Introduction to Stochastic Dynamic Programming

Justin C. Goodson

1 Modeling

A stochastic dynamic program is a model of an optimization problem that consists of the sequence *decision, information, decision, information, decision, . . .*

A Markov decision process (MDP) is a type of stochastic dynamic program. It consists of:

Decision Epochs

States

Actions

Random Information

Contributions

Transition Function

Objective

The process proceeds as follows:

- Decisions are made at epochs $0, 1, \ldots, K$
- At epoch k the system occupies state s_k
- In state s_k, choose action x_k from the set of feasible decisions $\mathcal{X}(s_k)$
- The expected contribution accrued is $C_k(s_k, x_k)$
- A realization of random information w_{k+1} is observed
- The transition to $s_{k+1} = S(s_k, x_k, w_{k+1})$ is made
- Begin at initial state s_0 and repeat until reaching a terminal state s_K

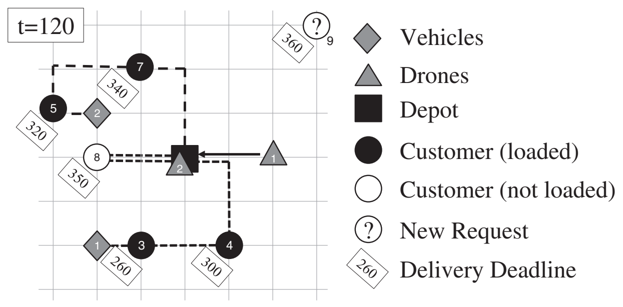
The optimization proceeds as follows:

- A policy π is a sequence of decision rules, $\pi = (X_0^\pi, X_1^\pi, \ldots, X_K^\pi)$
- A decision rule $X_k^\pi(s_k) : s_k \mapsto \mathcal{X}(s_k)$ maps states to feasible actions
- The set of all policies is Π
- The objective is to optimize the total expected contribution, conditional on the initial state:
$$\max_{\pi \in \Pi} \mathbb{E} \left[\sum_{k=0}^{K} C_k \left(s_k, X_k^\pi \left(s_k \right) \right) \middle| s_0 \right]$$

Draw a picture of the stochastic dynamic program.

Example. "Same-Day Delivery with Heterogeneous Fleets of Drones and Vehicles," Ulmer & Thomas. *Networks*, 2018:72, 475-505.

Problem Description. A combined fleet of vehicles and drones deliver goods from a depot to customers who dynamically request service over a finite operating horizon. Vehicles operate across a road network, have unlimited capacity, and travel slowly. In contrast, drones travel quickly in straight-line distances, can carry only one package, and must recharge between trips. Requests follow a known spatial-temporal distribution, but their times and locations are unknown before the request is made. Requests may be accepted or rejected. Accepted requests must be serviced by a hard deadline and may be satisfied via drone or vehicle. The objective is to maximizes the expected number of serviced requests.



Model. Review Section 3.3 to identify decision epochs, states, actions, information, contributions, transition function, and objective. Briefly describe these elements in words. Mathematical formalism is not required.

2 Essential Theory

A stochastic dynamic program asks us to identify a policy with value

An exact solution method identifies an optimal policy in recursive fashion. We show that the value of any policy can be calculated recursively. A backwards solution method for an optimal policy then follows.

The expected total contribution from state s_k if we follow policy π:

We can show that $F_k^\pi(s_k)$ can be calculated recursively as

Proposition 1. $F_k^\pi(s_k) = V_k^\pi(s_k)$.

The value of an optimal policy from state s_k is $F_k^\star(s_k) = \max_{\pi \in \Pi} F_k^\pi(s_k)$. We can show that $F_k^\star(s_k)$ may be calculated recursively as

This is called the *value function*. We can show that solving the value functions identifies the value of an optimal policy.

Theorem 1. $F_k^\star(s_k) = V_k(s_k)$.

