

A Markov Decision Problem Approach for the Deployment and the Maintenance of a Satellite Constellation

Frédéric Garcia¹, Laurent Peret¹, Linda Tomasini², Gérard Verfaillie³

¹ INRA, BIA, BP 27, 31326 Castanet-Tolosan, France
{fgarcia,peret}@toulouse.inra.fr

² CNES, 18 avenue Édouard Belin, 31401 Toulouse Cedex 4, France
Linda.Tomasini@cnes.fr

³ ONERA, DCSD, 2 av. Edouard Belin, BP 4025, 31055 Toulouse Cedex 4, France
verfaillie@cert.fr

Keywords AI for space applications, planning and scheduling, uncertainty, optimization, modeling and simulation, machine learning

Abstract

This paper is dealing with finding policies for the deployment and the maintenance of satellite constellations that would be cost-optimal and robust to failures. The solution we propose consists in modeling the deployment and maintenance problem as a Markov decision problem (MDP), where launching plans over a given decision horizon are included in the description of the current state. The preliminary results we present in this paper concern the optimization of the maintenance phase, assuming that the deployment and end of service policies are already defined. Two distinct optimization algorithms are compared: (i) approximate policy iteration, where the policy evaluation is done through simulations, and (ii) stochastic optimization, where a parameterized expert policy is directly stochastically optimized through simulations. Our first results confirm the complexity of the optimization problem, and seem to establish the superiority of the stochastic optimization method in finding near optimal maintenance policies.

1 Introduction

Satellite constellations (GLOBALSTAR, TELEDESIC, GALILEO...) are large, complex, and expensive space programs that lead engineers

to face new design and planning problems. Due to the use of some of the most recent technologies, failures occurring during either the deployment or the operational service phases are not rare events. Due to the strong commercial competition conditions, delays in operational service and partial unavailability periods are very costly.

This paper is dealing with finding strategies for the deployment and the maintenance of satellite constellations that would be cost-optimal and robust to failures. The reference case we are studying is a constellation involving 32 operational satellites, evenly distributed over 8 orbital planes. In order to deploy or to replace these satellites, different types of launcher can be used, with different features: capacity, cost, reliability, minimum time between successive launches ... Different types of failures may appear during the deployment or the operational service phases: launcher failure, satellite failure, satellite end of life ... Since the preparation and the launch of satellites is a long process that needs to be planned, the maintenance of the constellation uses redundant spare satellites. These satellites are kept on parking orbits that drift regularly relatively to the orbital planes. As for operational satellites, they are subject to failures.

2 A Markov Decision Problem Model

The deployment and the maintenance of a satellite constellation can be seen as a *sequential decision problem* under *uncertainty*, but *complete observabil-*

ity. Indeed:

- decisions have to be made in sequence about the preparation and the launch of new satellites, and about the operational use of spares;
- the consequences of a given decision are not known with certainty, because of the failures that may occur;
- it can be however assumed that the current state of the system at the decision time is known with certainty.

Thus, the formal *MDP* framework (*Markov Decision Process* [5]) can be used to model it. In a few words, a *MDP* is defined by a quintuple $\langle S, A, T, P, R \rangle$ where:

- S is the set of all the possible *states* of the system;
- A is the set of all the *actions* that can be applied to it;
- T is the ordered set of *instants* at which decisions can be made, that is the *global temporal horizon*;
- P defines the *transition probabilities* between any pair of states in S after executing an action in A ;
- R defines the *local costs* or *rewards* associated with these transitions.

In this framework, the usual request is to compute what is called an *optimal policy*, that is a function π^* that associates with any *state* $s \in S$ and any *time* $t \in T$ an *optimal action* $\pi^*(s, t)$, that is an action that minimizes the *expected global cost* (or maximizes the *expected global reward*) on the remaining temporal horizon. This global cost or reward may be defined as the *sum*, the *discounted sum*, or the *mean value* of the local costs or rewards associated with the actual transitions. When the optimal policy does not depend on the time t and only depends on the state s , it is said to be *stationary*.

