Minimizing Thermal Impacts of Hydropower using Reinforcement Learning

Isabel Bush
ibush@stanford.edu

Matthew Shultz
shultzm@stanford.edu

Introduction

Operational policy for hydropower dams must balance energy generation goals with environmental impacts under federal regulation, such as the Endangered Species and the Clean Water Acts. Multiple impoundments, reservoirs, outlet structures, and stochastic weather patterns complicate decision making to minimize impacts on downstream ecosystems. Currently, hydropower management to meet water quality goals is largely based on seasonal rules-of-thumb. Non-optimal management combined with recent, climatically novel heat waves has led to high die-off rates for fish, who cannot tolerate high water temperatures. For example, high temperatures in spring and summer of 2015 caused a die-off of over 95% of a sockeye salmon cohort returning to the Snake River in the state of Washington[7].

Water undergoes significant increases in density as its temperature drops. In a reservoir, this property can create stratification, sequestering cool water low in the vertical profile while warmer water sits at the top. Dams often have two or more outlets from top to bottom through which water can be released, with the lower outlets passing cooler water and the upper outlets passing warmer water. However, impounded waters heat over the course of the year, undergoing complex nonlinear meteorological and hydrological forcing. Thus, maintaining available volumes of water at appropriate temperature for release during the warmest months requires nontrivial planning.

We demonstrate the applicability of Q-learning to optimize policy for dam operations. We computationally simulate the hydrodynamics of a hydropower dam, and apply Q-learning to identify optimal actions day by day. We report here on two separate learning objectives that optimize two subgoals of dam operations: keeping water elevation levels within operational limits, and minimizing the temperature increase across the reservoir.

Related Work

Computational hydrodynamic simulations are common assessment tools for water management policies, but models are used much less frequently for policy optimization. Theoretical work for water supply optimization has focused on stochastic dynamic programming, in which a Markov decision process is solved backwards[9]. However, this approach requires an explicit state transition model and suffers from the curse of dimensionality with complexity increasing exponentially with state. As an alternative, some work has leveraged Q-learning to optimize distribution from storage reservoirs to different user classes[2][3] and to optimize multiple and sometimes conflicting water management objectives[10][4]. Novel approaches to managing the continuous state space used in these works include discretization using percentile values and k-means clustering[10], tree-based regressors[3], and neural network function approximations[1]. While other recent work has leveraged machine learning to predict water quality chemical concentrations[18][8][12], attention has not focused on optimizing controls for civil infrastructure that affect these variables. Specifically, we are unaware of previous Q-learning for optimizing control of dams to achieve water quality goals.

Simulation

We leverage CE-QUAL-W2, a two dimensional (longitudinal-vertical) FORTRAN hydrodynamical simulation code[5], to model water flow and thermal advection within the physical problem domain (Figure 1). CE-QUAL-W2 requires bathymetry and hydrodynamics parameters as well as hydrological and meteorological time-series to compute outflow temperatures. We used bathymetry and hydrodynamic parameters from an EPA thermal effects study of the Lower Granite Dam on the Snake River[6]. To demonstrate advanced temperature control, we segmented the single lower outlet into two outlets at different elevations. Required simulation input data retrieved for years 2005-2015 includes meteorology, upstream flows, and upstream temperatures. Meteorological data was retrieved from the National Weather Service Quality Controlled Local Climatological Data[13]. Upstream flows and temperatures were retrieved from USGS[15].

We started the simulation at a standardized depth on day 90, and allowed Q-learning to control dam operations through day 215. This captures the time of the year in which migrating fish are most thermally sensitive, and allows enough lead-in for control decisions to have impact. State variables were measured and actions chosen on a daily basis, the typical granularity of control managed in a real dam.

