Designing Robust Control Rules for Stochastic Engineered Systems

A Thesis Presented to
The Faculty of of the School of Engineering and Applied Science
University of Virginia

In Partial Fulfillment
of the Requirements for the Degree
of Doctorate of Philosophy

Department of Systems and Information Engineering

By
Hossein Kavianihamedani
June 2024

ACKNOWLEDGMENTS

Before I thank anyone else, I must express my profound gratitude for Dr. Julianne Quinn. Dr. Quinn has not only been my mentor, but has also been someone I can turn to for motivation and guidance. I would not be where I am today without her constant support. Of course, I also owe my thanks to my committee members: Dr. Negin Alemazkoor, Dr. Laura Barnes, Dr. Antonios Mamalakis, and Dr. Seokhyun Chung. You all have played a crucial role in the culmination of my doctorate.

I would also like to thank the entirety of the University of Virginia's Research Computing Department. In particular, I'd like to extend my appreciation to Dr. Ruoshi Sun who played an influential role in assisting with my experiments. I'd like to wholeheartedly thank each one of you, I couldn't have done it without you.

Needless to say, I also owe much of my experience to my labmates within the Quinn Group: Samarth Singh, Sarah Jordan, Jared Smith, Daniel Lassiter, Mashood Ur Rahman, and Kajol Basnet. Each and every one of you has been instrumental to my success. I deeply appreciate the time we've spent working together and the friendship we built, which I am confident will last a lifetime.

Last and certainly not least, I would like to thank my friends and family, who have supported me every second during my pursuit of higher education. I thank my mom and dad for all the sacrifices they made, my brothers who always supported me, and especially my partner Leah, who encouraged and stood by me throughout this journey. I would also like to thank the sand volleyball community in Charlottesville. I was fortunate enough to be a part of this community while pursuing my PhD studies.

ABSTRACT

Limited availability of in-situ hydrological data stands as a challenge to modeling and managing water resources systems. This is because effective water infrastructure planning and management requires an understanding of natural system processes and their interactions with the built environment; yet limited in-situ data requires the estimation of model parameters describing these processes through calibration. Such parametric model uncertainty can have significant implications for infrastructure design; however, it is often ignored in the design stage. In response to this pressing issue, this research proposes to improve the quantification and management of uncertainty in hydrological models. First, we introduce innovative diagnostic tools to assess the performance of Markov Chain Monte Carlo (MCMC) algorithms in calibrating complex physical models with high-dimensionality and multimodality using analytical test problems as benchmark examples. Second, we propose to utilize our knowledge of effective algorithms gained through the first study to quantify parametric uncertainty in a Stormwater Management Model (SWMM) of an urbanizing system. We then propose to design stochastic multi-objective control rules for flood risk reduction that are robust to this uncertainty. The latter step will contribute to the literature through two papers: The first study will introduce Evolutionary Multi-Objective Direct Policy Search (EMODPS) to the stormwater control literature and compare it with Deep Deterministic Policy Gradient (DDPG), which has been used in designing stormwater control rules. The second study will provide insights into how to most effectively design stormwater control rules that account for parametric hydrological model uncertainty. This research promises to advance our understanding of how to better quantify and manage uncertainty in water resources systems.

CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS AND ABBREVIATIONS

AM ..	Adaptive Metropolis
ANN ..	Artificial Neural Networks
CDF ..	Cumulative Distribution Function
DDPG .	Deep Deterministic Policy Gradients
DE ...	Differential Evolution
DRAM .	Delayed Rejection Adaptive Metropolis
DREAM	Differential Evolution Adaptive Metropolis
DPS ..	Direct Policy Search
EMODPS	Evolutionary Multi-Objective Direct Policy Search
ESS ..	Effective Sample Size
GR ...	Gelman-Rubin
KDE ..	Kernel Density Estimate
KLD ..	Kullback-Leibler Divergence
LH ...	Latin hypercube
MAP ..	Maximum A-Posteriori
MCMC .	Markov Chain Monte Carlo
MH ..	Metropolis-Hastings
MPC ..	Model Predictive Control
MOEAs	Multi-Objective Evolutionary Algorithms
MORL .	Multi-Objective Reinforcement Learning
MORO .	Multi-Objective Robust Optimization
MSE ..	Mean Squared Error

MVN .	Multi-Variate Normal distribution
NCRBFs	Nonconvex Gaussian Radial Basis Functions
NFE ..	Number of Function Evaluations
O1 ...	Orifice 1
O2 ...	Orifice 2
RBC ..	Rule Based Control
RBFs ..	Radial Basis Functions
RL ...	Reinforcement Learning
RTC ..	Real Time Control
S1 ...	Subwatershed 1
S2 ...	Subwatershed 2
SUs ..	Service Units (
SSE ..	Sum of Squared Errors
SWMM	Stormwater Management Model
TD ...	Temporal Distance
WCSS .	Within Cluster Sum of Squares
WD ..	Wasserstein Distance

CHAPTER 1

Introduction

Decision making systems must contend with inherent uncertainties (Chankong & Haimes, 2008; Koutsoyiannis & Economou, 2003; Srikrishnan et al., 2022). These uncertainties could be structural uncertainties in the governing relationships that define how the system functions, parametric uncertainties within the structural relationships, or stochastic uncertainty in exogenous forcing (Srikrishnan et al., 2022). All these uncertainties can significantly impact the outcomes and effectiveness of decisions, making it crucial to account for them in the decision making process (Marchau et al., 2019; Lempert et al., 2013; Giuliani & Castelletti, 2016). This becomes even more challenging when optimizing complex systems with multiple conflicting objectives (Quinn et al., 2017).

One important real-world application of multi-objective decision making under uncertainty is in managing stormwater systems. Between 1980 and 2010, the urban land area in the United States expanded by 43% (Demographia), and it is expected to grow by an additional 43% by 2050 (Nowak & Walton, 2005). This urban expansion brings with it heightened risks of increased flooding. For instance, a study examining streamflows across the United States since 1980 revealed that with every 1% increase in impervious surface area, annual flood magnitudes, on average, increased by 3.3% (Blum et al., 2020). These risks will be further exacerbated by more frequent and severe weather events associated with climate change (Coumou & Rahmstorf, 2012). To mitigate the impact of rising flood risks, there will be a need for enhanced control of stormwater infrastructure.

Stormwater control decisions regarding flood mitigation are usually guided by physically-based models that suffer from significant parametric uncertainty. However, decision making frameworks applied to inform flood control in stormwater systems often focus strictly on designing for stochastic uncertainty from precipitation, potentially

under climate change(Dotto et al., 2012, 2014). Yet ignoring parametric uncertainty could have severe consequences. Oftentimes, multiple combinations of hydrological model parameters can yield similar performance in simulating values close to observations, called "equifinality" (Beven & Binley, 1992, 2014). These different possible parameter values may have different design implications. Therefore, it is crucial to identify system designs that are robust to parametric uncertainty and can be implemented with minimal expected regrets (Lempert, 2003; Marchau et al., 2019).

Bayesian inference methods can be utilized to characterize parameter uncertainty by updating prior beliefs about parameters and deriving their posterior distributions (Lim et al., 2006; Ning & You, 2017; Xie et al., 2023). Monte Carlo Markov Chain (MCMC) is a well-known statistical method used to estimate these posterior distributions and has been applied in stormwater and hydrology studies (Castelletti & Soncini-Sessa, 2007; Vrugt et al., 2009; **?**). However, employing MCMC can be computationally expensive, especially for models with a large number of parameters. Moreover, existing metrics (Gelman & Rubin, 1992; Vallender, 1974) and diagnostics (Roberts, 1992; Ritter & Tanner, 1992; Mykland et al., 1995) for assessing MCMC performance primarily focus on convergence rate and often neglect other critical aspects. These include the sensitivity of the algorithm's efficiency to hyperparameter settings, the consistency of the algorithm across various problem ranges, and the speed at which the algorithm estimates the posterior distributions.

While choosing the right statistical method to characterize parameter uncertainty is important, selecting the appropriate optimization tool is equally significant. Stormwater systems are inherently complex and involve multiple conflicting objectives (Di Matteo et al., 2017; Kumar et al., 2022). Control rules in these systems must be designed efficiently to meet various stakeholder requirements, such as mitigating flooding, improving

water quality, or ensuring habitat protection (Kumar et al., 2022). Deciding on a solution to balance these conflicting objectives is often best informed by deliberation over a set of diverse, "non-dominated" trade-off policies, known as the Pareto front. Unfortunately, prior studies on stormwater optimization have only searched for a single policy to balance conflicting objectives, rather than illustrating the full Pareto front of different policies and their trade-offs to stakeholders (Saliba et al., 2020; Yu et al., 2022). Additionally, because no studies have performed multi-objective optimization of stormwater control rules, the stormwater literature lacks a comprehensive comparison of alternative approaches to finding a Pareto front. It is not clear whether it is better to use value-based reinforcement learning approaches that have been used for single-objective problems (Bowes et al., 2021, 2022) with different weights on the conflicting objectives, or policy-based reinforcement learning with a multi-objective evolutionary algorithm, as has been done in the reservoir operations literature (Giuliani et al., 2021). This gap underscores the need for a comprehensive evaluation of multi-objective reinforcement learning algorithms to enhance the decision-making process in stormwater management.

In this thesis, we seek to address these gaps in the literature to advance robust optimization methods for multi-objective stormwater control. Multi-objective robust control requires 1) selecting an appropriate algorithm for uncertainty quantification, 2) selecting an appropriate algorithm for multi-objective reinforcement learning (MORL), and 3) determining how to couple the uncertainty quantification with the reinforcement learning algorithm to design control rules that are robust to it. We devote one chapter to each of these steps.

First, in chapter 2, we present new diagnostics to assess MCMC algorithms in terms of their effectiveness (ability to accurately find representative posterior modes), efficiency (speed of posterior characterization), reliability (consistency across different random

seeds), and controllability (insensitivity to hyperparameter variation) (Kavianihamedani et al., 2024). The findings from this study can help users select the right MCMC algorithm for their specific problem. For example, if a problem is low-dimensional or known to be unimodal, engineers may choose one algorithm based on our findings, whereas if it is high-dimensional or its modality is unknown, they may choose another.

Second, in chapter 3, we introduce Evolutionary Multi-Objective Direct Policy Search (EMODPS) (Giuliani et al., 2016) to the stormwater control literature. We then compare the performance of this strictly policy-based RL approach in generating a Pareto front of control rules with that of a value-based RL approach that has been used in the stormwater control literature for single-objective problems: Deep Deterministic Policy Gradient (DDPG) (Mullapudi et al., 2020; Bowes et al., 2021). This study provides insights into how to most effectively design stormwater control rules for multiple conflicting objectives.

Lastly, in chapter 4, leveraging the insights from the first two studies, we design a robust optimization framework to address parameter uncertainty in multi-objective stormwater control. We introduce two robust optimization approaches: Multi-Objective Robust Optimization (MORO), which designs control rules to optimize the posterior-weighted average of multiple likely parameterizations, and Min-Max Optimization, which designs control rules considering the worst-case of the likely parameterizations. These are compared with the traditional approach of simply designing control rules to the Maximum A-Posteriori (MAP) parameter set. The findings from this study can help decision-makers incorporate robust optimization strategies into their systems, thereby improving stormwater management.

Finally, the overarching insights across all three of these studies is summarized in chapter 5, along with a discussion of areas for future work.

CHAPTER 2

New Diagnostic Assessment of MCMC Algorithm Effectiveness, Efficiency, Reliability, and Controllability

This chapter is largely reproduced from the following publication:

> Kavianihamedani, H., Quinn, J.D., & Smith, J.D. (2024). New Diagnostic Assessment of MCMC Algorithm Effectiveness, Efficiency, Reliability, and Controllability. *IEEE Access*. *12*, 42385-42400.

2.1 ABSTRACT

Markov Chain Monte Carlo (MCMC) is a robust statistical approach for estimating posterior distributions. However, the significant computational cost associated with MCMC presents a considerable challenge, complicating the selection of an appropriate algorithm tailored to the specific problem at hand. This study introduces a novel and comprehensive framework for evaluating the performance of MCMC algorithms, drawing inspiration from diagnostics used for multi-objective evolutionary algorithms. We employ visualizations to evaluate key algorithmic characteristics: Effectiveness (the ability to accurately find representative posterior modes, quantified by the Kullback-Leibler Divergence (KLD) and Wasserstein Distance (WD)), Efficiency (the speed of posterior characterization), Reliability (consistency across different random seeds), and Controllability (insensitivity to hyperparameter variation). Evaluating three prominent MCMC algorithmsMetropolis-Hastings (MH), Adaptive Metropolis (AM), and Differential Evolution Adaptive Metropolis (DREAM)on high-dimensional and bimodal test problems, our analysis uncovers several insights. First, across algorithms, the number of function evaluations most controls performance on the high-dimensional problem, while the

number of chains most controls performance on the bimodal problem. While this suggests similar controllability across algorithms, differences emerge on the other algorithmic characteristics. For high numbers of functions evaluations, AM performs best on the high-dimensional problem, while for low (<5) and high (>15) chain counts, MH and AM perform best on the bimodal problem, as measured by KLD. However, outside these specific cases, DREAM consistently demonstrates superior efficiency and reliability, making it a robust choice for both high-dimensional and multimodal problems. These findings can inform MCMC algorithm selection for Bayesian inference applications, as well as hyperparameterization of the chosen algorithm. More importantly, the diagnostics represent a generalizable contribution to research on MCMC diagnostics that can be used to evaluate and inform the design of new algorithms.