For our problem:

- S is defined by a set of *state variables* and their associated *domains*; the chosen state variables are the *number* of operational satellites and spares on each orbital plane, and the *age* of each of these satellites;

- the same way, A is defined by a set of *decision variables* and their associated *domains*; the chosen decision variables are the *launching decision* (to launch or not to launch and, in case of launch, the launch type), the associated *orbital plane*, and the *number* of spares to be sent on the operational orbit on each orbital plane; moreover, constraints between state and decision variables allow us to specify what decisions cannot be applied to a given state;
- T is *discretized* with one month (the time taken by a spare to drift from one orbital plane to the next) between two successive decision instants, and is considered as *infinite*;
- P is defined by a set of *random variables* associated with the possible failures and by *conditional probabilities* between state variables associated with a given instant, decision variables associated with the same instant, random variables associated with the transition following the decision, and state variables associated with the next instant;
- R defines the *local costs* associated with *states* (either partial or complete unavailability of the constellation) and *decisions* (preparation and launch costs);
- the *global cost* is defined as the *discounted sum* of the local costs, and a *stationary policy* that minimizes the expected global cost is looked for.

But, whereas some of the decisions are *immediately executable* (launch orbital plane in case of launch, spares to be sent on the operational orbit), others are not (satellite preparation date and number, launch date and type), because they need time to be prepared. To overcome this difficulty, we consider that a satellite preparation and launching plan must be *committed* at each instant over a given *decision horizon* of some months and that this committed plan takes part in the current state.

Thus, the state variables are divided into two classes: the ones associated with the current *state of the constellation* (number of operational satellites and spares on each orbital plane, age of each of these satellites) and the ones associated with the current *committed plan* (satellite preparation date and number, launch date and type over the decision horizon).

The same way, the decision variables are divided into two classes: the *immediately executable* decisions (launch orbital plane in case of launch, spares to be sent on the operational orbit) and the decisions *with*

delayed execution (extension of the satellite preparation and launching committed plan).

3 Simulation and Optimization for MDPs

The MDP theory associates with each *policy* π a *value function* V_π , that associates itself with each state $s \in S$ and each time $t \in T$ the expected global cost or reward $V_\pi(s, t)$, obtained by applying π from s and t . *Bellman's optimality equations* (equations 1 [2]) characterize in a compact way the *unique optimal value function* V^* from which an *optimal policy* π^* can be straightforwardly derived (equations 2). In case of a stationary policy and a global cost defined as the discounted sum of the local costs, these equations are the following :

$$V^*(s) = \min_{a \in A} [R(s, a) + \gamma \cdot \sum_{s' \in S} P(s'|s, a) \cdot V^*(s')] \quad (1)$$

$$\pi^*(s) = \operatorname{argmin}_{a \in A} [R(s, a) + \gamma \cdot \sum_{s' \in S} P(s'|s, a) \cdot V^*(s')] \quad (2)$$

For MDPs with small finite S and A sets, *value iteration* or *policy iteration* are *dynamic programming* algorithms that exploit efficiently the problem data and structure, in order to compute this optimal value function and to derive an associated optimal policy.

Unfortunately, the constellation deployment and maintenance problem does not belong to this category. The size of the state and action spaces is huge, and dynamic programming algorithms cannot be directly used. In that paper we consider two different optimization approaches, both based on simulation: *reinforcement learning* and *stochastic optimization*.

The *Artificial Intelligence* community has recently developed *reinforcement learning* methods that make possible the optimization of policies for large MDPs, by means of simulations and approximations of the value function and/or of the policy. The principle of reinforcement learning is to update locally, on the basis of simulated trajectories of the MDP, a set of parameters that approximate the optimal value function.

Another alternative approach for solving large MDPs consists in using an *a priori* parameterized policy, defined by experts, and in optimizing the values of

these parameters through the use of simulated trajectories. Note that the Markovian feature of the problem is no more exploited in this approach and that the whole process is considered as a *black box* that transforms input parameters (the policy parameters) into an output criterion (the expected global cost or reward).

3.1 Reinforcement Learning

The reinforcement learning approach consists in learning an optimal policy by estimating iteratively the optimal value function of the problem on the basis of simulations. Today, reinforcement learning is one of the main approaches able to solve sequential decision problems with unknown transition probabilities and/or MDPs with large state spaces. Various reinforcement learning algorithms exist, like *Q-learning*, *R-learning*, *Sarsa*, and others [7].

In this paper we consider the *approximate policy iteration* algorithm [3], that can be used when the transition probabilities of the MDP are known, as it is the case in our problem.