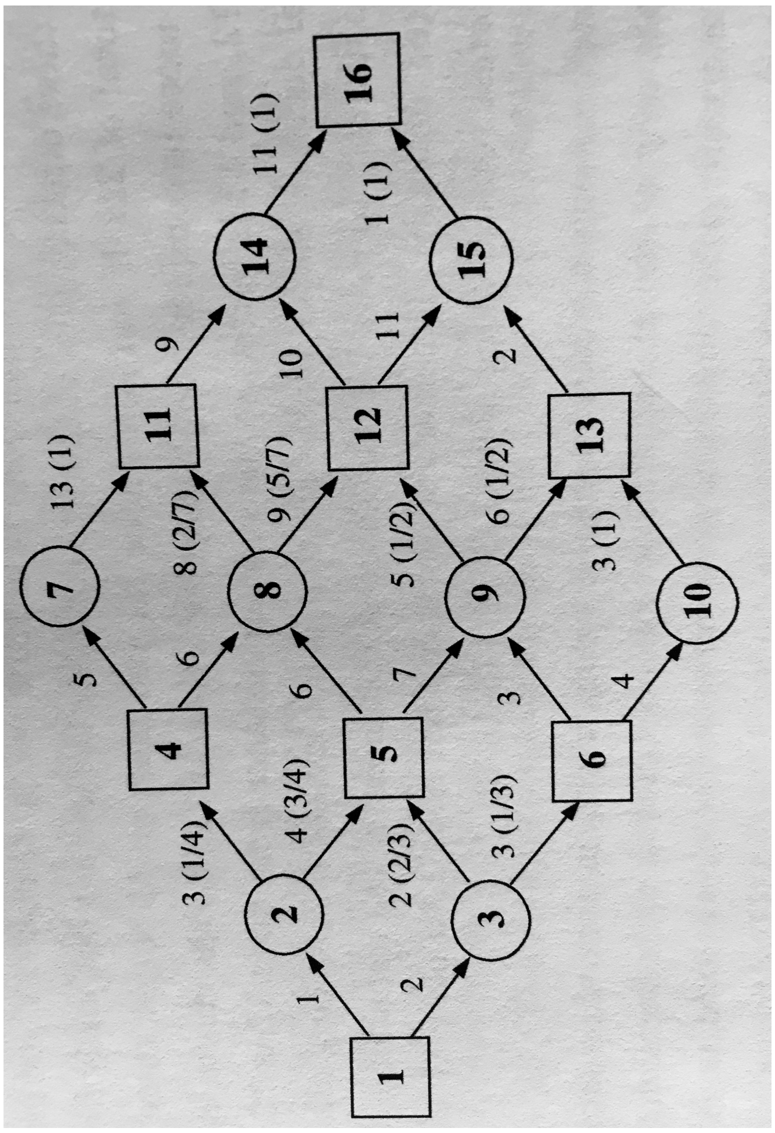
Consequently, an optimal policy can be identified by choosing an action that satisfies the value function:

This result leads to the *Backward Induction* algorithm for finite horizon stochastic dynamic programs:

In many problems, it may be helpful to build the decision tree first. Begin with initial state s_0, then use the model to transition until the terminal states are reached.

Become familiar with the backward induction algorithm by identifying the expected value of the shortest path through the network below. At each decision node (squares), you must choose an arc. The choice results in a known reward plus a stochastic reward depending on which outcome (circle nodes) is realized. Complete the calculations by identifying the value function at nodes 4 and 1. The value at node 1 is the value of an optimal policy.

$$\begin{aligned}
V_{16} &= 0 \\
V_{13} &= 2 + 1(1) = 3 \\
V_{12} &= \min\{10 + 11(1), 11 + 1(1)\} = \min\{21, 12\} = 12 \\
V_{11} &= 9 + 11(1) = 20 \\
V_6 &= \min\{3 + (5 + V_{12})(1/2) + (6 + V_{13})(1/2), 4 + (3 + V_{13})(1)\} \\
&= \min\{16, 10\} = 10 \\
V_5 &= \min\{6 + (8 + V_{11})(2/7) + (9 + V_{12})(5/7), 7 + (5 + V_{12})(1/2) + (6 + V_{13})(1/2)\} \\
&= \min\{6 + 23, 7 + 13\} = 20
\end{aligned}$$


