1 Approximate DP

ADP (learning-based MDP) is needed in place of DP (MDP) for sequential decision making problems if the following occurs

1. Transition probabilities are not known
2. Number of system states is large (remember the difference between high dimenstional and large system state)

1.1 Issues due the above

1. Since transition probabilities are not known (curse of modeling), a simulation framework is needed to generate the Markov jumps. **Solution:** Use a learning-version of Bellman's equation, which will only yield near-optimal solutions under certain conditions. This will cause convergence to a band but not to a point like the g(R) as in MDP. This will cause the results to be approximate.
2. Value functions cannot be stored due to high computational storage requirement for the large number of states (curse of dimensionality). Also, reading and writing into a large matrix of states and its value function values is not computationally feasible. Therefore, value functions must be estimated. **Solution:** value function approximation with or without state space aggregation. This will cause the results to be approximate.
3. Synchronous update as in MDP where the values of all states are updated in every iteration is not possible due to high computational time to update the values because of the large number of states: **Solution:** Asynchronous update of only one state in each iteration, which introduces the notion of sampling one realization of the Markov jump. This will cause slower convergence and will need millions of iterations.
4. Since values of states are no longer stored, the actions are not tractable. A scheme to find the best actions must be devised. **Solution:** Create an argmin or argmax equation.
5. Since ADP is an unsupervised learning scheme, exploration, learning and learnt phases are required to obtain the near-optimal solutions.

1.2 Robbins-Monro Stochastic Approximation Scheme

1.3 The notion of a neighborhood jump

1.4 Definition of a post decision state

1.5 Learning-version of Bellman's equation with pre-decision state

$$\bar{V}^n(S^n) = (1-\alpha^n)\bar{V}^{n-1}(S^n) + \alpha^n(v^n) \tag{1}$$

where v^n is one sample realization of the stochastic process (one Markov jump)

$$v^n = \min_{x^n}[Expected\ Cost\ c(S^n, x^n) + \beta \sum_j p(S^n, x^n, S^{n+1})\bar{V}^{n+1}(S^{n+1})] \tag{2}$$

$$x^n(S^n) = arg\min_{x^n}[Expected\ Cost\ c(S^n, x^n) + \beta \sum_j p(S^n, x^n, S^{n+1})\bar{V}^{n+1}(S^{n+1})] \tag{3}$$

where α^n is a learning parameter that is decayed over the iterations.

1.6 Learning-version of Bellman's equation with post-decision state

$$\bar{V}^n(S^{x,n}) = (1-\alpha^n)\bar{V}^{n-1}(S^{x,n}) + \alpha^n(v^{n+1}) \tag{4}$$

where v^{n+1} is one sample realization of the stochastic process (one Markov jump)

$$v^{n+1} = \min_{x^{n+1}}[Expected\ Cost\ c(S^{n+1}, x^{n+1}) + \beta\bar{V}^{n+1}(S^{x,n+1})] \tag{5}$$

$$x^{n+1}(S^{n+1}) = arg \min_{x^{n+1}}[Expected\ Cost\ c(S^{n+1}, x^{n+1}) + \beta\bar{V}^{n+1}(S^{x,n+1})] \tag{6}$$

Note the $p(S^n, x^n, S^{n+1})$ factor is no longer there in the post-decision state version. Final learning equations are

$$\bar{V}^n(S^{x,n}) = (1-\alpha^n)\bar{V}^{n-1}(S^{x,n}) + \alpha^n(\min_{x^{n+1}}[c(S^{n+1}, x^{n+1}) + \beta\bar{V}^{n+1}(S^{x,n+1})]) \tag{7}$$

$$x^{n+1}(S^{n+1}) = arg \min_{x^{n+1}}[c(S^{n+1}, x^{n+1}) + \beta\bar{V}^{n+1}(S^{x,n+1})] \tag{8}$$

To derrive the above and to solve this with value function approaximation and to get convergence to a band for large size stochastic DP problems, take OR 774.