This algorithm is based on two phases that are iteratively repeated. At each iteration k of the algorithm, (i) the value function V_{π_k} , associated with the current policy π_k , is estimated, and (ii) a new policy π_{k+1} is derived from V_{π_k} , using equations 3.

$$\pi_{k+1}(s) = \operatorname{argmin}_{a \in A} [R(s, a) + \gamma \cdot \sum_{s' \in S} P(s'|s, a) \cdot V_{\pi_k}(s')] \quad (3)$$

The value function, associated with a policy π is defined by equations 4.

$$V_\pi(s) = E[\sum_{t=0}^{+\infty} \gamma^t \cdot R(s_t, \pi(s_t)) | s_0 = s] \quad (4)$$

But, for large MDPs, this computation cannot be done exactly. The value function V_π is thus approximated by a function V_w , where w is a set of parameters. The first phase of each iteration of the algorithm consists now in estimating the parameters w_k such that V_{w_k} approximates the best V_{π_k} . This is done through simulations, using n trajectories of length l , starting from various initial states and following the policy π_k . If s_{0_i} is the initial state of the i^{th} trajectory, w_k is defined by equations 6.

$$w_k = \underset{w}{\operatorname{argmin}} \quad (5)$$

$$\sum_{i=1}^n [V_w(s_{0_i}) - \sum_{t=0}^{l-1} \gamma^t \cdot R(s_t, \pi(s_t)) | s_0 = s_{0_i}]^2$$

Either on-line methods like $TD(\lambda)$, or off-line methods like *Monte-Carlo* algorithms have been proposed to compute w_k [3].

This algorithm defines a sequence of parameters w_k , that is expected to converge towards \hat{w} , such that the associated policy $\hat{\pi}$ is a near optimal policy.

The quality of its result widely depends on the quality of the approximation V_w . In this paper, we consider a simple *linear approximation* $V_w(s) = \sum_{j=1}^m w^j \cdot \phi_j(s)$, where w^j is the j^{th} component of w and where $\phi_1(s), \dots, \phi_m(s)$ are real-valued features that characterize the state of the system.

3.2 Stochastic Optimization

Stochastic optimization methods can be used for optimizing a stochastic real-valued function J of a vector θ of p real parameters. If $\Theta \subseteq R^p$ is the domain of the parameter vector θ , the problem is to find a value $\theta^* \in \Theta$ that minimizes the expected value of J :

$$\theta^* = \underset{\Theta}{\operatorname{argmin}} E(J(\theta)). \quad (6)$$

Various methodologies [1] have been proposed in order to solve this problem. In this paper, we consider a *gradient based procedure*, due to [6] and [4], that uses noisy observations of the function J . A projected version of this iterative algorithm is defined by equations 7:

$$\theta_{n+1} = \pi(\theta_n + \alpha_n \frac{1}{2\beta_n} (J(\theta_n + \beta_n) - J(\theta_n - \beta_n))) \quad (7)$$

where, for all $\theta \in R^p$, $\pi(\theta)$ is the point in Θ that is the closest to θ , and where α_n and β_n are positive real numbers that satisfy the following conditions:

$$\sum_1^\infty \alpha_n = \infty, \quad \lim_{n \rightarrow \infty} \beta_n = 0, \quad \sum_1^\infty \alpha_n \beta_n < \infty, \quad \sum_1^\infty \frac{\alpha_n^2}{\beta_n^2} < \infty$$

In our problem, the stochastic function J to optimize is the global cost from the initial state of the system.

The starting point of the algorithm can be given by experts or randomly defined. At each iteration of the algorithm, the function J is estimated through simulations. The algorithm stops when a specified criterion is satisfied (e.g. a maximum number of iterations).

4 Experiments

4.1 Reference Study Case

The preliminary results we present in this paper concern the optimization of the maintenance phase, assuming that the deployment and end of service policies of the constellation are already defined.

We consider a constellation involving 32 operational satellites, evenly distributed over 8 orbital planes. Three types of launchers are available with a payload of 2, 4, and 6 satellites. We assume that the probabilities of launcher failure and satellite failure are known with certainty : the satellite monthly failure probability is assumed to be constant and equal to 0,003 ; the launch failure probabilities are assumed to be respectively equal to 0.10 , 0.08 and 0.05 for the 2, 4, and 6 satellite launchers. The life of each satellite is assumed to be 6 year long. A satellite costs 1 unit ; the 2, 4, and 6 satellite launchers cost respectively 1, 1.5, and 2 units. Degradation costs 1 unit per satellite out of order and per month. In the initial state of the maintenance phase, all the 32 satellites are assumed to be operational and 0 year old. In practice we simulate trajectories over a 10 year horizon.