1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
1
2
3 (1/4)
4 (3/4)
2 (2/3)
3 (1/3)
5
6
6
7
3
4
13 (1)
8 (2/7)
9 (5/7)
5 (1/2)
6 (1/2)
3 (1)
9
10
11
2
11 (1)
1 (1)

3 Policies

To identify an optimal policy $\pi^\star$, solve the value function for every state:

$$X_k^{\pi^\star}(s_k) = \underset{x \in \mathcal{X}(s_k)}{\arg\max} \left\{ C_k(s_k, x_k) + \mathbb{E}\left[V_{k+1}(s_{k+1}) \middle| s_k, x \right] \right\}.$$

Solving the value functions presents several challenges:

Exact solution methods are computationally intractable for many problems of practical interest. It is often difficult or impossible to fully characterize the structure of an optimal policy via mathematical analysis. Often, we must resort to heuristic solution methods.

Recall the definition of a policy:

Within this definition, we can be very creative in choosing policies.

There are two fundamental strategies for creating policies, each of which can be further divided into two classes, creating four classes of policies. The two strategies are:

General Strategy 1: Policy Search

Policy Class 1: Policy Function Approximations (PFAs)

PFAs are functions that map a state to a feasible action. Useful when you have a good idea of how to make a decision and can design a function that captures the structure of the problem.

Policy Class 2: Cost Function Approximations (CFAs)

Select actions using a parameterized optimization model. Both objective and constraints may depend on the parameters:

$$X^{\text{CFA}}(s|\theta) = \arg\max_{x \in \mathcal{X}(s|\theta)} \bar{C}_k(s, x|\theta)$$

CFAs are like parametric PFAs but with explicit optimization over actions.

General Strategy 2: Lookahead Approximations

Policy Class 3: Value Function Approximations (VFAs)

Policies that approximate the value of being in a state, typically via the value function.

Policy Class 4: Direct Lookahead Policies (DLAs)

DLAs establish the value of being in a state by optimizing over some approximation of what lies ahead.

Policy Evaluation

No matter what policiy we design, we must evaluate it. The value of a policy is the expected sum of its contributions:

$$F_0^\pi(s_0) = \mathbb{E}\left[\sum_{k=0}^{K} C_k(s_k, X_k^\pi(s_k)\Big| s_0\right]$$

When $F_0^\pi(s_0)$ is difficult to calculate, we can use simulation to estimate it. Simulate $w = (w_1, w_2, \ldots, w_{K+1})$, a realization of random information $W = (W_1, W_2, \ldots, W_{K+1})$. Then, we have the sample path $(s_0, X_0^\pi(s_0), w_1), (s_1, X_1^\pi(s_1), w_2), \ldots, (s_K, X_K^\pi(s_K), w_{K+1})$. Simulate N sample paths and calculate

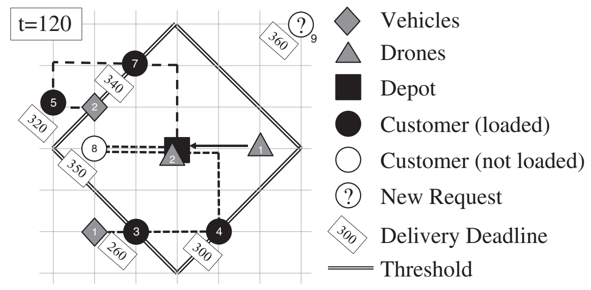
By the law of large numbers, $\hat{F}_0^\pi(s_0)$ is an unbiased and consistent estimator of $F_0^\pi(s_0)$.