2.2 INTRODUCTION

Bayesian inference can be used to estimate the parameters, θ, of a model and their associated uncertainty, given the available data. This is useful for informing robust engineering designs that can tolerate this uncertainty; see example applications in Lim et al. (2006); Mandur & Budman (2012, 2014); Campbell & How (2015); Liang & Mahadevan (2015); Ning & You (2017); Xie et al. (2023). The approach relies on Bayes' theorem in which the modeler uses their knowledge of the system's physical behavior and mathematical constraints to develop a prior probability distribution for the parameters, $p(\theta)$, that is updated by the likelihood of observing the data x, $p(x|\theta)$. This allows the estimation of the posterior probability of that parameter set given the observed data, $p(\theta|x)$:

$$p(\theta|x) = \frac{p(x|\theta)p(\theta)}{\int p(x|\theta)p(\theta)d\theta}. \tag{2.1}$$

The posterior distribution represents the uncertainty of the model parameters to which we would like engineering designs to be robust.

Markov Chain Monte Carlo (MCMC) is a powerful statistical method used to estimate posterior distributions. MCMC uses a Monte Carlo simulation to sample from a Markov Chain whose transition probabilities from the current point in the chain to the next proposed point in the parameter space are determined by the relative posterior density of the current and proposed locations. In the long run, the distribution sampled by the Markov chain will become stationary and represent the true posterior distribution. However, using posterior estimates from MCMC prematurely before the algorithm has converged to its stationary distribution could result in a poor characterization of uncertainty, and consequently over- or under-designed systems. As such, a vast body of literature has focused on developing diagnostics to assess the convergence of MCMC search processes (see reviews of MCMC diagnostics by Cowles & Carlin (1996); Roy (2020)). Specifically, these diagnostics focus on determining 1) how many initial iterations should be discarded as "burn-in" because they are far from the posterior mode and not representative of the stationary distribution; and 2) how many iterations are sufficient to stop the algorithm, as the chain has converged to its stationary distribution (Jones & Hobert, 2001).

In some cases, one can calculate these numbers theoretically. For example, if it can be shown through drift and minorization criteria (Rosenthal, 1995) that the chain converges at an exponential rate, a bound on the number of burn-in iterations needed for the total variation distance between the estimated and true posterior to reach some tolerance can be calculated (Roberts & Tweedie, 1999). A similar estimation can be performed using Wasserstein distance (Durmus & Moulines, 2015). With respect to the total number of iterations, if a Central Limit Theorem exists for some function g of the samples X (such as their mean or variance), one can estimate the sample size needed for $g(X)$ to converge

to its true value at some confidence level (Roy, 2020).

Unfortunately, for black box-type parameter estimation problems that are common in engineering, no such theoretical bounds can be estimated (Geyer, 2011). Instead, MCMC users must rely on visual or quantitative metrics of convergence, many of which require running multiple chains (Cowles & Carlin, 1996). Visual MCMC diagnostics monitor the progress of the search using graphical tools like trace plots, histograms, and correlograms. These visual aids cannot identify with certainty that a chain has converged, but they can identify problems that indicate it has not. For example, trace plots of the chain location vs. iterations illustrate if the chains are getting stuck, moving too slowly due to high auto-correlation, or trending and therefore not yet stationary; histograms indicate if the posterior estimates across chains are inconsistent with one another; and correlograms indicate if the autocorrelation in the chains is too high, reducing the effective sample size (ESS) (Roy, 2020). Plots of more complex chain statistics have also been proposed (Roberts, 1992; Ritter & Tanner, 1992; Mykland et al., 1995), but these are problem or algorithm-specific.

Because graphical plots can only identify problems in convergence, these diagnostics are typically complemented by numerical convergence metrics that provide more objective stopping criteria. The most common metric, the Gelman-Rubin (GR) diagnostic, calculates the ratio of the variance across chains to the average within-chain variance, with a value < 1.1 recommended as a stopping criterion (Gelman & Rubin, 1992). In multivariate settings, i.e. when calibrating multiple parameters, one can enforce this across all parameters or use the multivariate adaptation of the metric (Brooks & Gelman, 1998). Another common stopping criterion is a threshold of the (multivariate) Effective Sample Size (mESS), which accounts for autocorrelation in the chain (Robert et al., 2004; Vats et al., 2019). Heidelberger & Welch (1983) perform hypothesis tests for stationarity of the

Markov chain at different points in the chain to determine how much to remove as burn-in because the null hypothesis that the chain is stationary is rejected when that portion is included. In a similar vein, others perform hypothesis tests comparing kernel density estimates across pairs of chains to determine if they have the same distribution (which is assumed to be the stationary distribution). Metrics for such tests include the L-1 distance (Yu et al., 1995), Hellinger distance (Boone et al., 2014) or Kullback-Leibler Divergence (KLD) (Dixit & Roy, 2017). If any of these graphical or quantitative diagnostics indicates non-convergence, adjustments to the search process can be made, such as "thinning" the chain by only retaining every k samples to reduce auto-correlation, modifying the probability of using different operators that are used to propose new chain locations, adapting the algorithm's hyperparameters (e.g. covariance matrix) to improve exploration, or simply extending the search duration (Gelman et al., 2013).

While these metrics are useful for identifying if an individual search process has not converged, they provide limited insights into how to improve convergence. The conventional approach of manually tuning algorithmic hyperparameters to improve performance can be laborious, and recommended default ranges may not always perform well. Ideally, an algorithm should exhibit robustness to its hyperparameterization and be primarily controlled by the number of function evaluations (NFE) (Hadka & Reed, 2012; Reed et al., 2013). However, existing diagnostics do not measure this controllability. Furthermore, simply diagnosing performance of an individual search process does not provide insights into which algorithms perform well on which class of problems, and which are robust across problems. To address these limitations, we propose new diagnostic tools to evaluate MCMC algorithms and inform the choice of suitable methods for specific types of inverse problems.

Drawing from diagnostics used to evaluate the performance of multi-objective evo-

lutionary algorithms (Hadka & Reed, 2012; Reed et al., 2013), in this study, we present a novel and comprehensive framework for evaluating MCMC algorithm performance. Our approach provides visualizations that show existing diagnostic metrics in a new way, illustrating the following algorithmic characteristics:

- *Effectiveness*: A measure of the ability of an MCMC algorithm to find a posterior mode (or multiple modes) that is (are) representative of the true uncertainty, and to characterize the full posterior distribution. Existing metrics include L-1 distance, Hellinger distance, and KLD.

- *Efficiency* The speed with which the posterior is able to be characterized. Existing metrics include the ESS and mESS.

- *Reliability*: How consistently the algorithm is able to characterize the posterior across different random seeds. This is typically quantified by the GR diagnostic.

- *Controllability*: The insensitivity of an algorithm's efficiency to its hyperparameterization, a desirable property so that the user does not have to fine-tune hyperparameters to achieve good performance. This is not typically quantified in MCMC diagnostics.

Diagnosing these features collectively across algorithmic hyperparameters and random seeds fills an important gap in the literature that only diagnoses convergence of a single search process, ignoring algorithmic controllability across hyperparameterizations. The visualizations we produce of these characteristics can inform the choice of a robust MCMC algorithm and corresponding hyperparameterization, whose convergence can then be assessed using existing diagnostics. As such, our new MCMC diagnostics play a complementary role to existing MCMC diagnostics.

Our study is organized as follows. Section briefly describes the MCMC algorithms we compare 2.3, Section 2.4 outlines the experimental design used for this comparison, Section 2.5 lists the metrics used to quantify performance, and Section 2.6 introduces the test problems on which the algorithms are evaluated. We illustrate the results of this computational experiment and our new diagnostics in Section 2.7. Finally, we close with our conclusions about MCMC algorithm performance illustrated by our new diagnostics in the conclusions.

2.3 ALGORITHMS

In this section, we describe the three Bayesian estimation algorithms examined in our study: Metropolis-Hastings (MH), Adaptive Metropolis (AM), and Differential Evolution Adaptive Metropolis with a snooker update and sampling from an archive of past states ($\text{DREAM}_{(ZS)}$). These algorithms serve as powerful tools for exploring and sampling from complex parameter spaces in Bayesian analysis. While there are other algorithms for Bayesian estimation, we limit our exploration to these three for illustrative purposes of our new diagnostics. However, our diagnostics can be extended to other algorithms.

All algorithms were implemented using the BayesianTools package in R (Hartig et al., 2023), which provides general-purpose MCMC samplers for Bayesian statistics. This package offers a wide range of functionalities for efficient implementation and analysis of Bayesian models, making it an accessible tool for conducting advanced Bayesian inference tasks, such as comparing alternative algorithms.

2.3.1 Metropolis Hastings (MH)

The MH algorithm (Metropolis et al., 1953; Hastings, 1970) is a widely used MCMC method that enables sampling from complex posterior distributions. First, an initial pa-

rameter set θ_0 is sampled from the prior distribution and then new parameters θ' are generated (proposed) from a proposal distribution that is centered about the current location. MH proposes new parameter sets by using a symmetric proposal distribution, typically a multivariate normal distribution (MVN), as is implemented in BayesianTools. This is referred to as Gaussian mutation. A proposed move is accepted with probability α, determined by equation 2.2:

$$\alpha = \min\Big(1, \frac{p(\theta'|x)g(\theta_t|\theta')}{p(\theta_t|x)g(\theta'|\theta_t)}\Big) \tag{2.2}$$

where $g(\theta'|\theta_t)$ is the probability of proposing parameters θ' given the current parameters are θ_t, and $g(\theta_t|\theta')$ is the reverse. Note that, $g(\theta'|\theta_t) = g(\theta_t|\theta')$ if the proposal distribution is symmetric. This is referred to as the Metropolis step, or accept-reject step.

In BayesianTools, the initial samples of the chain can be optimized at an estimate of the maximum of the posterior distribution, with the goal of reducing the amount of burn-in by starting in a high posterior density region. This is controlled by a binary hyperparameter Optimize = True or False. If true, BayesianTools utilizes the Brent algorithm (Brent, 1971) for single-parameter estimation problems, and the Nelder-Mead algorithm (Nelder & Mead, 1965) for multi-dimensional problems, both of which are derivative-free. Nelder-Mead algorithm may converge to a non-stationary point (McKinnon, 1998), and it is a local optimizer, therefore it may not do well on multi-modal problems. The other hyperparameters of MH algorithm are the total number of function evaluations, the number of chains, and percent of function evaluations to remove as burn-in (see Table 3.1 for a list of the hyperparameters in each algorithm).

2.3.2 Adaptive Metropolis (AM)

MH provides a foundational framework for Bayesian inference and has been successfully applied in various fields. However, one limitation of MH is the fixed proposal distribution, which may not effectively explore high-dimensional or multi-modal parameter spaces. To address this limitation and improve exploration efficiency, the AM algorithm (Haario et al., 2001) incorporates adaptive strategies for updating the covariance of the proposal distribution throughout the search. This adaptation is determined by the points sampled during the MCMC process. By adaptively updating the proposal covariance, AM strikes a balance between exploration and exploitation in the parameter space. It allows the algorithm to explore regions of high uncertainty by increasing the variance when uncertainty across sampled points is high, leading to better mixing of the Markov chains. However, it also allows the algorithm to exploit regions of high probability density by decreasing variance when uncertainty across sampled points is low, thus improving convergence. While this may be unnecessary for low-dimensional problems for which MH may be faster, the adaptive nature of AM makes it more effective in high-dimensional parameter spaces and when dealing with complex posterior distributions. This adaptivity enhances the exploration capabilities of the algorithm, resulting in improved efficiency and convergence rates (Roberts & Rosenthal, 2009). In BayesianTools, adaptation is controlled by two hyperparameters: AdaptStart, which indicates the percent of evaluations after burn-in at which adaptation begins, and AdaptInterval, which indicates the fraction of remaining evaluations after AdaptStart at which adaptation occurs.

We allow AM to be employed with delayed rejection, also called Delayed Rejection Adaptive Metropolis (DRAM) (Haario et al., 2006). In DRAM, once a proposed point has been rejected, instead of proceeding to the next time step and remaining in the current state, a second-stage proposal is made that depends on *both* the current state and

the state that was just proposed and rejected. The second-stage proposal is then accepted or rejected based on a modified acceptance probability that preserves reversibility of the Markov chain. This can be repeated multiple times, the number of which is controlled in our experiment by the parameter DRlevels (see Table 3.1). We also allow for optimization of initial starting points in the AM search, controlled by a binary Optimize hyperparameter, as in MH.

2.3.3 Differential Evolution Adaptive Metropolis (DREAM)

While the ability to adapt the proposal distribution through AM can speed up convergence with respect to MH, it is still limited by using a single proposal operator (typically, Gaussian mutation). The DREAM$_{(ZS)}$ algorithm (Laloy & Vrugt, 2012) is a population-based MCMC method that advances AM further by adding additional proposal operators to the AM algorithm: differential evolution (DE) and a snooker update (S). This can further enhance exploration on high-dimensional, multi-modal problems, but may come at the expense of deeper exploitation of high-posterior regions. For simplicity, we refer to this algorithm as simply "DREAM" throughout the remainder of the chapter.