4.2 Reinforcement Learning

Because of the large number of combinations of decisions to consider at each instant, we introduce an arbitrary *order between decisions* : first extension of the satellite preparation and launching plan, then launch orbital plane in case of launch, and finally spares to be sent to the operational orbit. Each instant is thus artificially decomposed into 3 instants and we introduce 3 value functions, 3 simplified optimality equations and 3 sub-policies. This decomposition of the problem is illustrated on figure 1.

Despite of this simplification of the decision space, the problem is still too large because of the huge size of the state space. We are unable to store the 3 value functions in a lookup-table. Thus we define a linear approximation of each value function based on one

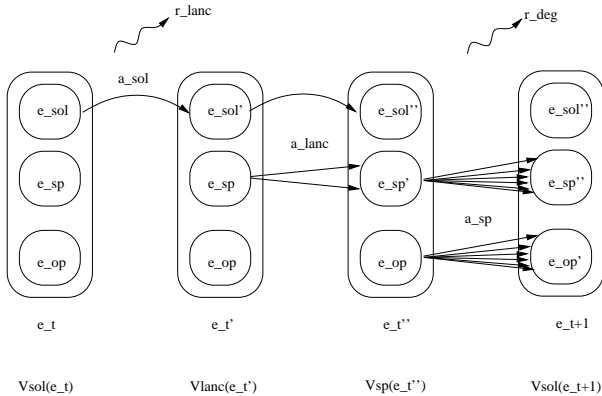


Figure 1. Sequential decomposition of the decision problem in 3 steps

hundred features ϕ_i of the state : basic features like the number and the ages of the operational and spare satellites on each orbital plane, synthetic features like the average age of the operational satellites on each orbital plane.

The first phase of the first iteration of the algorithm consists in estimating the value function associated with the starting expert policy π_0 (see section 4.3), by using a set of trajectories obtained by following π_0 and by minimizing the quadratic error between the observed trajectory costs and the parameterized value function (see equations 6).

The on-line $TD - \lambda$ algorithm we used to update the weights w_i incrementally after each transition of each trajectory seems to converge towards a (local) minimum of the quadratic error after a few iterations. To validate these results, we implemented a second algorithm of the *Monte-Carlo* family, that is an off-line algorithm. This algorithm is computationally more expensive than the previous, but the quadratic error we obtain is about 20 % smaller (see figure 2).

The following iterations of the algorithm consists in trying to improve on this starting expert policy. Until now, these iterations did not provide satisfying results : the maintenance policies we got are of poor quality. However, when we use an expert policy for the first 2 sub-policies (extension of the satellite preparation and launching plan, launch orbital plane in case of launch), the learned third sub-policy (spares to be sent to the operational orbit) performs as the same level than an optimized expert policy (see section 4.3).

A possible explanation to the relative inefficiency of the approximate policy iteration might be the poor quality of the linear approximation of the value function : the numerical imprecision of our value func-

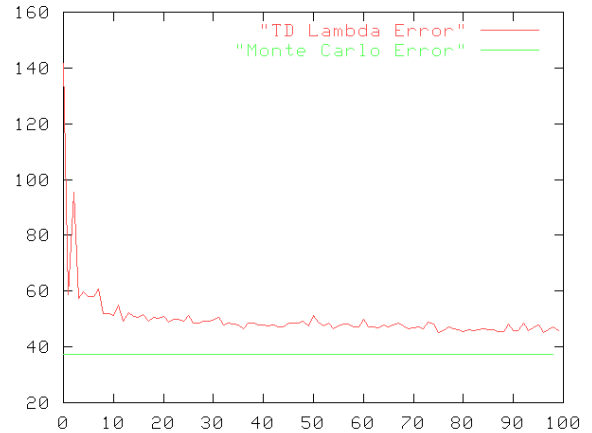


Figure 2. Evolution of the quadratic error with the TD- λ algorithm - Quadratic error obtained by the Monte-Carlo algorithm

tion, that does not allow the algorithm to make the appropriate decisions for the extension of the satellite preparation and launching plan, and for the choice of the launch orbital plane in case of launch. Because of the γ discounting factor, the precision of the approximation is indeed critical when decisions have long-term benefits (by saving degradation costs) but immediate costs (like satellite preparation and launch costs). We are currently considering *neural network* architectures for getting better approximations.