4 Case Study: Policy Function Approximation

"Same-Day Delivery with Heterogeneous Fleets of Drones and Vehicles," Ulmer & Thomas. *Networks*, 2018:72, 475-505.

Problem Description. A combined fleet of vehicles and drones deliver goods from a depot to customers who dynamically request service over a finite operating horizon. Vehicles operate across a road network, have unlimited capacity, and travel slowly. In contrast, drones travel quickly in straight-line distances, can carry only one package, and must recharge between trips. Requests follow a known spatial-temporal distribution, but their times and locations are unknown before the request is made. Requests may be accepted or rejected. Accepted requests must be serviced by a hard deadline and may be satisfied via drone or vehicle. The objective is to maximizes the expected number of serviced requests. For more details, see the illustrative example in Section 3.2.

Policy Function Approximation. Study Section 4 and describe the PFA. How are states mapped to actions? Is the PFA a lookup table, a parametric function, or a nonparametric function? State the class of policies characterized by the PFA. How is the best policy within the class identified?

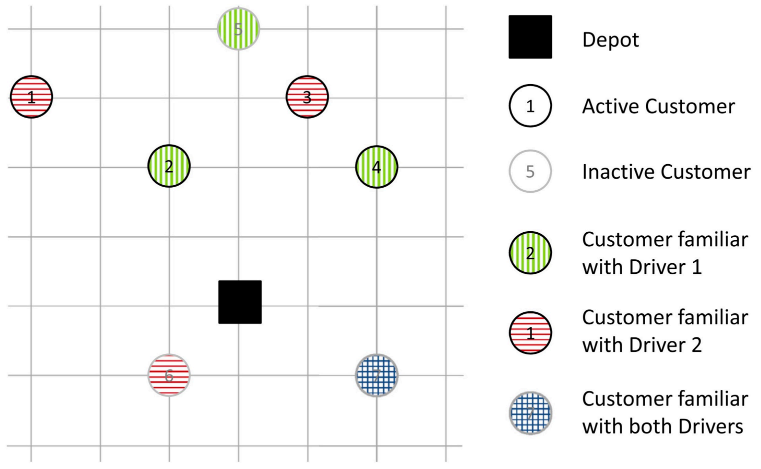


Does it work? Look at Figure 4. Is the policy successful?

5 Case Study: Cost Function Approximation

"Binary Driver-Customer Familiarity in Service Routing," Ulmer, Nowak, Mattfeld, & Kaminski. *European Journal of Operational Research*, 2020:286, 477-493.

Problem Description. Over a sequence of days, a provider serves customers with a fleet of drivers. Drivers' working time each day is limited. Customers request service intermittently. Requests are known to the provider at the beginning of each day and the driver designs routes to service these requests. Cost is determined by time required to traverse routes and service customers. Service time depends on whether the assigned driver is familiar with the customer. If the driver has previously serviced the customer, then this leads to a smaller service time. Consequently, a routing decision made today can impact service times on subsequent days. For example, consider a set of retail stores serviced from a central warehouse. Stores frequently request delivery, but not every day. A driver becomes familiar with a customer by locating the store, finding the correct unloading area, reporting to the store manager, unloading goods, and completing delivery review. On the first visit, these steps take some time, but on subsequent visits, because the driver is familiar with the customer, the process proceeds faster. The objective is to minimize expected routing and service costs.


5
1
3
2
4
6
Depot
1
Active Customer
5
Inactive Customer
2
Customer familiar with Driver 1
1
Customer familiar with Driver 2
Customer familiar with both Drivers

Model.

- *Decision Epochs.* Decisions are made at the beginning of each day.
- *States.* A state consists of the set of active customers and the current familiarity between drivers and customers. The familiarity matrix induces a service time matrix.

- *Actions.* An action is an assignment of active customers to drivers as well as the routing of drivers. The set of feasible actions is defined by the constraints of an uncapacitated VRP with route duration limits.
- *Information.* A set of new customer requests, revealed at the beginning of the day.
- *Contribution.* The sum of routing and service costs.
- *Transition function.* An action updates the familiarity matrix. New requests update the set of active customers.
- *Objective.* Minimize expected sum of contributions.