To extend training data for generalization, we simulated 500 inflow datasets by fitting normal distributions for in-

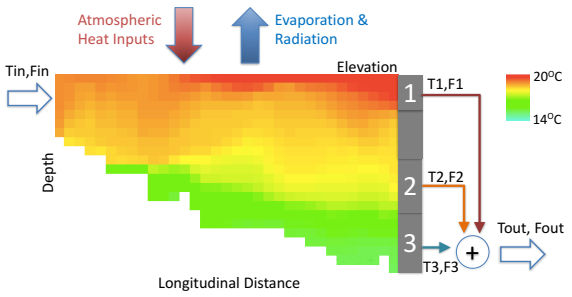

Figure 1: Schematic of CE-QUAL-W2 hydrodynamics model with simulated water temperatures for May 30th, 2015. Positions of three dam outlets are pictured. T1-3 and F1-3 are temperatures and flow rates at the output gates, Tin and Fin are temperature and flow at the inflow, and Tout and Fout are the total temperature and flow downstream of the dam.

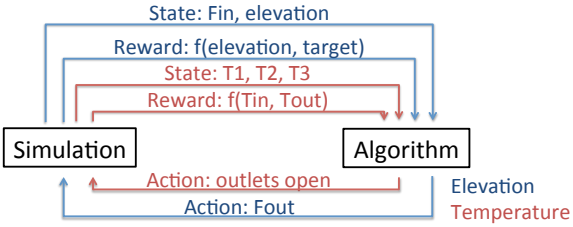

Figure 2: System model showing state, rewards, and actions for both elevation and temperature optimization objectives. All state variables are as defined in Figure 1, and target is a target elevation of 220m.

flows observed on each year-day, randomly selecting daily flows from these distributions, and applying a five day smoother. We created simulated advection inputs as a combinatoric set of all inflow temperatures and flow rates from 2010-2015.

System Model

The learning algorithm uses state drawn from the hydrodynamics simulation and rewards calculated from that state to choose actions, which are passed back to the simulation (Figure 2).

For controlling water surface elevation, we provided the algorithm with the input flow rate and the current elevation as state. Rewards were calculated as a quadratic function of the distance from the target elevation, with an additional high negative reward for elevations beyond a desired range, a fail state that restarted the model (Equation 1). There were 24 discretized potential outflow actions which spanned the range of potential inflows observed at the site. In a particular state, we limited the allowed actions to the $k_{actions}$ nearest to the state's inflow amount to reduce the substantial state-action space. Since the full set of outflow actions contains many that, for a given inflow, can drain or overflow the simulated reservoir, limiting this action space *a priori* is reasonable without loss of learning generality.

$$reward = \begin{cases} 16 - (elev - target)^2 & \text{if } (elev - target)^2 < 25 \\ -100 & \text{otherwise} \end{cases} \quad (1)$$

The water temperature optimization algorithm used center temperatures of the three dam outlets as state, and calculated the reward as the difference between the inflow and outflow temperatures for the day, with an offset to enhance positive rewards (Equation 2). Available actions, totaling five, included release of all inflow through one of the three outlets or to split the inflow evenly between the middle and either top or bottom outlets.

$$reward = Tin - Tout + 1 \quad (2)$$

For algorithms requiring discretization of the state space, each state variable was discretized into evenly-spaced levels spanning the expected range. There were eight levels spaced 500cfs apart for inflow, 19 levels spaced one-meter apart for elevation, and 22 levels spaced 0.5°C apart for temperature. State-action features $\phi(s, a)$ are described in Equation 3 and 4, where v_i is the i^{th} variable of the state and v_{ij} is the j^{th} discretization level for v_i.

$$I(v_i, v_{ij}, v_{i(j+1)}) = \mathbb{1}\{v_{ij} \leq v_i \leq v_{i(j+1)}\} \quad (3)$$

$$\phi(s) = \begin{bmatrix} I(v_1, v_{11}, v_{12}) \\ I(v_1, v_{12}, v_{13}) \\ \vdots \\ I(v_1, v_{1m-1}, v_{1m}) \\ \vdots \\ I(v_n, v_{nm-1}, v_{nm}) \end{bmatrix} \quad \phi(s, a) = \begin{bmatrix} \phi(s)\mathbb{1}\{a = a_1\} \\ \phi(s)\mathbb{1}\{a = a_2\} \\ \vdots \\ \phi(s)\mathbb{1}\{a = a_k\} \end{bmatrix} \quad (4)$$

Learning Algorithm

Q-learning is a model-free form of reinforcement learning, which optimizes a policy without explicitly defining the transition probabilities between all states.