The population of DREAM refers to the states of multiple chains, as well as an archive of their past states. These are used jointly to propose new chain locations using operators beyond Gaussian mutation, including DE and a snooker update, which are accepted according to the Metropolis rule. DE is a vector translational operator originally developed for use in evolutionary optimization algorithms (Storn & Price, 1997). Mathematically, DE can be described by equations 2.3-2.4 (Vrugt et al., 2009):

$$\theta_i' = \theta_{i,t} + \gamma(1+e)\left[\sum_{n=1}^{p}\theta_{j(n)} - \sum_{m=1}^{p}\theta_{k(m)}\right] + \epsilon N(0,1) \tag{2.3}$$

$$\gamma = \frac{2.38}{\sqrt{p * d}} \tag{2.4}$$

where $\theta_{i,t}$ and θ'_i are the current and proposed states of the i-th chain, respectively; $\theta_{j(n)}$ and $\theta_{k(m)}$ are the n-th and m-th of p samples from the archive of current or past states of the j-th and k-th chains, respectively; d is the problem dimension (i.e., number of model parameters); e is a constant chosen by the user to scale γ if desired (Ter Braak & Vrugt (2008) choose the default value of γ in equation 2.4 to yield acceptance rates close to 0.44 for d = 1 and 0.23 for large d, which have been shown numerically and theoretically to be optimal acceptance rates for random walk Metropolis (Gelman et al., 1996; Roberts & Rosenthal, 2001)); and ϵ is the variance of a Gaussian mutation after DE translation, whose value is also chosen by the user.

The DE translation in equation 2.3 is typically only applied to some of the dimensions. These are referred to as "crossover points" and the number of crossover points is determined by the nCr parameter. The value of this parameter can be updated throughout the search with frequency determined by the parameter UpdateInterval. Similarly, the archive of past states, **Z**, is updated with frequency zUpdateFrequency.

A snooker update is another vector translational operator originally proposed by Gilks et al. (1994) to adapt sampling in the direction of the highest density. DREAM$_{(ZS)}$ uses an updated snooker proposal operator developed by Ter Braak & Vrugt (2008), described mathematically by equation 2.5:

$$\theta'_i = \theta_{i,t} + \gamma_s(\theta^P_{j,t} - \theta^P_{k,t}) \tag{2.5}$$

where γ_s is another constant hyperparameter of the algorithm, while $\theta^P_{j,t}$ and $\theta^P_{k,t}$ are orthogonal projections of $\theta_{j,t}$ and $\theta_{k,t}$ onto the line $\theta_{i,t} - \theta_{n,t}$, where $\theta_{n,t}$ is the current state of another chain, n.

The additional operators of DREAM, as well as its use of an archive and interaction across chains, serve several beneficial purposes. The archive, which maintains a history of accepted samples from all chains, enables a more efficient exploration of the parameter space and improved mixing of the chains. By sampling from the past archive, the algorithm gains access to valuable information about the posterior distribution, enhancing its ability to explore diverse regions and locate multiple modes. The incorporation of DE and snooker moves within DREAM$_{(ZS)}$ further enhances exploration by introducing a stochastic perturbation mechanism. This mechanism helps to overcome local optima and encourages the chains to traverse the posterior distribution more effectively. Finally, the interaction across chains allows for greater exploration and facilitates convergence to the same posterior across chains (Laloy & Vrugt, 2012; Nishihara et al., 2014).

2.4 COMPUTATIONAL EXPERIMENT

In order to evaluate the performance of the MCMC algorithms used in this study, a comprehensive experimental setup was devised, representing the key contribution of this paper. The experimental design, which is inspired by Hadka & Reed (2012) and Reed et al. (2013), aims to assess the effectiveness, efficiency, reliability, and controllability of the MCMC algorithms in converging to the true posterior distribution.

Figure 2.1 illustrates the experimental design that enables this assessment. The first step is to generate a Latin hypercube (LH) sample of algorithmic hyperparameters. Here we use a sample of 1000. Two of the hyperparameters are the number of function evaluations (NFE) and number of chains. For each of the LH samples, MCMC is performed with the corresponding hyperparameters and the posteriors are estimated empirically. The final posteriors consist of the elements from all chains, excluding the initial burn-in period. For instance, with 1000 iterations, 5 chains, and a 100-iteration burn-in, the resulting pos-

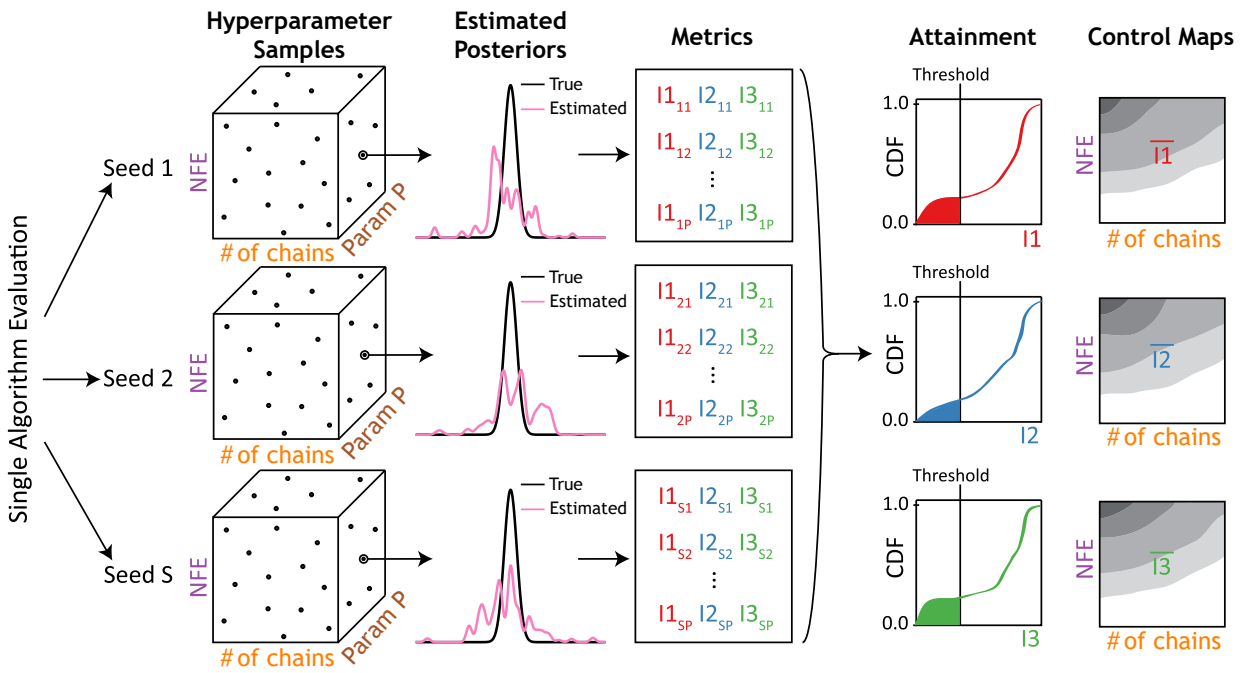

Figure 2.1: Experimental design of this study. For each algorithm and test problem, calibration is performed across a Latin hypercube sample of algorithm hyperparameters for multiple seeds. Performance metrics are computed based on the proximity of the estimated posterior to the true posterior. The reliability of the algorithm is illustrated by CDFs of the probability of attaining certain performance levels on the metrics, while the effectiveness, efficiency and controllability are illustrated in control maps of the average performance metric as a function of the number of function evaluations (NFE) and chains. This figure is adapted from Reed et al. (2013).

terior consists of (1000 - 100) * 5 = 4500 chain locations. Since the Monte Carlo aspect of MCMC is random, this process is repeated for multiple random seeds, here 25.

Next, several metrics of the algorithm's *effectiveness* are computed (described in Section 2.5) for each random seed of each LH sample. The *reliability* in achieving these metrics is visualized by a Cumulative Distribution Function (CDF) across random seeds, also called an "attainment map" as it illustrates the probability of attaining different metric values. The *efficiency* is visualized by a contour map of the average metric across random seeds of each LH sample, shown on a 2D projection of the LH samples' NFE and number of chains. The sooner effective values are reached vs. NFE, the more efficient the algo-

rithm. This plot, also called a "control map", illustrates how *controllable* the algorithm is; the noisier the contour map the less its performance is controlled by the NFE and number of chains and more by its other hyperparameters.

We also measure controllability quantitatively by performing variance-based sensitivity analysis, decomposing how much variance in the performance metric is explained by each hyperparameter. The more variance explained by NFE (and subsequently, the number of chains), the more controllable the algorithm, as these are the easiest hyperparameters for the user to set. The fraction of the variance in the performance metric Y explained by the i-th hyperparameter X_i individually is denoted its first-order sensitivity index, S_i:

$$S_i = V_i / Var(Y) \tag{2.6}$$

$$V_i = Var(E[Y|X_i]). \tag{2.7}$$

Any remaining variability is assumed to be explained by interactions across hyperparameters. Sensitivity indices were estimated using the method of Plischke et al. (2013) using the Python SALib package (Herman & Usher, 2017).

The ranges of the hyperparameters for the LH samples are detailed in Table 3.1. These ranges were informed by values from the literature and were carefully selected to cover a broad spectrum of possible configurations, ensuring a thorough exploration of the algorithm's behavior (Laloy & Vrugt, 2012) and (Hartig et al., 2023). By varying the hyperparameters, we not only are able to assess the algorithm's controllability, but also to identify the settings that yield optimal results for different types of problems (e.g. high-dimensional or multi-modal). Sensitivity to additional hyperparameters could be explored in future work, such as the initial covariance matrix of the Gaussian proposal

distribution, or the interval of samples that should be dropped via thinning. Sensitivity to thinning could be further investigated to determine the extent that autocorrelation decreases the effective sample size which also increases the standard error estimates of the posterior mean. Investigating the sensitivity on an MCMC algorithm's performance to the thinning interval could inform the choice of effective ranges to reduce the impact of autocorrelation on the reliability of MCMC simulations.

The experimental framework was implemented on the Rivanna high-performance computing cluster at the University of Virginia. The insights derived from this experiment can provide guidance for selecting appropriate algorithms and corresponding configurations for inverse problems with the tested characteristics, as well as inform how to develop new algorithms with improved controllability by adapting more sensitive hyperparameters throughout the search. Finally, it illustrates a new framework for evaluating MCMC algorithms developed in the future.

2.5 METRICS

To evaluate the effectiveness of the MCMC algorithms, we use three performance metrics that quantify different aspects of convergence: the GR diagnostic (Gelman & Rubin, 1992), KLD (Kullback & Leibler, 1951), and WD (Dobrushin, 1970). These metrics provide valuable insights into the quality of the MCMC samples and the approximation of the target distribution.

2.5.1 Gelman-Rubin (GR) diagnostic

The GR diagnostic is a widely used measure to assess convergence when multiple, independent MCMC chains are employed and the true posterior is unknown. It compares the within-chain variance to the between-chain variance:

Table 2.1: Ranges of algorithmic hyperparameters sampled uniformly by Latin hypercube sampling.

Hyperparameters across algorithms		
Hyperparameter	Description	Range
NFE	Number of function evaluations	10,000-200,000
nChains	Number of chains	2-20
Burn-in	Percent of function evaluations to discard	1-20
Additional MH and AM hyperparameters		
Optimize	Binary variable indicating whether to optimize the starting locations	0 (no) or 1 (yes)
Additional AM hyperparameters		
AdaptStart	Percent of Evaluations after Burn-in at which adaptation begins	0.5-5
AdaptInterval	Fraction of remaining evaluations after AdaptStart at which adaptation occurs	0.1-1
DRlevels	Number of levels for a delayed rejection sampler	1-2 (integer)
Additional DREAM(ZS) hyperparameters		
Adapt	Portion of iterations used in adaptation	0-1
P(snooker)	Probability of a snooker update at each iteration	0-1
nCr	Number of crossover points, i.e. dimensions of the current state changed in the proposal	1-5
p	Number of state pairs used to generate proposal with DE	1-3
ϵ	Variance of Gaussian mutation of DE translation (equation 2.3)	0-0.00005
e	Constant in equation 2.3 for DE proposal computation	-0.1-0.1
UpdateInterval	Interval number of iterations at which P(crossover) is updated	1-20
zUpdateFrequency	Interval number of iterations of evaluations after burn-in at which the archive is updated	1-20

$$\hat{R} = \sqrt{\frac{\hat{V}}{\hat{W}}} \tag{2.8}$$

where $\hat{V}$ is the estimated marginal posterior variance of the target parameter across all chains and $\hat{W}$ is the estimated average within-chain variance of the target parameter. A GR value close to 1 indicates convergence to the same variance across chains, making it an easy-to-interpret metric. The GR diagnostic is computed using BayesianTools.

We note that the GR diagnostic is meant to be used to ensure convergence to the same variance across *independent* chains, and is therefore not an appropriate measure of convergence for DREAM since the chains communicate. This communication will likely result in a low GR early in the search, even if the algorithm has not converged. However, consistent variance across chains may not be an appropriate measure of convergence even in the case of independent chains, as the chains could represent consistently poor approximations of the true posterior. Despite these limitations, GR is still the most commonly employed MCMC convergence metric when the posterior is unknown, including for the DREAM algorithm (Laloy & Vrugt, 2012). As such, we still compute the GR for all algorithms, but also compute additional performance metrics that allow us to assess the utility of GR as an MCMC performance metric.

2.5.2 Kullback-Leibler Divergence (KLD)

GR is a proxy measure of convergence used when the true posterior is unknown. However, as discussed above, it can prematurely indicate convergence, particularly when the true posterior is multi-modal. For test problems where the true posterior is known, we can assess convergence using the KLD. KLD measures the difference between two probability distributions, $D_{\mathrm{KL}}(P \parallel Q)$, as the integrated divergence in probability of one pdf $P(\theta)$ (here, the estimated posterior) to another $Q(\theta)$ (here, the true posterior):

$$D_{\mathrm{KL}}(P \parallel Q) = \int P(\theta) \log \left(\frac{P(\theta)}{Q(\theta)} \right) d\theta \tag{2.9}$$

KLD is commonly used in Bayesian statistics to assess the approximation of the true posterior distribution obtained from an MCMC algorithm when the true posterior is known, as is the case on test problems. It provides a measure of the dissimilarity between the approximate and true posterior distributions, allowing for flexible comparison of distributions with different parametric forms. However, the choice of the reference distribution (P vs. Q) can influence the results, as the measure is not symmetric. While there is a symmetric measure of KLD (Jeffrey's divergence)(Jeffreys, 1948), we simply set the reference distribution to the true posterior for consistency. We use the R function KL.divergence in the FNN library to computed KLD (Boltz et al., 2007, 2009).