Cost Function Approximation. The CFA is a myopic policy plus a cost correction term. The contribution function in Equation (2) is augmented with the term

$$I_{ikp} = (1 - f_{ikp}) \times \mathcal{M} \times \left(\frac{1}{|F(i)| + 1} \right)^q \times n_{ikp}.$$

Study Section 4 and describe the CFA. What is the function of each part of I_{ikp}? How are states mapped to actions, meaning how does the CFA choose a feasible action from a given state? If the CFA is parametric, how are the parameter values chosen?

Does it work? Look at the first bar in Figure 4. Is the policy successful? If yes, how can you tell?

6 Case Study: Value Function Approximation

"Meso-Parametric Value Function Approximation for Dynamic Customer Acceptances in Delivery Routing," Ulmer & Thomas. *European Journal of Operational Research*, 2020:285, 183-195.

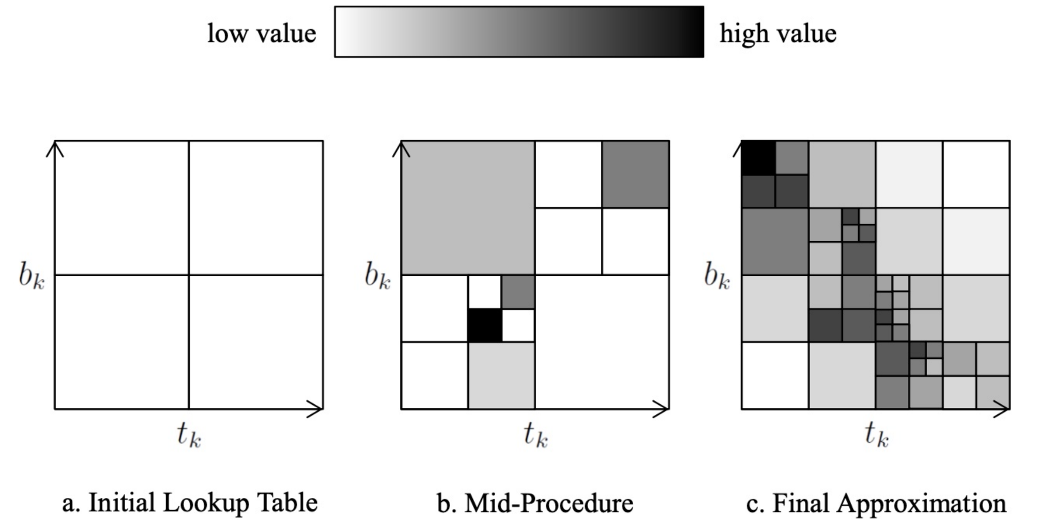
Problem Description. A dispatcher receives delivery requests at random times and from random locations across a geographic area serviced by a single vehicle. The dispatcher must decide to accept or reject requests. If accepted, a request requires some amount of the vehicle's capacity and yields some revenue. The objective is to maximize expected revenue subject to constraints on vehicle capacity and total service time.

Model.

- *Decision Epochs.* Decisions occur whenever a request for service is received. The number of epochs is a random variable.
- *States.* A state consists of the time that a request occurs; the set of already accepted orders; the new order, its location, load, and revenue; and the current route plan.
- *Actions.* An action is a pair consisting of a decision to accept or reject the request and, if accepted, a route plan to serve the request. Route plans must respect constraints on capacity and duration.
- *Information.* The time and location of a new request.
- *Contribution.* The revenue associated with the new request, if it is accepted, otherwise zero.
- *Transition function.* If an action accepts a request, then this updates the route plan and the set of already accepted orders. Otherwise, these state elements do not change. Exogenous information updates the state with information about the new request: time, location, capacity, and revenue.
- *Objective.* Maximize expected revenue.