In each step of the Q-learning algorithm, an action a is taken at current state s, and this results in a reward r and a transition to a new state s'. The Q-value $Q(s, a)$ is updated to reflect the new estimate of the expected utility, or the discounted sum of future rewards, of taking action a at state s and following the optimal policy from the resulting state. Thus the Q-value update is given by Equation 5, where α and γ are the learning rate and discount factor, respectively[16].

$$Q(s, a) \leftarrow Q(s, a) + \alpha(r + \gamma \max_{a' \in A} Q(s', a') - Q(s, a)) \quad (5)$$

The learned policy is to choose the action a that maximizes $Q(s, a)$ for current state s. However, since Q-learning is an on-line learning algorithm, it must explore the state-space in addition to following the current optimal policy. We use an epsilon-greedy approach in which a random action is taken with probability ϵ while the current best action is taken with probability $(1 - \epsilon)$.

The estimation of the Q-value may be determined in many ways. The simplest is to store $Q(s, a)$ in a Lookup table and update it according to Equation 5. However, this requires discretization of the state and action spaces and does not generalize at all to unseen state-action pairs, thus random actions will be taken during testing from any states not visited during training.

To handle the continuous state space and improve generalization, we leveraged a k-Nearest Neighbors (kNN) scheme for calculating Q-values developed by Martin *et al*. [11]. In this approach, we store Q-values for points spanning the state space and update the Q-values of the kNN to the current state at each step. To ensure that the kNN are not biased to smaller dimensions, all state variables are scaled to be in the range [-1, 1].

To determine state Q-values and update amounts for the stored points, weights are assigned to the kNN with an inverse relationship to their Euclidean distance d to the current state (Equation 6).

$$\begin{aligned} w_i &= \frac{1}{1 + d_i^2} \\ p(i) &= \frac{w_i}{\sum w_i} \quad \forall i \in knn \end{aligned} \tag{6}$$

Using these weights, the expected utility of an action a in state s is given as a weighted sum of the kNN Q-values (Equation 7).

$$Q(s, a) = \sum_{i=1}^{knn} Q(i, a)p(i) \tag{7}$$

Once the reward r and next state s' are revealed for taking action a at state s, this information is incorporated into the Q-values for the kNN to s, with closer neighbors receiving a greater update (Equation 8).

$$Q(i, a) \leftarrow Q(i, a) + \alpha(r + \gamma \max_{a' \in A} Q(s', a') - Q(s, a))p(i) \quad \forall i \in knn \tag{8}$$

The kNN approach generalizes to unseen states that share at least one neighbor with a visited state.

We also applied a linear function approximation to estimate the Q-value[14]. State is again discretized as in Equation 4. The Q-function approximation and weight update are given in Equations 9 and 10. This generalizes to unseen state-action pairs that share both the action and at least one discretized state variable with observed state-action pairs.

$$Q(s, a; \theta) = \theta \cdot \phi(s, a) \tag{9}$$

$$\theta \leftarrow \theta - \alpha[Q(s, a; \theta) - (r + \gamma \max_{a' \in A} Q(s', a'; \theta))]\phi(s, a) \tag{10}$$

Experimentation and Results

We trained and tested all three Q-learning algorithms using our simulation setup for optimization of both the elevation and temperature objectives. For each algorithm, we trained the elevation objective for approximately 1500 simulation runs and the temperature objective for approximately 200 simulation runs. A major challenge for learning is the considerable computation cost of running the hydrodynamics models, with a run-time of approximately five minutes for a single year (125 simulated days) of input data. Average simulation run-times for the elevation objective were much shorter because this training included a fail state that triggered restart.

Hyperparameters

For both objectives, we were able to achieve reasonable results with a discount factor for future rewards (γ) of 0.75, exploration parameter (ϵ) of zero, and learning rate (α) of 0.1. We reduced γ from 0.95 because the algorithm was not learning very quickly, and we decided to focus on near term optimization. Constant non-zero values for ϵ above 0.5 led to well explored state very early in the year but quickly declined to poor performance. When ϵ was low, learning was slow but consistent. We attempted to anneal ϵ, which produced somewhat better results than constant ϵ, but still had poor performance later in the year. To more fully explore the entire yearly time series, we experimented with dynamically annealing ϵ from one to zero such that epsilon annealed more quickly early in the year than later. Ultimately we report here on learning with ϵ equal to zero since it produced clear results within the computational time available.