2.5.3 Wasserstein Distance (WD)

The WD, or "Earth mover's distance" is another measure of the similarity between two distributions. It measures the minimum transport distance to transform one probability distribution into another (Kantorovich, 1960). It can be used in MCMC diagnostics to compare the true posterior distribution to the estimated posterior distribution obtained from the algorithm:

$$\mathrm{W(P, Q)} = \inf_{\gamma \in \Gamma(P,Q)} \int \int \|x - y\| \, d\gamma(x, y) \tag{2.10}$$

where $W(P, Q)$ represents the WD between distributions P and Q; $\inf$ denotes the infimum, which represents the minimum value over all possible transports γ that move mass from P to Q; $\Gamma(P, Q)$ is the set of all joint probability distributions $\gamma(x, y)$ with marginals P and Q; and $\|x - y\|$ represents a chosen distance metric between points $\overrightarrow{x}$ and $\overrightarrow{y}$ in the

underlying space.

The WD provides a measure of the discrepancy between two distributions, considering their underlying structure (Vallender, 1974). It can handle distributions with different supports. However, it can be computationally demanding, especially for high-dimensional distributions. Additionally, the choice of the distance metric may influence the results. To estimate WD, we generate n points $\overrightarrow{y}$ from the true posterior Q where n is equal to the total number of samples $\overrightarrow{x}$ from the MCMC chains after removing burn-in, which represent the estimated posterior P. The WD between these sets of points is computed using the Sinkhorn approximation (Cuturi, 2013) from the Python geomloss library, with transport distance between two points quantified by Euclidean distance.

Comparing the KLD and WD, KLD quantifies how dissimilar the estimated posterior probability is at each point θ compared to the true posterior probability, while WD compares how far the distributions are from one another in parameter-space. As such, KLD may be a better approximation of how close the estimated posterior is from the truth, while WD may be a better approximation of how far the search is from finding the true posterior in parameter-space.

2.6 TEST PROBLEMS

Because MCMC is typically applied to estimate the parameters of complex physical models, it would be useful to apply our diagnostics to such models. However, the KLD and WD metrics require a known posterior, so one would have to set a synthetic true parameter set to apply our diagnostics to a physical model. The posterior would then be a dirac delta function at the synthetic truth, and the KLD would be infinite. However, the WD could still be computed as the average Euclidean distance between all chain elements and the synthetic truth. This would capture closeness of the estimated posterior to the truth

in parameter space, but not probability space. Because of these challenges, we simply focus our diagnostics on two analytical test problems that address two prevalent challenges encountered in complex models: high dimensionality and multi-modality.

2.6.1 High-Dimensional Test Problem

Physical and data driven systems often involve a large number of interconnected variables, leading to high-dimensional parameter spaces. To simulate such scenarios, we employ a 100-dimensional multivariate normal distribution with a mean of $[0]^d$ and covariance Σ where the off-diagonal elements $\sigma_{i,j} = \frac{1}{2}\sqrt{i * j}\ \forall i \neq j$ and the diagonal elements $\sigma_{i,i} = i$. This test problem is commonly used to represent high-dimensional data (Vrugt et al., 2009).

The choice to target high dimensionality is motivated by the need to develop robust techniques capable of effectively exploring and optimizing parameter spaces in physical and statistical models. Relevant model applications span a wide range of fields including but not limited to machine learning (Radovanovic et al., 2010), climate (Cannon, 2018), and finance (Heaton et al., 2016).

2.6.2 Bimodal Test Problem

Multi-modal behavior, characterized by the simultaneous existence of distinct modes or regions of high probability in the parameter space, is a prevalent phenomenon observed in various domains, including machine learning and statistics (Choi & Lee, 2019), natural language processing (Wang et al., 2020), climate modeling (Mann & Park, 1994), and economics (Giles et al., 2010), where data often exhibits multiple diverse patterns or states. To address this characteristic, we employ a 10-dimensional bimodal mixed Gaussian distribution as our multi-modal test problem. The bimodal mixed Gaussian distribution con-

sists of two distinct modes, each following a Gaussian distribution with means of $[-5]^d$ and $[5]^d$ and a common covariance matrix $\Sigma = I$, the identity matrix. The mode with mean $[-5]^d$ occurs with probability 1/3 and the mode with mean $[5]^d$ with probability 2/3. By employing such a distribution, we can assess the ability of our proposed approach to effectively locate and characterize multiple optima within the parameter space, a key challenge encountered in physical modeling.

2.7 RESULTS AND DISCUSSION

2.7.1 Diagnostics on 100D MVN Test Problem

In this section, we present our diagnostics on the 100D MVN test problem. Figure 2.2 displays two sets of maps to illustrate the controllability, reliability, and efficiency of the MCMC algorithms using the WD metric: Control and Attainment Maps. SI Figures S1-S2 show the same maps for the KLD and GR of the first dimension, respectively, which revealed similar findings for KLD as presented in Figure 2.2, while all algorithms did well on GR across hyperparameterizations. Because our sensitivity analysis revealed that MH and AM were most sensitive to whether or not optimization was used to initialize the starting locations of each chain, we present these maps separately for the cases where optimization = True vs. False, yielding five algorithms for the comparison: MH_{noOpt}, MH_{opt}, AM_{noOpt}, AM_{opt}, and DREAM.

The control maps in Figures 2.2a-2.2e illustrate the average WD between the estimated and true posterior across 25 random seeds as a function of the algorithm's chain count (x-axis) and number of function evaluations (NFE; y-axis). The sooner a low value in blue is reached along the y-axis, the more efficient the search is. Noise in the control maps indicates less controllability, i.e. greater sensitivity to other hyperparameters beyond NFE and number of chains. Some hyperparameter combinations failed to yield posterior dis-

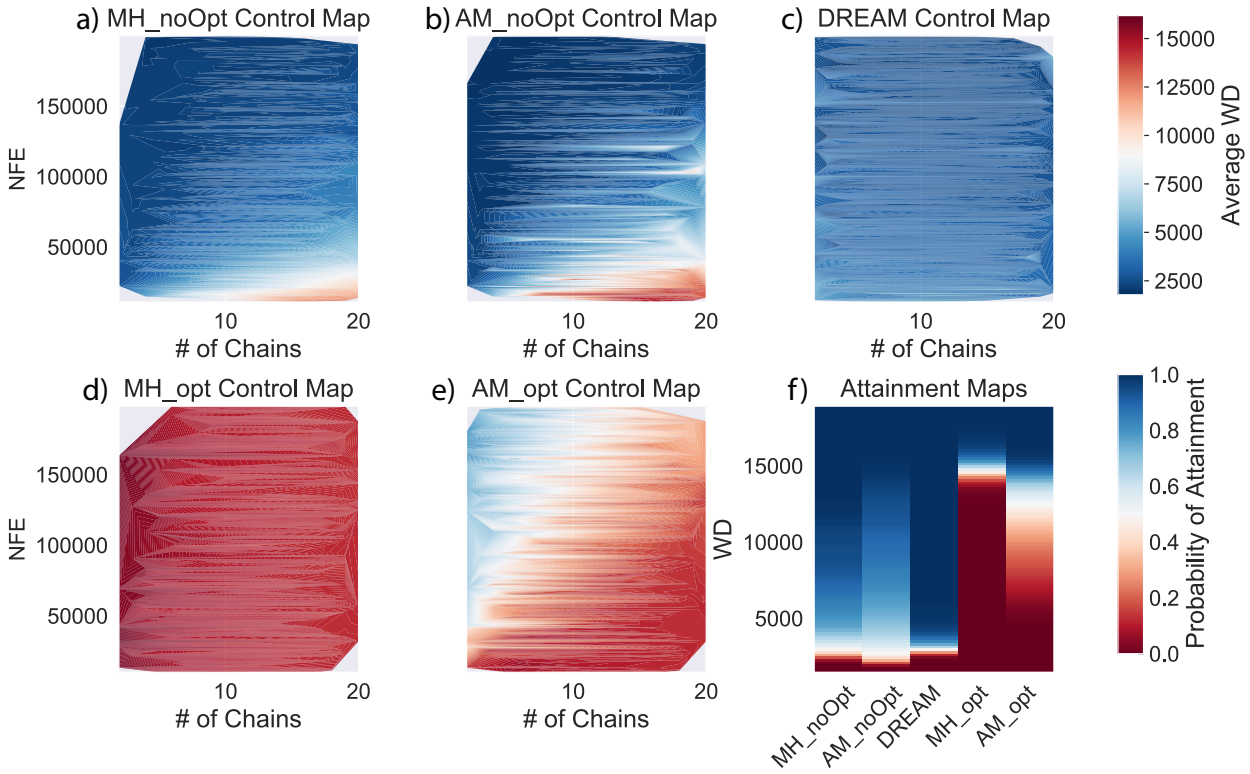

Figure 2.2: (a-e) Control maps for the 100D MVN test problem illustrating the average Wasserstein distance (WD) across random seeds as a function of the number of function evaluations (NFE) and number of chains for (a) MH without optimization, (b) AM without optimization, (c) DREAM, (d) MH with optimization, and (e) MH with optimization. (f) Attainment maps illustrating the probability of attaining different WDs (shown on the y axis) across all seeds and hyperparameters for each algorithm.

tributions and are shown as gray.

When using optimization to initialize chain locations, MH and AM both perform very poorly (Figures 2.2d-2.2e). MH yields high WD across all combinations of chains and NFE (Figure 2.2d). AM's adaptation begins to improve performance at low chain counts (Figure 2.2e), but both algorithms perform much better when optimization is not used to initialize chain locations (Figures 2.2a-2.2b). We investigate the reason optimization performs poorly through additional visual diagnostics of the estimated marginal posterior distributions in Figure 2.3. Without optimization, MH and AM show slower convergence

at higher chain counts (Figures 2.2a-2.2b), indicating that for high-dimensional but unimodal problems, it is better for these algorithms to maximize iterations of a few chains to increase exploitation than to spread them across chains to increase exploration. DREAM (Figure 2.2c) is less hampered by spreading its iterations across chains thanks to the interaction between them, whereby the states of multiple chains are used to propose new chain locations. This insensitivity to chain count results in more controllability and robustness in DREAM's performance. However, its robustness does come at the expense of optimality under certain configurations, as AM's adaptation initially slows convergence, but ultimately results in better posteriors. Consequently, AM with low chain counts is the best choice if one is not computationally limited and knows their problem is unimodal, but DREAM is the best choice if one is more computationally limited.

In addition to finding posteriors that match the true posterior across hyperparameters, we would also like algorithms that do this reliably across random seeds. We investigate this for the 100D MVN problem using attainment maps in Figure 2.2f, which illustrate the probability of attaining different WDs across random seeds. The more blue the attainment map, the higher the probability of attaining low WDs, i.e. the more reliably effective the algorithm is. MH and AM with optimization are shown to be not only inefficient, but unreliable, with a low probability of attaining low WDs. AM without optimization has the highest probability of achieving the lowest WDs (e.g. lowest WD achieved 50% of the time); however, this comes at the expense of increased variability as the probability of attaining near optimal WDs is much lower than both MH and DREAM. We see from the control maps that the higher WDs occur at higher chain counts and lower NFE. DREAM has the highest probability of attaining near-optimal WDs, proving to be not only the most robustly efficient across hyperparameters, but also the most reliably efficient across random seeds.

To verify the patterns seen in the control maps, we illustrate the posterior marginals for a random seed from the hyperparameter nearest each corner and the centroid of the control maps. Figure 2.3 illustrates these marginals for the 50th dimension of the 100D MVN, while SI Figures S3-S4 show them for the 1st and 100th dimensions, respectively, which yield similar conclusions. Across algorithms, as the NFE increases (higher plots), the posterior distributions tend to more closely approach the true posterior (black), with the exception of MH with optimization (light green), which performs poorly across hyperparameters. AM with optimization (light blue) also poorly matches the true posterior, but moves in the right direction as NFE increases for low chain counts. MH and AM without optimization (dark green and dark blue) ultimately come closest to the true posterior, but DREAM (red) performs better when the chain count is high but NFE is low (Figure 2.3e), illustrating its improved robustness at the expense of optimality. Further investigation is needed to understand why using optimization to initialize chains in MH and AM does not direct the search toward the true, single mode. It appears the Nelder-Mead optimization does not converge to the true mode, instead initializing the search in different regions of the space for different chains, and the algorithm takes a long time to explore beyond those estimated modes toward the truth.