4.3 Stochastic Optimization

The policy we try to optimize is based on a set of expert parameterized or non-parameterized decision rules. One parameterized rule is defined for each of the three steps of the decision process (see figure 1). These rules are parameterized via 7 real-valued parameters. For instance, the rule for the replacement of operational satellites involves a parameter that is a maximum age, beyond which a satellite must be replaced if there is an available spare on the same orbital plane.

The *Kiefer-Wolfowitz* algorithm is stopped after a specified number of iterations. Providing an optimized parameter vector takes a few hours. We have no guarantee related to the quality of the result, but the convergence of the algorithm does not seem to depend on the starting point of the algorithm (the maximum differences between the values obtained for the final vector equal 2%). We noticed that, for a specified number of simulations, there is a trade-off to make between the number of simulations used at each iteration in order to estimate the function J

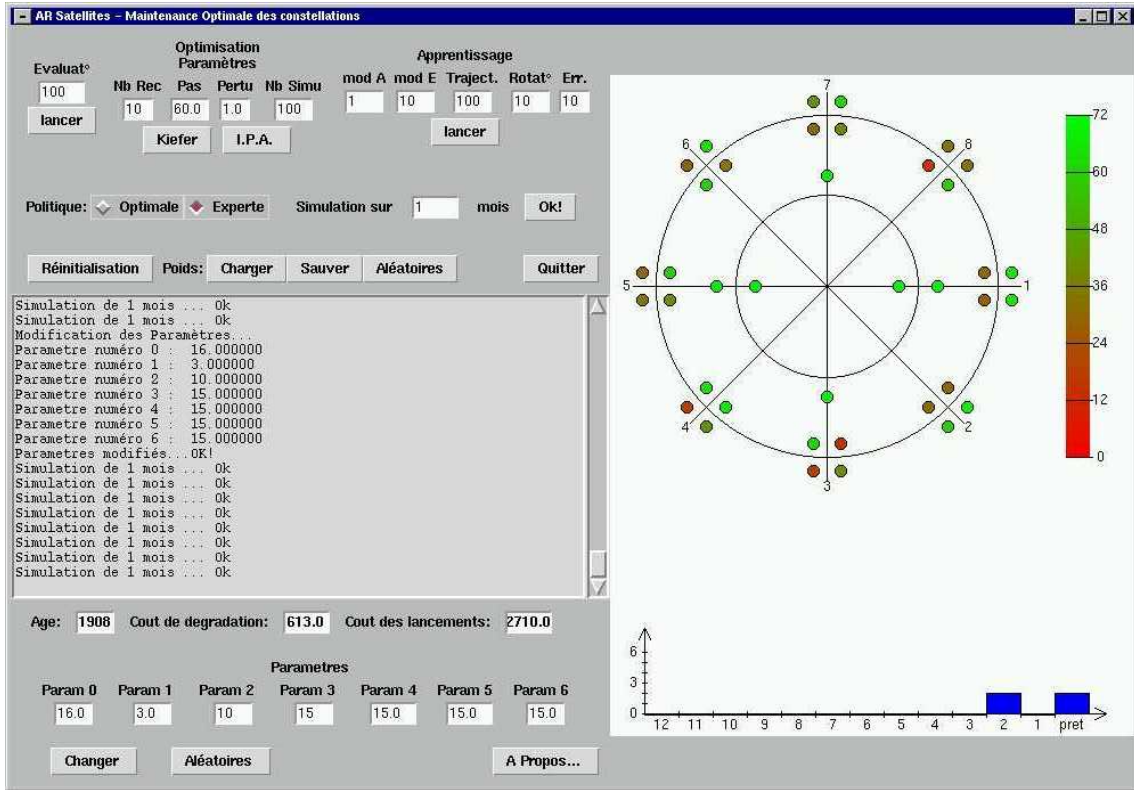


Figure 3. Satellite constellation markovian simulator

and the total number of iterations. Indeed, if we use more simulations at each iteration, the estimations of the function J will be more precise, the quality of each update of the parameter vector θ will be better, but the number of these updates will be smaller.

We succeeded to improve the initial policy (parameterized with a set of experimentally good, but non optimal values), decreasing the mean cost of about 23 %. We also reduced the variance of the cost distribution, decreasing the probability of high-cost trajectories.