For the k-Nearest Neighbors algorithm we selected hyperparameters of five nearest neighbors, and twenty points per state dimension. These values maximized resolution while keeping average time per model step to under two seconds.

For the elevation objective, we experimented with varying the number of allowed actions from each state. With too few available actions, it is not possible to meet the objective, while with too many, learning is very slow since the algorithm is exploring state-action pairs that are unreasonable. We experimented with values of five, eight, and twelve for $k_{actions}$, and generally found gradually increasing the allowed actions produced improved results by slowly expanding the action space for problematic states.

Metrics

Elevation performance was evaluated as the average number of days in each testing run until elevation moved outside permitted boundaries (5m from the target elevation). Elevation control skill was also evaluated as the mean absolute error from the target elevation. Outflow temperature optimization was evaluated as the number of test period days in which outflow temperature was more than two degrees above inflow temperature. This is a metric developed by the EPA for the Lower Granite Dam[17].

For both elevation and temperature objectives we compared metrics against a theoretical random policy, producing an average metric of results from 100 simulated years run with actions chosen completely at random. For temperature visualization, we also compared learned performance to a static policy in which all flow is split between the dam's top two gates. A random policy would never be practically implemented on a hydropower dam, while a static policy approximates one feasible management strategy.

Elevation Results

Initially, the elevation objective was trained for a single year (2007) to observe learning (Figure 3). All three algorithms learned to maintain elevation within the boundaries for longer as training progressed, however there was some negative progress as new areas of the state-space were explored. While the Linear algorithm converged to a successful policy more quickly than the Lookup and kNN algorithms, the latter algorithms were able to reduce their elevation mean absolute error below that which was achieved by the converged Linear policy. In Figure 4 learned policies maintain elevations within the desired boundaries for training data while a random policy falls outside. Lookup and kNN policies track closer than the Linear policy to the target elevation of 220m.

After subsequent training on simulated inflow data, generalization was assessed on ten years of unseen real inflows (Table 1). All three algorithms were able to stay within the elevation boundaries for longer than a random policy, and the Lookup and kNN algorithms also decreased the mean distance from the target elevation.

Learning was quite sensitive to reward function design, and our final reward function was the result of heuristic experimentation. In general, negative rewards were necessary to mark states that should not be explored further and speed up learning without requiring high ϵ values. Using only negative rewards, however, did not guide the algorithm to further explore potential optimal paths near the target elevation.

Temperature Results

Initial learning on only a single year (2015) shows all three algorithms rapidly improving from a random policy, with kNN converging quickly while the Linear and Lookup algorithms still oscillate after more than 80 passes over the training data (Figure 5). The first policy at the start of learning is very poor since 2015 was the hottest year on record.

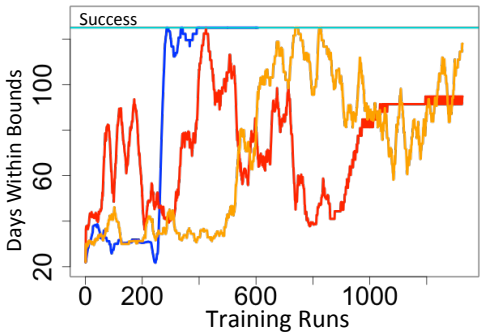

Figure 3: Learning curves showing a running average (over 25 simulation runs) of the number of days elevation was maintained within the boundaries for each algorithm. Intersection with the horizontal blue line indicates that all 25 runs in the running average stayed within elevation boundaries for the full simulation period. Running average was reported because number of days per run oscillates widely until convergence.