Finally, we combine our illustration of reliability and controllability in Figure 2.4, which illustrates the CDF of WD for each hyperparameter. SI Figures S5-S6 show the same for the KLD and GR of the first dimension, respectively. In Figure 2.4, the color of each CDF represents the value of the hyperparameter that most explains variability in WD (yellow=low, purple=high). This hyperparameter is indicated by the variance decomposition shown in Figure 2.4f. The steeper the CDF, the more reliable the algorithm; the closer the CDFs are to 0, the more effective it is; and the more sensitive the algorithm is to NFE (blue), the more controllable it is. Fortunately, NFE is the most influential hyper-

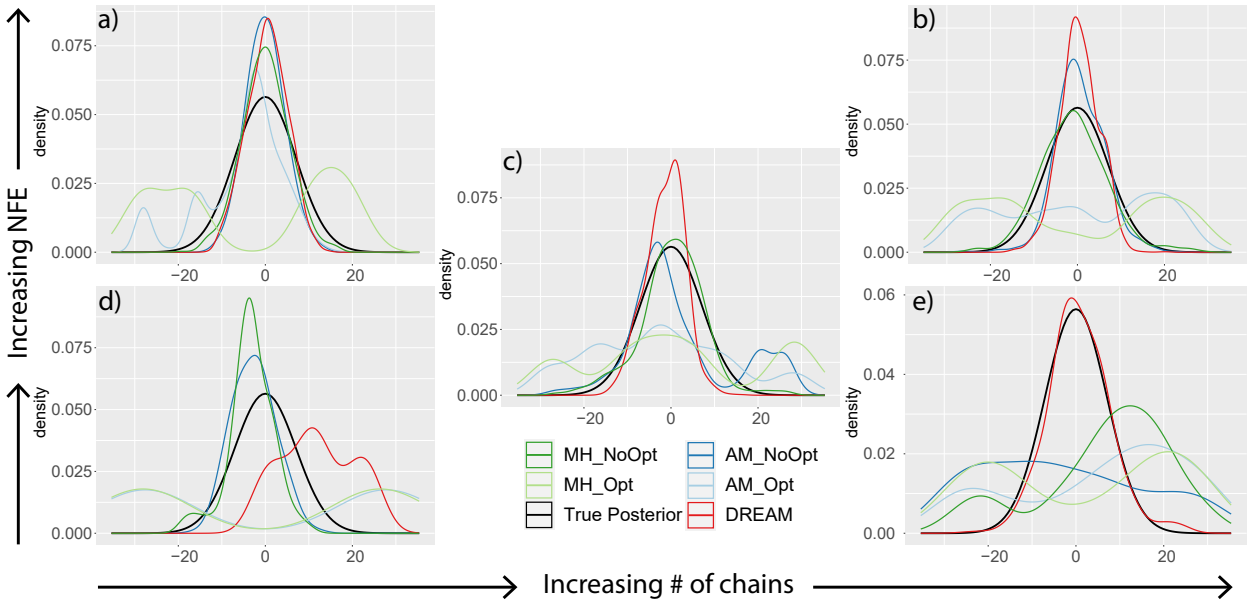

Figure 2.3: Posterior marginals of the 50th dimension of the 100D MVN test problem when using the hyperparameter closest to (a) the least number of chains and the most NFE, (b) the highest number of chains and the most NFE, (c) the median number of chains and the median NFE, (d) the least number of chains and the least NFE, and (e) the most number of chains and the least NFE.

parameter across all algorithms except for MH_{opt}, which is most sensitive to the number of chains (orange). AM_{opt} is also fairly sensitive to the number of chains. This sensitivity to the number of chains, and the multimodal nature of the posteriors estimated by these algorithms in Figure 2.3, suggests that the optimization may be resulting in different chains converging to different modes near their starting locations, which may not be near the true mode.

Analyzing the CDFs, they are steep for nearly all algorithms and hyperparameters, indicating reliability across random seeds. However, they are far more consistently close to 0 for DREAM, which aligns with the findings illustrated by the control maps. Across all algorithms, as the most explanatory hyperparameter increases, the CDFs tend to converge toward lower WD values. This is desirable for the algorithms that are most sensitive

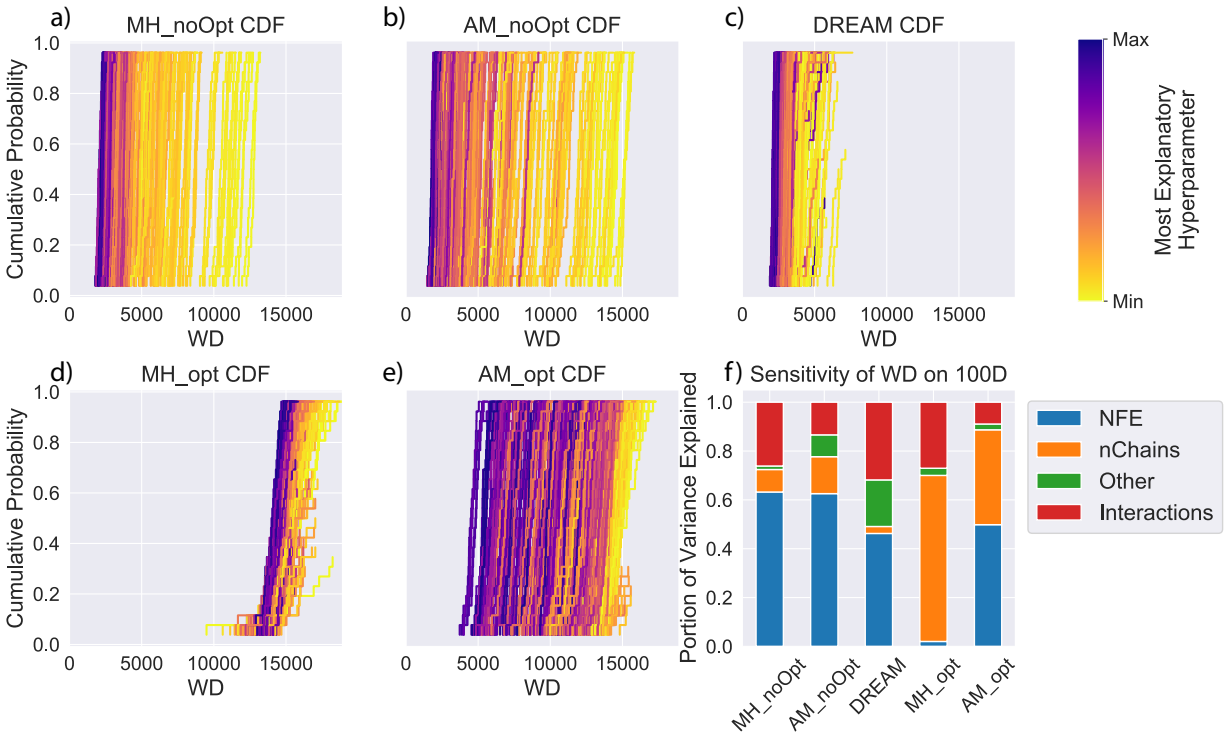

Figure 2.4: (a-e) CDFs of WD across random seeds for each hyperparameter on the 100D MVN test problem. The color of the hyperparameter indicates the value of the parameter to which that algorithm's WD was most sensitive. (f) Decomposition of how much variance in WD is explained by each hyperparameter and their interaction for each algorithm.

to NFE. Among these algorithms (all but MH_{opt}), DREAM exhibits the greatest sensitivity to other parameters beyond the number of chains (green) and interactions between hyperparameters (red). This suggests that although this algorithm is robust across hyperparameters, the additional operators do reduce controllability. This could perhaps be reduced by adapting their values and probabilities throughout the search as has proven successful in multi-objective evolutionary algorithms (Reed et al., 2013), something that could be explored in future work on algorithm development.

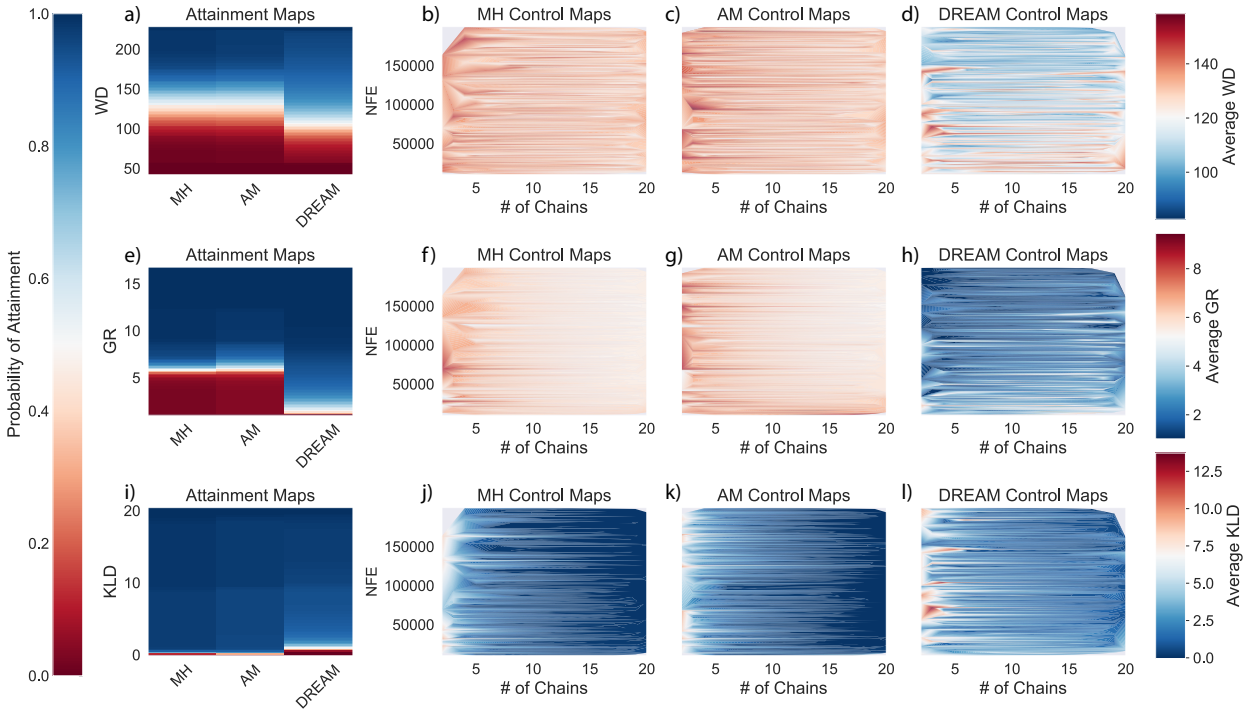

Figure 2.5: Attainment and control maps of each algorithm on the 10D Bimodal test problem based on (a-d) WD, (e-h) GR diagnostic of the first dimension, (i-l) KLD.

2.7.2 Diagnostics on 10D Bimodal Mixed-Gaussian Test Problem

Here, we present our diagnostics assessing the performance of MH, AM, and DREAM on the 10D Bimodal Mixed-Gaussian test problem. Interestingly, unlike for the 100D MVN, the performance of MH and AM was not sensitive to whether optimization was used to initialize chain locations, so we include all hyperparameters together in our visualizations. We hypothesize that the different estimated modes across chains from the Nelder-Mead algorithm was less problematic than for the 100D MVN problem because there is in fact more than one mode on the bimodal problem.

Similar to the 100D MVN test problem, we display the control and attainment maps on the bimodal problem for the three MCMC algorithms in Figure 2.5. However, unlike for the 100D MVN problem, different metrics yielded different conclusions, so we show these

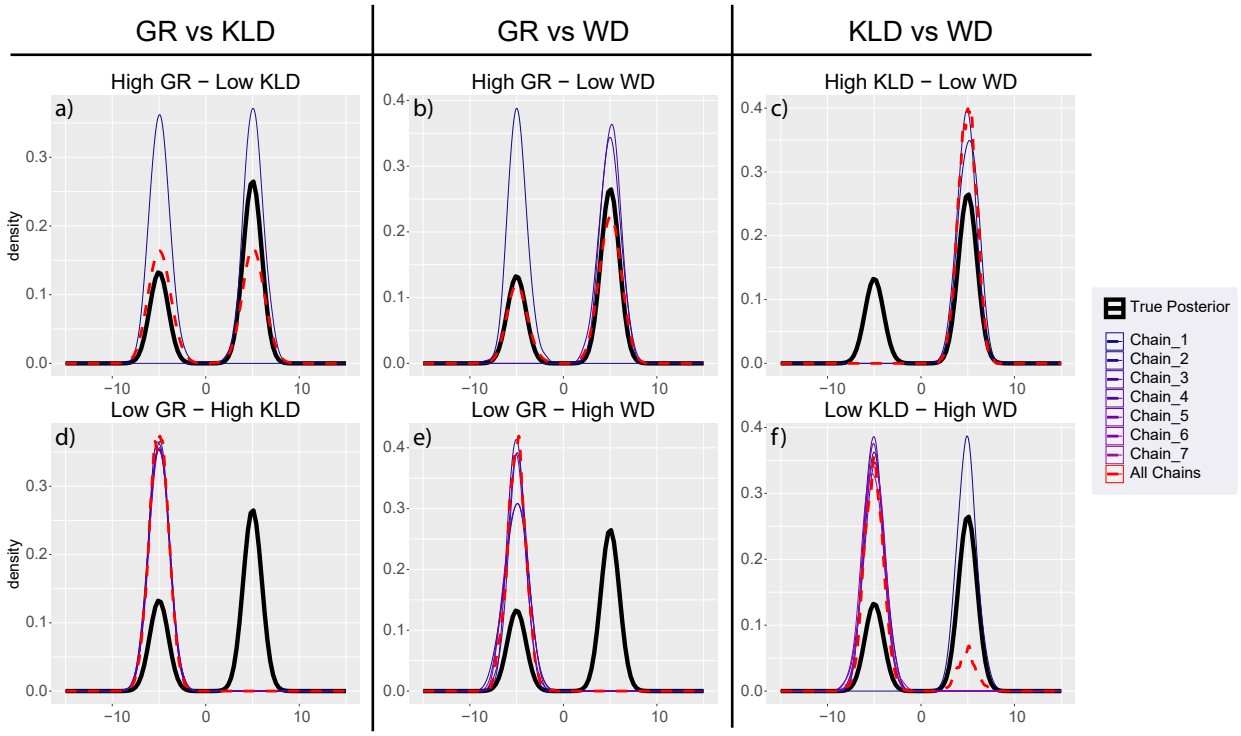

Figure 2.6: Comparison of the estimated MH marginal posterior of the first dimension of the 10D bimodal test problem from individual chains and across chains using select hyperparameter sets with (a) a high GR and low KLD and (d) the reverse; (b) a high GR and low WD and (e) the reverse; and (c) a high KLD and low WD and (f) the reverse.

maps for all three metrics: WD (Figures 2.5a-2.5d), GR of the first dimension (Figures 2.5e-2.5h), and KLD (Figures 2.5i-2.5l).