Value Function Approximation. Study sections 4.1 and 4.2. Describe the authors' general strategy for approximating the value function. What is the difference between a parametric approximation and a non-parametric approximation? What is the motivation to use both?

Application. Study Section 5.3. Describe the parametric and non-parametric value function approximations for the *Capacitated Customer Acceptance Problem with Stochastic Requests*. The illustration below partially depicts the non-parametric approximation.



Does it work? Look at Table 1. Is the policy successful? If yes, how can you tell?

7 Case Study: Direct Lookahead

"A Rollout Algorithm Framework for Heuristic Solutions to Finite-Horizon Stochastic Dynamic Programs," Goodson, Thomas, & Ohlmann. *European Journal of Operational Research*, 2017:258, 216-229.

Problem Description. The objective of the *Dynamic and Stochastic Multi-Compartment Knapsack Problem* (DSMKP) is to pack a knapsack with items presented over a finite time horizon such that total expected reward is maximized subject to constraints on compartmental capacities and overall knapsack capacity. At each decision epoch, the decision maker is presented with a random set of items, any subset of which may be considered for potential inclusion in the knapsack. Although multiple items may be presented simultaneously, we assume at most one item is available to each compartment at each epoch. Compartment capacities, item sizes, and the reward for including a given subset of items in the knapsack are fixed and known. However, prior to the presentation of items, the set of items available at an epoch is known only in distribution.

Model.

- *Decision Epochs.* A decision epoch occurs whenever the decision maker is presented with a new set of items.
- *States.* The state includes remaining compartment capacities, remaining overall capacity, and the subset of items available at the current epoch.
- *Actions.* An action is a choice to accept or reject each available item. Actions are feasible if they respect compartmental and overall capacity constraints.
- *Information.* Whether an item is available to accept or reject for each compartment.
- *Contribution.* A fixed reward for accepting items into each compartment plus a bonus for exceeding some threshold, e.g., a commission plus an incentive to exceed a sales threshold.
- *Transition function.* Action selection updates compartment and overall capacities. New information updates the available items.
- *Objective.* Maximize expected sum of rewards.

Heuristic Policy. The lookahead mechanism is a heuristic policy. The heuristic policy maps a given state to an action via a greedy construction procedure:

- Sort items in descending order of rewards.
- Randomly select a compartment from the first α percent compartments.
- Accept the item if doing so is feasible.
- Update capacities.
- Repeat until no more items can be accepted.

The choice of α determines how greedy versus random the heuristic is.

Rollout Policies. A rollout policy is a type of direct lookahead policy. The rollout policy explicitly builds some portion of the decision tree, then uses the heuristic policy to approximate the remainder. Figure 2 illustrates four ways to do this. Section 3.2 details each rollout policy. Briefly explain how a one-step rollout policy works.

Heuristics and Heuristic Policies. The paper refers to a “heuristic” and a “heuristic policy.” Definition 2 in Section 2 formalizes the concept, but what is the practical distinction as it relates to the DSMKP? There are three parts:

1. The heuristic is a method to identify the policy, e.g., Algorithm 6.
2. The policy is the mapping of state to action, e.g., the output of Algorithm 6.
3. The value of the policy must be calculated or estimated, e.g., the last paragraph of Section 5.2.

Each part has its own computational burden, thus it’s sometimes helpful to separate one from the other.

Does it work? Look at Table 2. Are the rollout policies successful? If yes, how can you tell?

8 Choosing a Policy Class

Use PFAs when you understand the structure of good policies.

Use CFAs when there is a reasonable deterministic approximation that can be optimized and we have an intuitive idea of how to handle uncertainty.

Use VFAs when we need to capture the impact of a decision now on the future, and this value can be expressed as a well-defined function.

Use DLAs when a decision now naturally requires a tentative plan for the future. Also, when other classes fail.

9 Dual Bounds

Motivation. How good is a policy? So far, a policy is good if it is provably optimal or if it performs better than a benchmark. In the latter case, how much better can we do? Without a dual bound, we cannot answer this question.