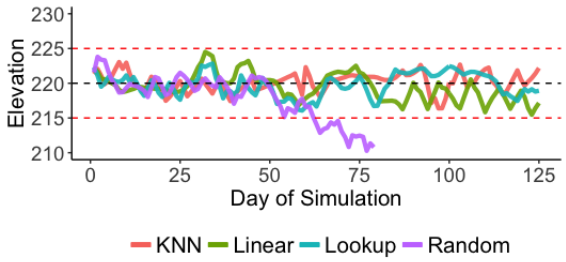

Figure 4: Comparison of elevation performance during training for the year of 2007 vs. control by a random policy. Algorithms were trained to maintain elevation as near as possible to 220m without deviating more than 5m.

Subsequent training on randomized data produced good average results on the training set for kNN and Linear algorithms, but only a small improvement over a random policy for the Lookup algorithm. kNN and Linear algorithms also generalized well, with nearly equivalent testing and training metrics, while Lookup performed nearly as poorly as a random policy under unseen data (Table 2).

Figure 6 compares results from learned policies to the static policy in 2007. The Lookup algorithm displays sudden temperature spikes, a signature of random guessing.

Table 1: Elevation generalization results to unseen data.

	Mean days within bounds	Mean absolute error
Random	17.3	2.37
Lookup	34.5	1.71
kNN	34.1	1.37
Linear	31.6	2.29

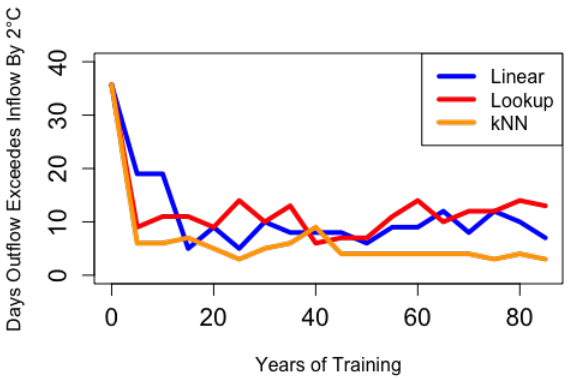

Figure 5: Learning curves for temperature showing the number of days each run that temperature exceeded the EPA metric during training on data from 2015.

Table 2: Temperature generalization results. Testing results are metrics under unseen data.

Mean days outflow temperature exceeds inflow by 2°C		
	Training	Testing
Random	11.4	11.7
Lookup	9.7	10.8
kNN	4.9	4.8
Linear	2.8	3.2

Both Linear and kNN algorithms, however, are able to largely stay below outflow temperatures under the static policy, signally good generalization to this year.

Discussion

For the elevation objective, all three algorithms were able to improve upon the objective over a random policy for both training and testing data. The Linear algorithm converges quickest as it has the fewest parameters to train and can generalize to unseen states. The other algorithms are slower to converge as they must explore more of the state space, but their learned policies better meet the objective on both training and testing data. This indicates that the discretized linear function approximation may have too much bias to adequately approximate the state-action Q-values.

For the temperature objective, better performance of the kNN and Linear algorithms on the test data were probably due to their abilities to generalize to unseen states, whereas Lookup table Q-learning requires that every state be explicitly visited. The Lookup algorithm was still choosing random actions from some states on the test data, which indicates the simulated training data may not have sufficient coverage of the test data's state-space. However, the near-random performance under training data as well indicates the Lookup algorithm may not have converged even after 24 hours of training run-time.

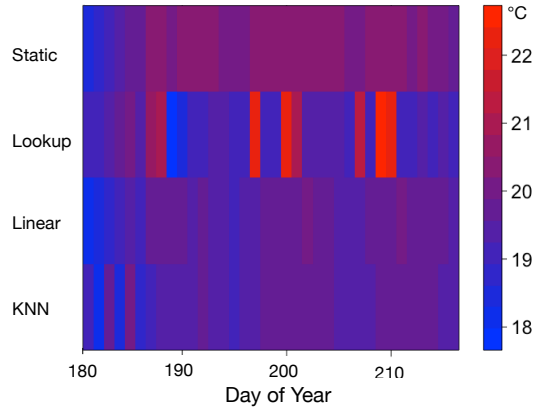

Figure 6: Example comparison of outflow temperatures under different control algorithms for unseen data.