Examining the control maps, it's clear that DREAM exhibits better performance in achieving lower values of GR and WD (Figures 2.5d and 2.5h) compared to MH (Figures 2.5b and 2.5f) and AM (Figures 2.5c and 2.5g). DREAM also appears more controllable, with low GR values regardless of the NFE and number of chains, and WD improving for higher NFE. On the contrary, WD is poor for MH and AM regardless of the hyper-parameterization, while GR is controlled primarily by NFE. DREAM is also shown to be more reliable on these metrics by the attainment maps (Figures 2.5a and 2.5e), as DREAM has a higher probability of achieving lower values of WD and GR across random seeds

than MH and AM. However, we should note the comparison on GR is not fair since the DREAM chains are not independent, thus potentially providing a false sense of improved convergence.

This false sense of improved convergence is confirmed by the control and attainment maps of KLD, which tell a different story. On this metric, all three algorithms show good performance in achieving low values of KLD, and in fact, MH and AM seem to outperform DREAM in achieving lower KLD values across hyperparameterizations and seeds. This is particularly true when employing a small (near 2) or high (near 20) number of chains, with all algorithms performing similarly at moderate numbers of chains (near 10). One can also see that the number of chains appears to be the controlling hyperparameter for this metric, similar to GR for MH and AM, but different from WD for DREAM.

To understand why conclusions about which algorithms perform best differ under these different metrics, we selected individual hyperparameterizations from the Latin hypercube sample of MH that yielded high values of one metric and low values of another. Figure 2.6 compares the true posterior marginal of the first dimension (black) to the estimated posterior marginals when using the elements of each individual chain of these hyperparameterizations (colored lines), as well when using the elements from all chains (red, dashed line). SI Figures S7-S8 show similar results for the 5th and 10th dimensions.

Analyzing the GR vs KLD plots (Figures 2.6a and 2.6d), we see that individual chains from the LH sample with a low KLD and high GR tend to find only one mode. Since these modes differ across chains, the GR diagnostic is high. However, the proportion of chains finding each mode is similar to those mode's likelihood, resulting in a close approximation to the true posterior across chains, i.e. a low KLD. Conversely, individual chains from the LH sample with a high KLD and low GR each converge to the same mode, resulting in low GR values. However, that mode is the less probable one, resulting in a

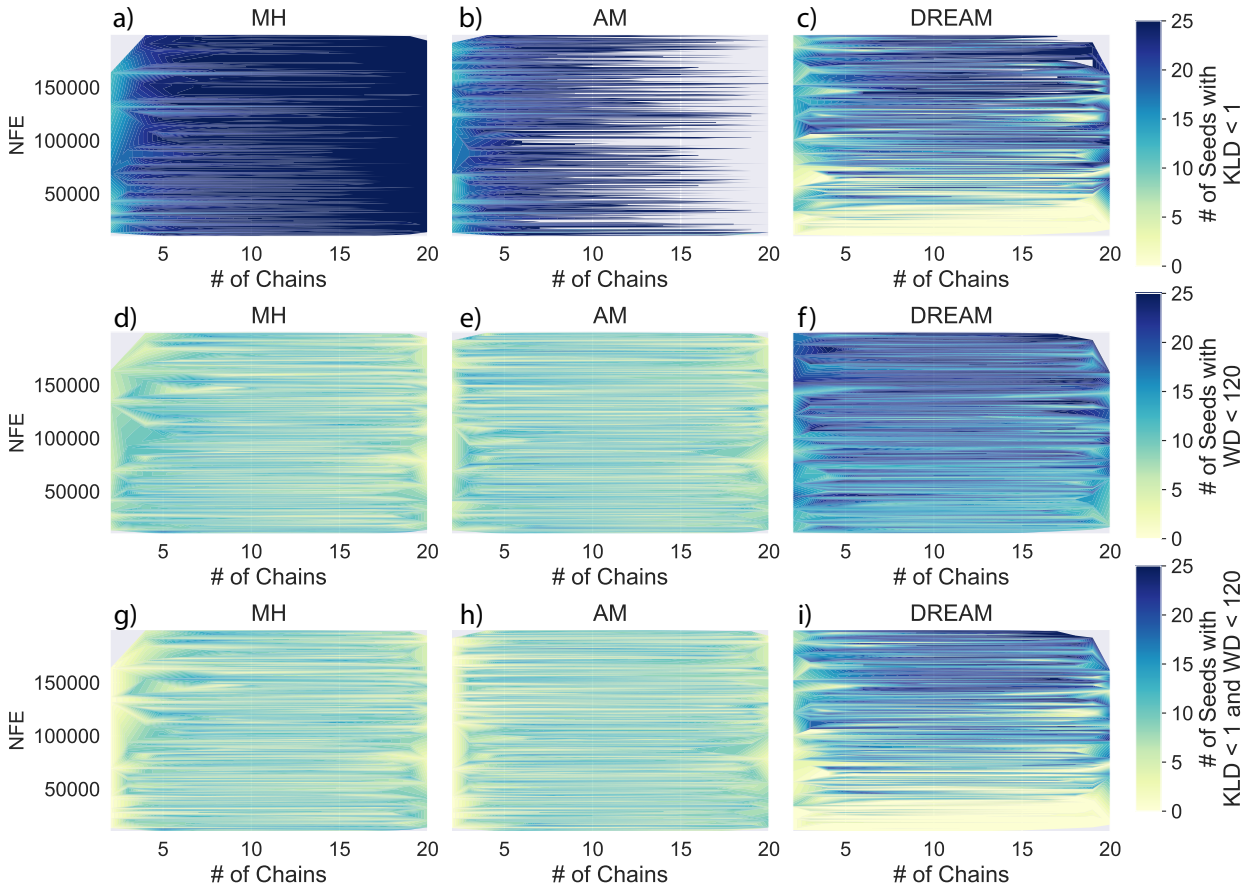

Figure 2.7: Control maps showing the number of seeds (out of 25) of each algorithm that achieved (a-d) KLD<1, (d-f) WD<120, and (g-i) both on the 10D Bimodal test problem.

high KLD. These results confirm what we would expect from theory (Dixit & Roy, 2017). Moving on to the GR vs WD plots (Figures 2.6b and 2.6e), we see similar phenomena. Individual chains from the LH sample with a high GR and low WD each identify different modes of the distribution, but in similar proportions to their likelihood, resulting in a close approximation to the true posterior across them. Conversely, individual chains from the LH sample with a high WD and low GR only detect the less likely mode. These findings again confirm theoretical understandings of these metrics, and illustrate that the GR diagnostic can be a poor metric of convergence on multi-modal problems, raising the question of how to best diagnose convergence on problems with unknown posteriors.

Understanding the disagreement between WD and KLD values requires more investigation. Figure 2.6c reveals that the selected LH sample with a high KLD and low WD only detects the more likely mode of the distribution. This results in a high KLD because the posterior probabilities diverge significantly in the less likely mode. However, the WD is fairly low because the cost of transporting some of the density in the more probable mode to the less probable mode is small. Individual chains from the LH sample with a high WD and low KLD (Figure 2.6f) find different modes, but in near opposite proportions to their true likelihood. This results in a high WD because it is much more costly to transport excess density from the less likely mode to the more likely mode. However, the divergence between the estimated and true posterior is less significant since both modes are found, just not in the right proportions.

Figures 2.6c and 2.6f reveal the importance of considering multiple metrics to assess algorithm performance, as both WD and KLD are capturing important elements of distribution closeness, while failing to capture others. Consequently, in Figure 2.7, we show control maps combining KLD and WD to see which algorithms perform best on both. These figures illustrate the number of seeds yielding KLDs$<$1 (Figures 2.7a-2.7c), WDs$<$120 (Figures 2.7d-2.7f), and both (Figures 2.7g-2.7i), with darker blue indicating a higher number of seeds. Consequently, these maps illustrate all four diagnostic metrics: reliability is indicated by the number of random seeds meeting thresholds of acceptable effectiveness; controllability is illustrated by a lack of noise in reliability, with its value a function primarily NFE or chains; and efficiency is indicated by increased reliability at lower NFE. Consistent with Figure 2.6, we see that MH and AM meet the KLD threshold across more hyperparameterizations than DREAM, particularly at low NFE, while DREAM meets the WD threshold more often. Combining these, we see that MH and AM are able to meet both thresholds more often for low NFE (in about 5-10 seeds for $<$50,000

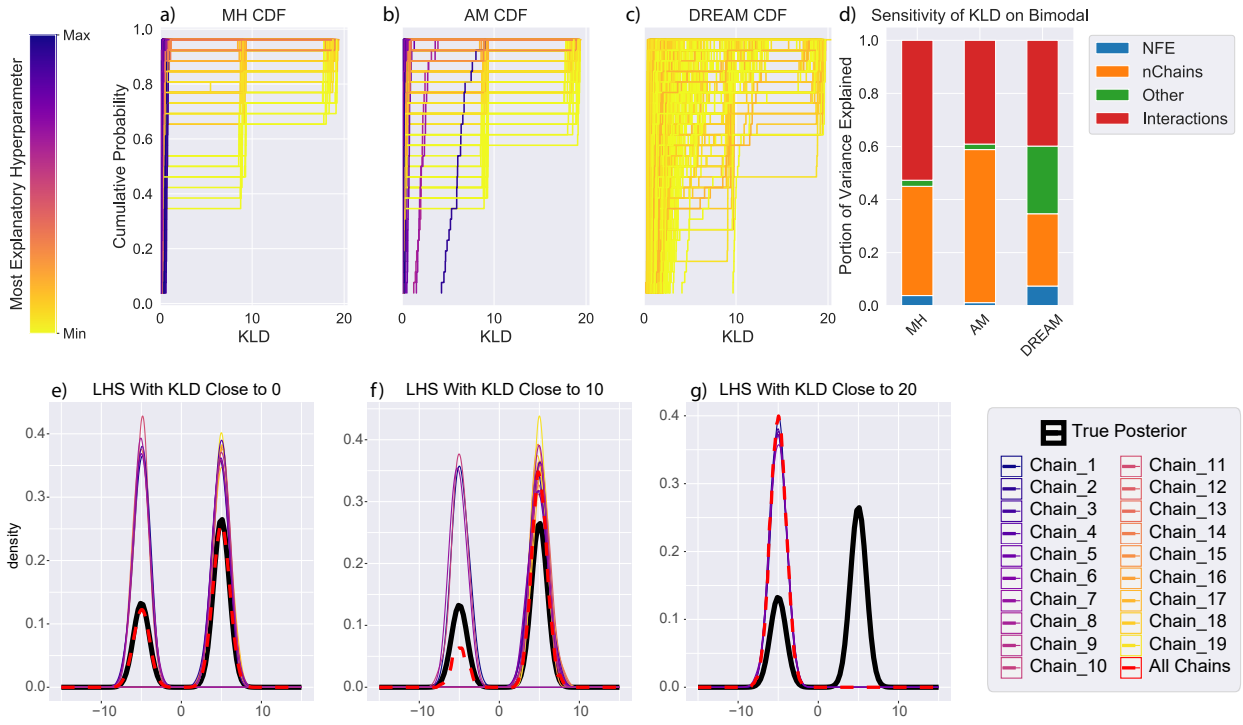

Figure 2.8: (a-c) CDFs of KLDs across random seeds for each hyperparameter. The color of the CDF indicates the value of the hyperparameter to which that algorithm's KLD is most sensitive. (d) Decomposition of how much variablity in KLD is explained by each hyperparameter and their interaction for each algorithm. (e-g) Comparison of the estimated marginal posterior of the first dimension of the 10D bimodal test problem from individual chains and across chains using select hyperparameter sets with (e) KLD near 0, (f) KLD near 10 and (g) KLD near 20.

NFE compared to <5 seeds for DREAM), while DREAM meets both thresholds more often for >50,000 NFE. Thus, for multimodal problems, it may be best to use MH or AM if computationally limited, and DREAM otherwise.

Finally, to hone further in on algorithmic reliability, we show CDFs of the KLD metric for each algorithm in Figure 2.8. We choose the KLD metric since it better captures divergence in probability estimates from the true posterior. We also show CDFs of WD and GR of the first dimension in SI Figures S9-S10. As suspected from the control maps, the sensitivity analysis in Figure 2.8d illustrates that the KLD of all algorithms is primarily

controlled by the number of chains (orange). Consequently, for each hyperparameter, we color the CDF of its WD across random seeds by its associated number of chains, with yellow being low (2 chains) and purple being high (20 chains).

For MH and AM, we can see that low KLDs occur for high chain counts (Figures 2.8a-2.8c), while the trend vs. number of chains is less pronounced for DREAM. This is likely due to DREAM's higher sensitivity to other hyperparameters. We also observe three distinct clusters of KLD values at approximately 0, 10, and 20, particularly for MH and AM. Plotting the marginal posteriors of the first dimension from LH samples of MH with KLDs near these values in Figures 2.8e-2.8g, we see that a KLD near 0 indicates that the algorithm successfully captures both modes in close to perfect proportions, while a value near 10 suggests most chains captured only the more likely mode, and a value near 20 signifies detection of solely the less likely mode. These distinct KLD values make their measure of performance more intuitive than the WD values (see SI Figure S11), indicating it may be a clearer, although less precise performance measure for multimodal problems.