4.4 Synthesis of the Experimental Results

Figures 4, 5, 6 present a synthesis of the results we obtained. They show the distribution cost respectively associated with the policy obtained by the reinforcement learning approach, the starting expert policy of the stochastic optimization approach, and the policy obtained by this latter approach. Figure 4 shows the poor quality of the policy resulting from the reinforcement learning approach. Figures 5 and 6 show the improvement of the starting expert policy by the stochastic optimization approach.

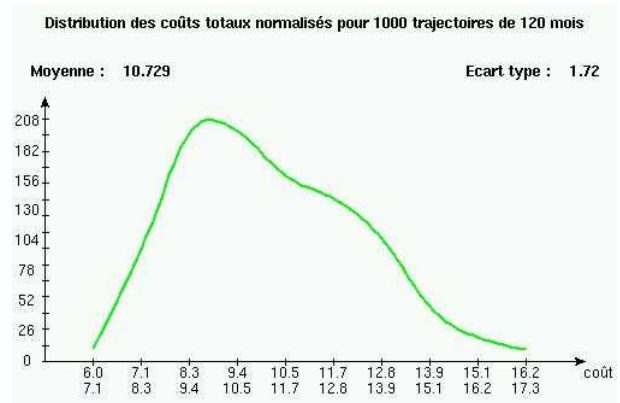


Figure 4. Cost distribution of the policy obtained by reinforcement learning

5 Conclusion

Until now, the *stochastic optimization* approach provided us with much better results than the *reinforcement learning* approach. However, the first approach requires the definition of an expert policy structure and the setting of initial values for the parameters, that the design engineers may have some difficulties to obtain.

If the results of the reinforcement learning approach

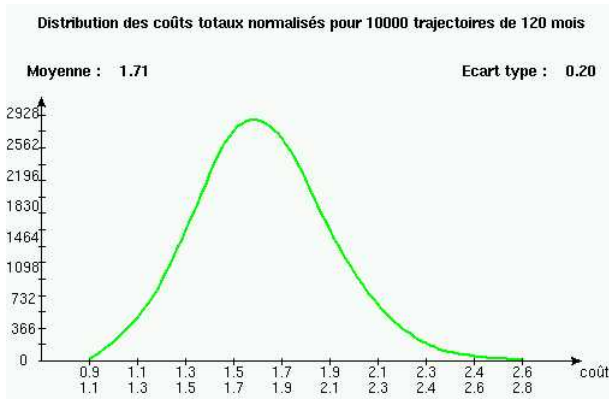


Figure 5. Cost distribution of the initial expert policy

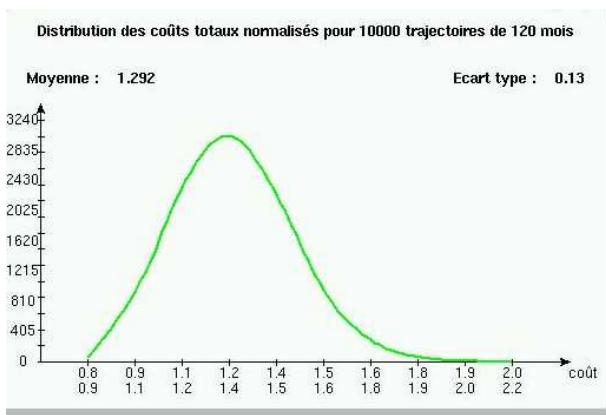


Figure 6. Cost distribution of the optimized expert policy

are quite disappointing, we hope to improve on them by different ways. For example, a *neural network* approximation of the value function would be certainly better than the linear approximation we used. A combination between *dynamic programming* and *branch and bound* algorithms for deriving a policy from a value function would deserve also more attention.

References

- [1] F. Azavidar. Simulation optimization methodologies. In *Proc. of the 1999 Winter Simulation Conference, USA.*, 1999.
- [2] Richard E. Bellman. *Dynamic Programming*. Princeton University Press, 1957.
- [3] D. P. Bertsekas and J. N. Tsitsiklis. *Neuro-Dynamic Programming*. Athena Scientific, Belmont (MA), 1996.

- [4] J. Kiefer and J. Wolfowitz. Stochastic estimation of the maximum of a regression function. *Annals of Mathematical Statistics*, 23:463–446, 1952.
- [5] M. L. Puterman. *Markov decision processes: discrete stochastic dynamic programming*. Wiley-Interscience, New York, 1994.
- [6] H. Robbins and S. Monro. A stochastic approximation method. *Annals of Mathematical Statistics*, 22:400–407, 1951.
- [7] R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, Cambridge, Massachusetts, 1998.