The greatest challenge we faced was the substantial size of the continuous state space and the large computational expense to repeatedly run hydrodynamic simulations. The kNN algorithm addressed this challenge somewhat by compactly representing the full continuous space, while Lookup and Linear algorithms relied on more coarsely discretized features. While kNN often had the best performance with enough training time, the Linear algorithm was able to learn decent policies in fewer simulation runs.

Conclusion

Our work applies Q-learning to an atypical control target, a hydropower dam, and demonstrates that it can be used to find a policy for compliance with mandated thermal loading targets. We explored two separate optimization sub-goals that are necessary for balancing dam operations with thermal impact mitigation. Optimizing for temperature goals achieved good generalization to unseen data for kNN and Linear models. Optimizing for water elevation targets produced improvements under unseen data for all algorithms, but did not appear to be fully converged within the training timeframe. For both objectives, learning was clearly apparent, and improved skill and learning speed may be achieved by tuning reward functions and hyperparameters.

Further work should combine both elevation and temperature objectives during training, which may give more flexibility in reaching water temperature goals but will require handling even larger state spaces. Implementing a linear function approximation model with continuous features may improve both convergence and generalization by lifting the requirement for discretization while training fewer parameters, although adequate non-linear feature identification is likely non-trivial.

References

[1] B. Bhattacharya, A. Lobbrecht, and D. Solomatine. Neural networks and reinforcement learning in control of water systems. *Journal of Water Resources Planning and Management*, 2003.

[2] A. Castelletti, G. Corani, A. Rizzoli, R. Sessa, and E. Weber. Reinforcement learning in the operational management of a water system. 2002.

[3] A. Castelletti, S. Galelli, M. Restelli, and R. Soncini-Sessa. Tree-based reinforcement learning for optimal water reservior operation. *Water Resources Research*, 2010.

[4] A. Castelletti, F. Pianosi, and M. Restelli. A multiobjective reinforcement learning approach to water resources systems operation: Pareto frontier approximation in a single run. *Water Resources Research*, 2013.

[5] T. Cole and S. Wells. *CE-QUAL-W2: A Two-Dimensional, Laterally Averaged, Hydrodynamic and Water Quality Model*. U.S. Army Corps of Engineers and Portland State University, 4.0 edition, 2016.

[6] B. Cope. Temperature simulation of the snake river above lower granite dam using transect measurements and the ce-qual-w2 model. *EPA*, 2002.

[7] M. Dehard. Fish passage center memorandum october 28, 2015 re: Requested data summaries and actions regarding sockeye adult fish passage and water temperature issues in the columbia and snake rivers. 2015.

[8] J. T. Ding and J. He. Water quality forecast based on bp-artificial neural network model in qiantang river. *Applied Mechanics and Materials*, 668-669:994–998, Oct 2014.

[9] J. Labadie. Optimal operation of multireservoir systems: State-of-the-art review. *Journal of Water Resources Planning and Management*, 2004.

[10] J.-H. Lee and J. Labadie. Stochastic optimization of multireservior systems via reinforcement learning. *Water Resources Research*, 2007.

[11] J. Martin, J. de Lope, and D. Maravall. The knn-td reinforcement learning algorithm. June 2009.

[12] R. J. May, G. C. Dandy, H. R. Maier, and J. B. Nixon. Application of partial mutual information variable selection to ann forecasting of water quality in water distribution systems. *Environmental Modelling & Software*, 23(10-11):1289–1299, Oct 2008.

[13] NCDC. Quality controlled local climatological data, 2016.

[14] P. Sabes. Approximating q-values with basis function representations. *Proceedings of the Fourth Connectionist Models Summer School*, 1993.

[15] USGS. Water data for the nation, 2016.

[16] C. J. C. H. Watkins. Learning from delayed rewards. *PhD thesis, University of Cambridge, England*, 1989.

[17] J. Yearsley. Developing a temperature tmdl for the columbia and snake rivers: Simulation methods. epa 910-r-03-003. Technical report, US Environmental Protection Agency Region 10, 2003.

[18] I. S. Yeon, J. H. Kim, and K. W. Jun. Application of artificial intelligence models in water quality forecasting. *Environmental Technology*, 29(6):625–631, Jun 2008.