Dual Bound. A dual bound is some number z such that

Optimality Gap. When we don't know the value of an optimal policy, we rely on the optimality gap to gauge quality. Let $V^\pi (\leq V^\star)$ be the value of our best heuristic policy:

A dual bound can be obtained by relaxing some part of the dynamic program, then optimally solving the resulting problem.

Tractability. In theory, many dual bounds exist, but a dual bound is only helpful if it can be identified, analytically or computationally.

Contribution Function Relaxation. Suppose we relax the contribution function such that

Action Space Relaxation. Suppose we relax the action space such that

Convex DPs. A dynamic program is convex if

Perfect Information Relaxation. Another way to calculate a dual bound is to reveal uncertainty before decisions are made. For example, seeing next week's stock prices before buying, or knowing how disease will spread before distributing vaccines. How does the decision tree change in the perfect information scenario?

Perfect information relaxation intuitively leads to a dual bound. It allows the decision maker to select a policy in response to every outcome instead of a single policy that is evaluated across all outcomes. To make the relationship precise, let $W = (W_1, \ldots, W_{K+1})$ be a random trajectory of random information and let w be a realization of W. Augment the notation for contributions to explicitly show dependence on W. The *expected value with perfect information* (EVPI) is a dual bound on the value of an optimal policy:

A perfect information relaxation is often tractable because

Information Penalties. The EVPI is often a weak dual bound because with perfect foresight we can make much better decisions. To improve the bound, we can punish the decision maker for using information about the future. One way to construct an *information penalty* is via approximate value functions. Let $\hat{V}_k(s_k, w)$ be the approximate value function along trajectory w. For example, $\hat{V}_k(s_k, w)$ might be the value of a heuristic policy across w constructed with any of the methods presented earlier. The penalty for selecting action x_k in state s_k when the trajectory is w is the difference between value functions evaluated across trajectory w and across all possible trajectories:

Incorporating the penalty into the EVPI yields a tighter bound:

Information penalties are attractive when the inner optimization problem, which remains deterministic, is not complicated by the inclusion of the penalty.

Recall the same-day delivery problem with drones and vehicles. How can the threshold policy be used to create an information penalty?

10 Learn More

Modeling and Policy Classes Chapters 9, 11, 12, 13, and 16 of
"Reinforcement Learning and Stochastic Optimization: A Unified Framework for Sequential Decisions" by Warren Powell (2022).

Theory.
"Markov Decision Processes: Discrete Stochastic Dynamic Programming," by Martin Puterman (2005).

"Dynamic Programming and Optimal Control Vol. 1 and Vol. 2," by Dimitri Bertsekas (2017).

Policy Function Approximation Example.
"Same-Day Delivery with Heterogeneous Fleets of Drones and Vehicles," Ulmer & Thomas. *Networks*, 2018:72, 475-505.

Cost Function Approximation Example.
"Binary Driver-Customer Familiarity in Service Routing," Ulmer, Nowak, Mattfeld, & Kaminski. *European Journal of Operational Research*, 2020:286, 477-493.

Value Function Approximation Example.
"Meso-Parametric Value Function Approximation for Dynamic Customer Acceptances in Delivery Routing," Ulmer & Thomas. *European Journal of Operational Research*, 2020:285, 183-195.

Direct Lookahead Example.
"A Rollout Algorithm Framework for Heuristic Solutions to Finite-Horizon Stochastic Dynamic Programs," Goodson, Thomas, & Ohlmann. *European Journal of Operational Research*, 2017:258, 216-229.

Information Relaxations.
"Information Relaxations and Duality in Stochastic Dynamic Programs: A Review and Tutorial," Brown & Smith. *Foundationas and Trends in Optimization*, 2022:5:3, 246-339.

Action Space Relaxation.
"Relaxations of Weakly Coupled Stochastic Dynamic Programs," Adelman & Merserau. *Operations Research*, 2008:56(3), 712-727.