2.8 CONCLUSIONS

This study introduced novel diagnostics for comparing MCMC algorithms in terms of their effectiveness, efficiency, reliability, and controllability via control and attainment maps. This fills an important gap in the MCMC literature, as existing diagnostics solely focus on diagnosing the effectiveness and efficiency of an individual search process, not on diagnosing its consistency (i.e. reliability and controllability) across multiple search processes with different random seeds and hyperparameter configurations. The findings from these new diagnostics have the potential to reduce the time required for hyperparameter tuning. While the diagnostics themselves require a non-trivial computational experiment, they can be performed on computationally cheap test problems with known

posteriors, as done here. Users can then leverage the findings from these diagnostics to choose the most efficient algorithm and corresponding hyperparameter configuration to calibrate a more computationally expensive real-world problem with similar characteristics to the test problems. Existing MCMC diagnostics can then be applied to the single calibration run of the real-world problem to assess convergence of that individual search process. As such, our new diagnostics fill a complementary role to existing diagnostics: our diagnostics can inform the choice of algorithm, while existing diagnostics can then assess convergence using that algorithm.

We illustrate how our diagnostics can reveal which algorithm is most effective, efficient, controllable and reliable by applying them to three widely used MCMC algorithms – MH, AM, and DREAM – on test problems characterized by high dimensionality and bimodality, attributes commonly found in physical systems. The diagnostics offered valuable insights into the performance of these algorithms on different types of problems, as well as on which performance metrics should be used to evaluate algorithms in different contexts. In the context of the high-dimensional (100D) MVN test problem, our analysis revealed a notable sensitivity of MH and AM to the binary optimization hyperparameter, ironically resulting in sub-optimal performance when using optimization to initialize chains. While, MH and AM without optimization exhibited improved convergence and closer alignment with the true posterior distribution, these algorithms needed significant NFE to do so, especially when using multiple chains. In contrast, DREAM consistently demonstrated strong performance, as evidenced by both control and attainment maps. For the 10D Bimodal Mixed-Gaussian test problem, DREAM continued to perform well, achieving lower WD and GR values compared to MH and AM. However, when considering the KLD metric, MH and AM displayed competitive performance, particularly in scenarios involving a smaller number of chains.

These conflicting findings across performance metrics on the bimodal problem analysis revealed intricate trade-offs between WD, GR, and KLD values, shedding light on their strengths and weaknesses in assessing algorithm performance. Critically, it was highlighted that low GR values do not necessarily indicate convergence, just consistent variance across chains. This is particularly uninformative if the chains are consistent only because they are not independent, but communicate as in DREAM. In reality, the chains may represent consistently poor approximations of the true posterior. Consequently, multiple metrics could be used to assess MCMC convergence on problems with unknown posteriors, and further research is needed on developing alternative convergence metrics for such problems. For algorithm development, test problems with known posteriors could be used for performance assessment to avoid these biases. When the true posterior is known, WD and KLD represent better measures of performance, but capture different aspects of that performance. KLD is a better measure of how close the estimated probabilities of different parameter values are to their true probabilities, while WD is a better measure of how close those two distributions are in parameter space. For multi-modal problems, KLD may then be more appropriate.

Finally, the analysis in this paper also points to new avenues of research. First, an important area of future research is in applying these diagnostics to assess not only efficiency in the mean estimate of the posterior, but in the variance of that estimate. As discussed in the introduction, because of the Markovian property of MCMC algorithms, consecutive elements in the chain are not independent. This autocorrelation can reduce the effective sample size of the chain, thereby increasing the Monte Carlo error, and corresponding standard error of the posterior mean estimate. Future work could investigate how this uncertainty changes across hyperparameter configurations and random seeds by making control and attainment maps of the standard error of the posterior mean esti-

mate across chains. Such analysis could also include thinning of the chain as a hyperparameter, whereby only every k elements in the chain are retained, to see what impact that hyperparameter has on the standard error.

Second, the finding that while DREAM was robust, it did exhibit greater sensitivity to its additional hyperparameters suggests DREAM's controllability could be improved by adapting its probability of using its different proposal operators based on their success in proposing new chain locations that are accepted. This idea comes from the observation in the literature that performance of multi-objective evolutionary algorithms can be improved by adapting the probability of using different operators based on their success in generating non-dominated solutions (Hadka & Reed, 2013). In testing such proposed advancements for MCMC, performance metrics such as KLD and WD could be used to evaluate performance on known test problems. The visual diagnostics proposed here can then be used to evaluate and inform the design and hyperparameterization of such new MCMC algorithms.

2.9 CODE AND DATA

We provide the scripts written to generate synthetic data and do the analysis in this study in our Zenodo repository.[1] Code development history may also be found on our GitHub repository.[2]

[1] https://zenodo.org/records/10433119
[2] https://github.com/hosseinkavianih/New-Diagnostics-Assessment-For- MCMC

CHAPTER 3

Tackling Complexity: EMODPS vs. DDPG for Multi-Objective Reinforcement Learning

3.1 ABSTRACT

Reinforcement learning (RL) is used frequently to optimize control rules for engineering systems. When these systems have multiple conflicting objectives, RL becomes more challenging. Applicable multi-objective optimization approaches differ depending on whether a value-based or policy-based RL method is used. In value-based methods, the value needs to be composed of a weighted sum of the multiple objectives. A gradient-based solver then seeks to find the actions that maximize the value for every state and stage. In policy-based RL methods, the optimizer simply needs to find the best parameters of a policy that maps the states to actions. This can be achieved using multi-objective evolutionary algorithms (MOEAs). However, the literature lacks comparative studies identifying whether it is better to find a non-dominated set of alternative control rules using policy-based RL with an MOEA, or value-based RL with a gradient-based solver and different weights on the objectives. For the first time, this study benchmarks Evolutionary Multi-Objective Direct Policy Search (EMODPS), a policy-based RL method, against Deep Deterministic Policy Gradients (DDPG), an actor-critic RL method that includes both policy optimization and value-approximation. The system consists of two storage ponds for urban stormwater that each have an orifice at the bottom that can control the rate of outflow from the ponds for flood control. The objectives are to minimize flooding upstream (at the storage ponds) and downstream. Our findings reveal that EMODPS generates more robust trade-off solutions than DDPG with simpler hyperparameter tuning, making it a preferred choice for real-world, multi-objective applications.

3.2 INTRODUCTION

Many complex control problems have conflicting objectives (Zaniolo et al., 2021; Vamplew et al., 2011; Liu et al., 2014). There are two main approaches for designing control rules for such systems (Liu et al., 2014). The first is to find a single control rule that best balances all the objectives. Methods that attempt to achieve this include the weighted sum method that weights the objectives based on their importance, elicited from stakeholders (Konak et al., 2006), the sequential method that optimizes for each objective in sequence by priority (Nakayama et al., 2009), or the min-max method that seeks to maximize the worst-performing objective (Lin, 2005). The second approach to multi-objective optimization does not seek to find one dominant solution in hopes that it best balances the conflicting objectives; rather it finds a set of policies that are "non-inferior" or "non-dominated", meaning between any pair of solutions, improvement on one objective must come at the expense of performance on at least one other objective. These policies, which collectively form the "Pareto front" (Pareto, 1964), represent alternative solutions stakeholders can choose from (Vamplew et al., 2008; Jin et al., 2001). Finding a Pareto front of alternative solutions is often preferable in complex systems, as it is difficult to know *a priori* how to formulate a single objective function to best capture stakeholders' preferences (Vamplew et al., 2008). In fact, stakeholders often choose different policies from the Pareto front than their elicited preferences would imply (Hobbs et al., 1992).

If one seeks to find a Pareto front of control rules, the best approach is typically to apply multi-objective reinforcement learning (MORL) (Hayes et al., 2022). RL is a subcategory of machine learning in which agents are trained to make sequential decisions that maximize the "rewards" they receive through interacting with the environment. The rewards (or penalties) the agents receive based on their actions allow them to learn optimal policies through trial and error. The objective function in RL is to maximize cumu-

lative rewards over time by navigating through the environment spatially and/or temporally (Ding et al., 2020). There are various RL categories that can be used for control tasks. These can be categorized into model-based and model-free methods (Zhang & Yu, 2020). Model-based methods utilize knowledge on the transition probability from state s_t at stage t to state s_{t+1} at stage $t+1$ given an action a_t. These methods can be efficient if the model is known, but slow if it has to be learned, or inaccurate if the given model is a poor representation of the true world (Zhang & Yu, 2020). As such, model-free methods are growing increasingly popular.

Within model-free RL, approaches can be categorized as either value-based, such as Q-learning (Watkins & Dayan, 1992; Nachum et al., 2017), or policy-based, such as Direct Policy Search (DPS) (Rosenstein & Barto, 2001; Giuliani et al., 2016). Value-based methods estimate $Q(s_t, a_t)$, the expected present and future value Q at stage t associated with taking action a_t from state s_t under the optimal policy π^* (Watkins & Dayan, 1992). A gradient-based solver then seeks to find the actions that maximize the value for every state and stage (i.e. the optimal policy π^*) based on the current estimate of the Q function. This estimate is updated as actions are taken and rewards received. Policy-based methods do not require estimating the value function Q; rather they simply estimate the optimal policy $a_t = \pi^*_\theta(s_t)$, defined as a simple state-action mapping determined by parameters θ (Rosenstein & Barto, 2001). The best parameters θ can be estimated iteratively based on rewards received from implementing the current best estimate of the optimal policy or using simulation-optimization based on the cumulative reward at the end of the simulation (Giuliani et al., 2017). Either gradient-based or heuristic optimization algorithms can be used to optimize the policy function. Some RL methods, called actor-critic methods, include both value-based and policy-based elements (Konda & Tsitsiklis, 1999), in which the "actor" learns the optimal policy, while the "critic" learns the value of the

actions being taken.

When optimizing control rules for multiple objectives using value-based methods, the value function needs to be composed of a weighted sum of those objectives to compute the gradient. This is true for the reward function when using gradient-based solvers with policy-based methods as well. Therefore, if one wants to obtain a Pareto set of alternative non-dominated control rules, they have to repeat the optimization process for multiple combinations of weights on the component objectives (Chankong & Haimes, 2008). However, when using heuristic optimization methods with policy-based RL, multi-objective evolutionary algorithms (MOEAs) can be used to find a Pareto set of non-dominated policies in one optimization (Coello, 2018). MOEAs can also capture non-convex regions of the Pareto front that weighting methods may miss (Chiandussi et al., 2012). However, MOEAs can be slow to converge, questioning whether it is better to perform MORL using policy-based methods with MOEAs, or value-based methods with a gradient-based solver and different weights on the objectives.

In this study, we seek to answer this question by comparing Evolutionary Multi-Objective Direct Policy Search (EMODPS) (Giuliani et al., 2016), a policy-based RL method coupled with an MOEA, and Deep Deterministic Policy Gradients (DDPG) (Lillicrap et al., 2015), an actor-critic method using alternative weights on competing objectives to represent the reward function. DDPG and EMODPS have been used in recent years to optimize complex control systems (Yan & Xu, 2020; Zatarain Salazar et al., 2016). Both approaches have known pros and cons, but it is not clear which is most effective for MORL.

DDPG has been known for its efficiency on problems with a continuous action space (Duan et al., 2016; Ding et al., 2020). However, it has multiple issues that hamper its ability on real-time control (RTC) problems. First, DDPG suffers from the exploration-exploitation trade-off where it may converge to a sub-optimal solution due to over-

exploitation or, conversely, fail to converge due to over-exploration, resulting in a noisy action set (Vinyals et al., 2017; Hao et al., 2023). Additionally, hyperparameter tuning in DDPG is often laborious due to the need for two distinct sets of hyperparameters: one for the reinforcement learning components, such as the actor and critic learning rates, discount factor, and exploration noise, and another for the deep learning components describing the value and policy functions, including the number of hidden layers, hidden units, and choice of optimizer. Tuning all these parameters can be very time-consuming and computationally expensive (Liessner et al., 2019; Ashraf et al., 2021; Kiran & Ozyildirim, 2022). While transfer learning can be used in DDPG to leverage previously learned policies or features to accelerate training in new but related tasks (Li et al., 2022), it is not always straightforward, and previously learned policies may not be available. There are workarounds for increasing the probability of convergence without transfer learning, such as using a replay buffer (Hou et al., 2017) or adding noise (Plappert et al., 2017) to actions to prevent premature convergence, but applying these techniques and tuning hyperparameters for new control systems remain challenging.

EMODPS on the other hand, is effective in deriving a diverse Pareto set by coupling policy search with an MOEA, avoiding modeling the value function (Zaniolo et al., 2021; Zatarain Salazar et al., 2017). However, there are challenges with applying it as well. First, prior definition of policy architecture is not always easy and requires trial and error experiments or analytical intuition (Zaniolo et al., 2021). This is important since it defines the search space for the control policies and therefore requires the user to fine-tune this to the problem. Another issue is a tendency for the algorithm to over-fit the policy parameters during the simulation and not generalize well for the unseen data (Giuliani et al., 2016; Zaniolo et al., 2021). Finally, the exploration-exploitation tradeoff applies to MOEAs as well (Herrera et al., 1996).

