Node*** A to ***Node*** J via the directional graph in the figure, with directions pointing ***L* → *R***

Representative Edge Costs: $A \rightarrow B: 2,\ B \rightarrow A: \infty$

$B \rightarrow F: 4,\ F \rightarrow B: \infty$



Representative Path Cost: Path $\{A, B, F, I, J\}$: $2 + 4 + 3 + 4 = 13$

Environment State: Node under consideration $\{A, B, \dots, J\}$

Agent Action: Next node in the path $\{up, down, staight\}$

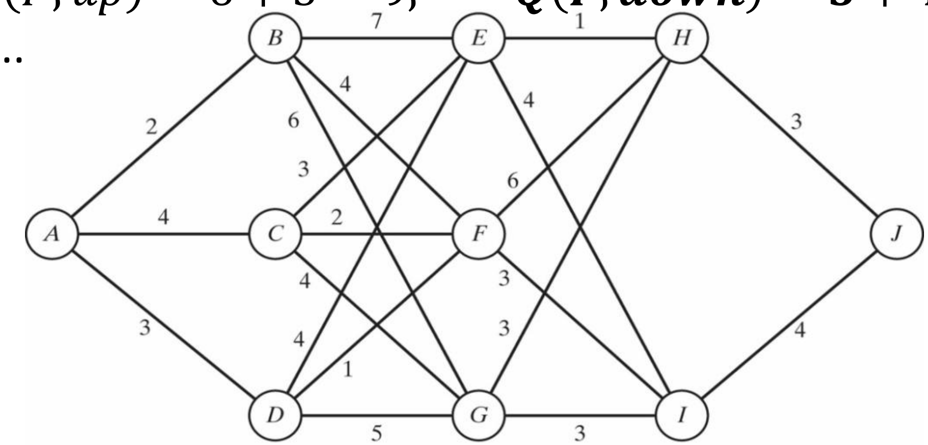
Recurive Evaluation of *Q-Factors* (the best choices in **bold**):

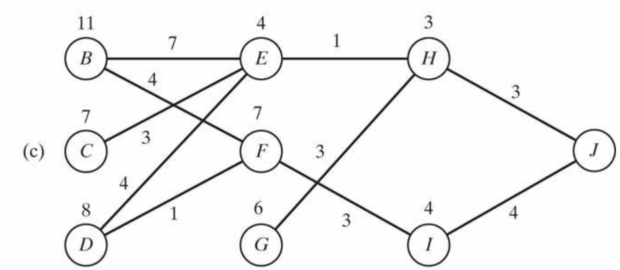
$\boldsymbol{Q(H, down) = 3}$, $\boldsymbol{Q(I, up) = 4}$

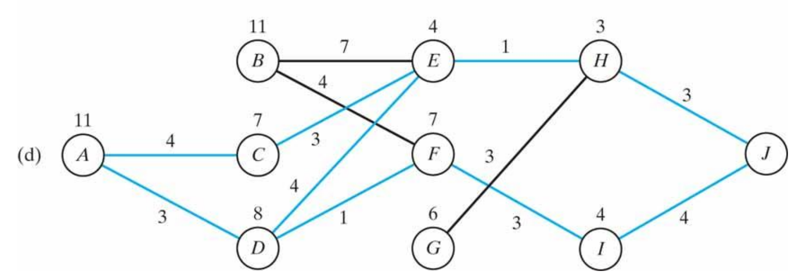
$\boldsymbol{Q(E, staight) = 1 + 3 = 4}$, $Q(E, down) = 4 + 4 = 8$

$Q(F, up) = 6 + 3 = 9$, $\boldsymbol{Q(F, down) = 3 + 4 = 7}$

.....







Direction of Edges

L (Left) ⟶ ***R*** (Right)

Optimal Path Cost 11:

$\{A, C, E, H, J\}, \{A, D, E, H, J\}, \{A, D, F, I, J\}$

Dynamic Programming Algorithms (***Bellman-Ford***) support global Internet Routing (***Border Gateway Protocols* - *BGP***) specified by the ~78,000 ***Autonomous Systems*** (***AS***) of the ***Internet*** to the ~1,000,000 known network destinations