An effective way to evaluate the performance of alternative methods to finding a Pareto set of a multi-objective control rules for engineering systems is to benchmark them on real-world problems. Example applications in the literature include control of autonomous vehicles (Yan et al., 2022), smart grids (Sauerteig & Worthmann, 2020), supply management (Reich et al., 2021), and water resources management (Giuliani et al., 2017; Tabas & Samadi, 2024; Giuliani et al., 2021). Within water resources management, a classic problem is stormwater control, which can support flood protection, pollution control, and environmental flows. Several studies have proposed using value-based RL methods to mitigate flooding in stormwater systems. Mullapudi et al. (2020) leveraged deep Q-learning to train agents to control valves in a distributed urban water system to improve flood control rules. Saliba et al. (2020) proposed DDPG for real-time stormwater control for flood mitigation, addressing the challenge of uncertain data. Bowes et al. (2021) similarly coupled DDPG with a stylistic urban storm water system and compared the results with model predictive control (MPC) and rule-based control (RBC). Bowes et al. (2022) then expanded on this to include multiple objectives for flood control and water quality, but utilizing a single reward function encompassing these two goals. Consequently, in this and all prior studies, solutions were represented as a single policy rather than showing the Pareto front of different policies and their trade-offs. This is one of the research gaps that we aim to address in our paper by utilizing DDPG to obtain a Pareto front of alternative control rules.

Another gap we seek to fill is to introduce EMODPS for urban stormwater control. Although many studies have explored value-based or actor-critic RL methods for stormwater control, there has been no research implementing EMODPS as a strictly policy-based RL tool for such systems. In this study, we integrate EMODPS with a hypothetical urban stormwater system, inspired by the work of Bowes et al. (2021), and conduct a com-

parative analysis with DDPG, as an actor-critic method for MORL that has been used for stormwater control before, but only for single-objective problems. To the best of our knowledge, such a methodological comparison has not been conducted previously, and we believe it can shed light on the strengths and weaknesses of these approaches for real-time control of systems with conflicting objectives. To achieve this, we design a stylistic stormwater control problem featuring conflicting flooding objectives in different areas of the stormwater system and assess the Pareto fronts derived by EMODPS and DDPG.

Our paper is organized as follows. Our methods are described in Sections 3.3-3.5. Section 3.3 briefly describes the RL algorithms we study and compare, Section 3.4 outlines the case study for this comparison, and Section 3.5 introduces the computational experiment we performed for the comparison. We display the results of the optimization experiments and the analysis in Section 3.6. Finally, we discuss our conclusions about which method provides better trade-offs in our stormwater system, and note areas for future work in Section 3.7.

3.3 ALGORITHMS

In this section, we describe EMODPS and DDPG as the two reinforcement learning algorithms examined in our study. These algorithms serve as powerful tools for addressing complex control problems in various domains, ranging from robotics and autonomous systems to finance and environmental systems.

3.3.1 EMODPS

EMODPS has two main elements: functions describing control policies and evolutionary algorithms that optimize the parameters of those functions. The main strength of EMODPS is its ability to handle multiple conflicting objectives simultaneously. This ca-

pability is crucial in stormwater systems management, where decisions must involve diverse trade-offs such as minimizing flood risk, minimizing water quality impacts, and maintaining natural flows (Bowes et al., 2022). EMODPS has three steps: parameterization, simulation, and optimization (Koutsoyiannis & Economou, 2003). It parameterizes control rules within a family of functions, simulates operations with those rules, and then couples the simulation with a multi-objective evolutionary algorithm (MOEA) to optimize the parameters of the control rules in order to achieve better values of the objective functions in the simulation.

The parameterization step involves finding parameters θ of a policy π that maps states to actions: $a_t = \pi_\theta(s_t)$. To parameterize our control policies, we represent π using non-convex Gaussian Radial Basis Functions (NCRBFs) with an additional constant. Gaussian RBFs are a type of neural network with Gaussian activation functions. They are universal approximators, making them suitable for representing control policies in multi-objective systems. Recent studies have shown that Gaussian RBFs offer advantages over Artificial Neural Networks (ANNs) with hyperbolic tangent activation functions in terms of their usability, simplicity, and generalization capabilities (Giuliani et al., 2016; Zaniolo et al., 2021).

Equation 3.1 shows the general form of π using non-convex RBFs with a constant, where the parameters to be optimized are $\theta = [\alpha^k, c_{i,j}, b_{i,j}, w_i^k]$:

$$a_t^k = \alpha^k + \sum_{i=1}^{N} w_i^k \exp\left(-\sum_{j=1}^{S} \frac{(s_{t,j} - c_{i,j})^2}{b_{i,j}^2}\right). \tag{3.1}$$

Here a_t^k represents the action to be taken from the k-th agent at stage t, normalized between 0 and 1, $s_{t,j}$ represents the value of the j-th of S state variables at stage t, normalized between 0 and 1, and N denotes the total number of Gaussian NCRBFs. The parameters representing the decision variables are w_i^k, the weight associated with the i-th RBF for the

k-th agent, $c_{i,j}$ and $b_{i,j}$, the center and radius of the i-th RBF associated with the j-th state variable, and α^k, an additional constant term for the k-th agent. For convex RBFs, the weights are constrained to sum to 1, but we relax this constraint for the NCRBFs. Note that the number of decision variables depends on the number of RBFs, N. This is the one hyperparameter for EMODPS that needs to be tuned when using NCRBFs, but other tuning components include the state variables and the parametric form of the policy (e.g. using a functional form other than NCRBFs, such as convex RBFs or ANNs with different activation functions).

For the simulation component of EMODPS, we simulate control rules using the PySWMM package in Python. PySWMM is a Python wrapper for running the Stormwater Management Model (SWMM) in a Python environment (McDonnell et al., 2020). More details are provided in Section 3.4.

Finally, we employ the Borg MOEA as the optimization tool for EMODPS using its single-master parallel Python wrapper (Hadka & Reed, 2013). Borg is an adaptive MOEA that has consistently shown superior performance on nonlinear, discontinuous, multi-objective engineering problems (Reed et al., 2013; Zatarain Salazar et al., 2016). Through multi-objective optimization, Borg identifies a Pareto-optimal set of solutions that are non-dominated relative to each other, meaning no single solution outperforms any other across every objective. To achieve this, Borg utilizes adaptive mutation and crossover operators, adaptive population sizes, and epsilon-dominance archiving, among other features.

3.3.2 DDPG

The DDPG algorithm is an actor-critic reinforcement learning method designed for environments with continuous action spaces (Lillicrap et al., 2015). It combines elements of

deep learning for function approximation and the deterministic policy gradient for stable and efficient learning of the policy and value functions. The key components of the DDPG algorithm include an actor network that learns the policy function and a critic network that estimates the value function, each of which is represented by a deep neural network. The actor network determines the best action to take in a given state, while the critic network evaluates the actions taken by the actor.

The actor network describes the current policy of the agent, $\pi_\theta(s_t)$, The behavioral policy that gives the actual actions taken by the agent adds noise to this network:

$$a_t = \pi_\theta(s_t) + \mathcal{N}(0, \sigma^2) \tag{3.2}$$

where σ^2 is the variance of normally distributed exploration noise. The exploration noise helps the agent explore the environment more effectively. In our implementation, we utilize the Ornstein-Uhlenbeck method for generating noise (Uhlenbeck & Ornstein, 1930). This method produces auto-correlated noise, preventing the noise from canceling out the overall dynamics of the system. Initially, the variance of the noise is set to 0.2, then it gradually decreases to 0.01 over the course of training (note: $a_t \in [0, 1]$). This gradual decrease encourages exploration in the early stages, then transitions to exploitation as the learning process progresses.

The critic network is represented as $Q_\phi(s_t, a_t)$ where ϕ are parameters of the value function. Both the actor and critic networks are updated throughout the simulation based on rewards received from utilizing the policy function. We utilize a replay buffer to store past experiences encountered during interactions with the environment. The replay buffer serves two primary purposes. First, it acts as a memory pool where experiences are stored, allowing the algorithm to sample mini-batches of experiences randomly during training. This random sampling breaks the temporal correlation in consecutive

experiences and significantly improves sample efficiency and stability. Second, DDPG is an off-policy algorithm, which means it learns from experiences generated by a behavior policy that includes noise (equation 3.2) rather than the current policy being learned (that without noise, $\pi_\theta(s_t)$). The replay buffer facilitates off-policy learning by providing a diverse set of experiences for training both the actor and critic networks (Hou et al., 2017).

The DDPG algorithm further leverages target networks to stabilize training through delayed updates of target values. This strategy helps to reduce variance in performance and enhances the convergence of the learning process. The parameters of the target actor and target critic networks (θ' and ϕ', respectively) are initialized as the parameters of the current actor and critic networks (θ and ϕ, respectively), as shown by equations 3.3-3.4).

$$\theta' \leftarrow \theta \tag{3.3}$$

$$\phi' \leftarrow \phi \tag{3.4}$$

The parameters of the current actor network are updated in an off-policy way through gradient descent with batches of experience (denoted as B):

$$\nabla_\theta J(\theta) = \frac{1}{B} \sum_{i=1}^{B} \nabla_a Q_\phi(s_i, a)|_{a=\pi_\theta(s_i)} \nabla_\theta \pi_\theta(s_i). \tag{3.5}$$

where $J(\theta)$ denotes the average value of the current policy across batches. Following the update of the actor network, the critic network is updated to close the distance between performance of the current and target actor networks.

The target actor network is denoted as $\pi_{\theta'}^{\text{target}}(s_t)$, and the target critic network is denoted as $Q_{\phi'}^{\text{target}}(s_t, a_t)$. The target Q-value, y_t, can be determined using the Bellman Equa-

tion:

$$y_t = r_t + \gamma Q_{\phi'}^{\text{target}}(s_{t+1}, \pi_{\theta'}^{\text{target}}(s_{t+1})) \tag{3.6}$$

In this context, r_t is the reward function at time step t and γ is the discount factor. The critic network is updated by minimizing the temporal distance error between the target and current Q-value, penalized by a regularization term determined by the hyperparameter λ:

$$\mathcal{L}_{\text{critic}} = \frac{1}{2}\Big[y_t - Q_\phi(s_t, a_t)\Big]^2 + \lambda \sum_i \phi_i^2 \tag{3.7}$$

Finally, after updating the parameters of the current actor and critic networks as outlined above, the parameters of the target actor and critic networks are updated as a weighted average of the parameters in the current and target networks:

$$\theta' \leftarrow \tau\theta + (1-\tau)\theta' \tag{3.8}$$

$$\phi' \leftarrow \tau\phi + (1-\tau)\phi' \tag{3.9}$$

In this context, τ is the update rate parameter that controls the rate at which the target actor and critic networks are updated. This parameter is usually small to achieve a soft update and can be treated as a hyperparameter.

To ensure optimal performance of our neural network, we conducted extensive hyperparameter tuning prior to model training. This consisted of an initial grid search across three values each of learning rates and batch sizes, followed by iterative tuning on less sensitive hyperparameters (see Table 3.1). The learning rates of the actor and critic networks are critical hyperparameters that control the step size during gradient descent optimization (∇ in equation 3.5). By fine-tuning the learning rate, we aim to strike a balance

between training convergence speed and stability, crucial for efficient learning in DDPG. If the actor network updates faster than the critic network, the estimated Q-value may become inaccurate, as it would be based on outdated policies (**?**). Given this, we selected 0.0001, 0.0005, and 0.001 as learning rates to test for the actor network, and 0.001, 0.005, and 0.01 for the critic network to perform the grid search. We found the best performance with learning rates of 0.0001 for the actor and 0.001 for the critic. Additionally, we explored batch sizes of 8, 16, and 32 to determine the optimal number of samples used in each training iteration. A batch size of 8 achieved the lowest temporal distance error during the grid search.

Table 3.1: Tuned Hyperparameters for RL and Deep Learning Components

Hyperparameter	**Value**
RL Hyperparameters	
Actor Learning Rate	0.0001
Critic Learning Rate	0.001
Discount Factor, δ	0.99
Batch Size, B	8
Exploration Noise	$\sigma_{\text{initial}} = 0.2$, $\sigma_{\text{final}} = 0.01$
Replay Buffer Size	1,500,000
Deep Learning Hyperparameters	
Number of Hidden Layers	2
Hidden Units	32 (first layer) and 16 (second layer)
Activation Function of Non-output Layers	Leaky ReLU ($\alpha = 0.01$)
Output Layer Activation Functions	Sigmoid (actor) and Linear (critic)
Optimizer	Adam
Regularization	L2-norm with $\lambda = 0.01$

In addition to the learning rate and batch size, we explored other hyperparameters, primarily related to the structure of the actor and critic networks, via trial and error. We summarize the tuned hyperparameter values in Table 3.1. As seen in the table, our actor

and critic neural networks each comprise three layers: an input layer with three neurons, followed by two hidden layers with 32 and 16 neurons, respectively, and an output layer. The input and hidden layers use a Leaky ReLu activation function with hyperparameter $\alpha = 0.01$:

$$f(x) = \max(x, \alpha x) \tag{3.10}$$

where x is the input and $f(x)$ the output. For the actor network, the output layers consist of two neurons, representing continuous actions between 0 and 1, which were captured by sigmoid activation functions. For the critic network, the output layer consists of a single neuron, representing the value associated with a given action and state, which was captured by a linear activation function. This architecture was designed to capture complex relationships and make precise continuous action predictions (between [0,1]) within the DDPG framework.

3.4 CASE STUDY

In order to evaluate the performance of EMODPS and DDPG in a multi-objective setting, we present a stylized stormwater system inspired by the work of Bowes et al. (2021) (see Figure 3.1). This system is designed to replicate the environmental conditions observed in an urban catchment located in Norfolk, Virginia, USA. It consists of two subcatchments, two storage ponds acting as storage units (SP1 and SP2) for flows from those subcatchments, and a network of connected pipes leading from the ponds to the system outfall. The two subcatchments have the same area, width, slope, impervious cover, and roughness coefficients. The two ponds have the same maximum depth, initial depth, and rating curve, but SP1 is at a 5 meter higher elevation. The key control features of this system are orifices at ponds 1 and 2, denoted O1 and O2, that allow us to control the rate at which

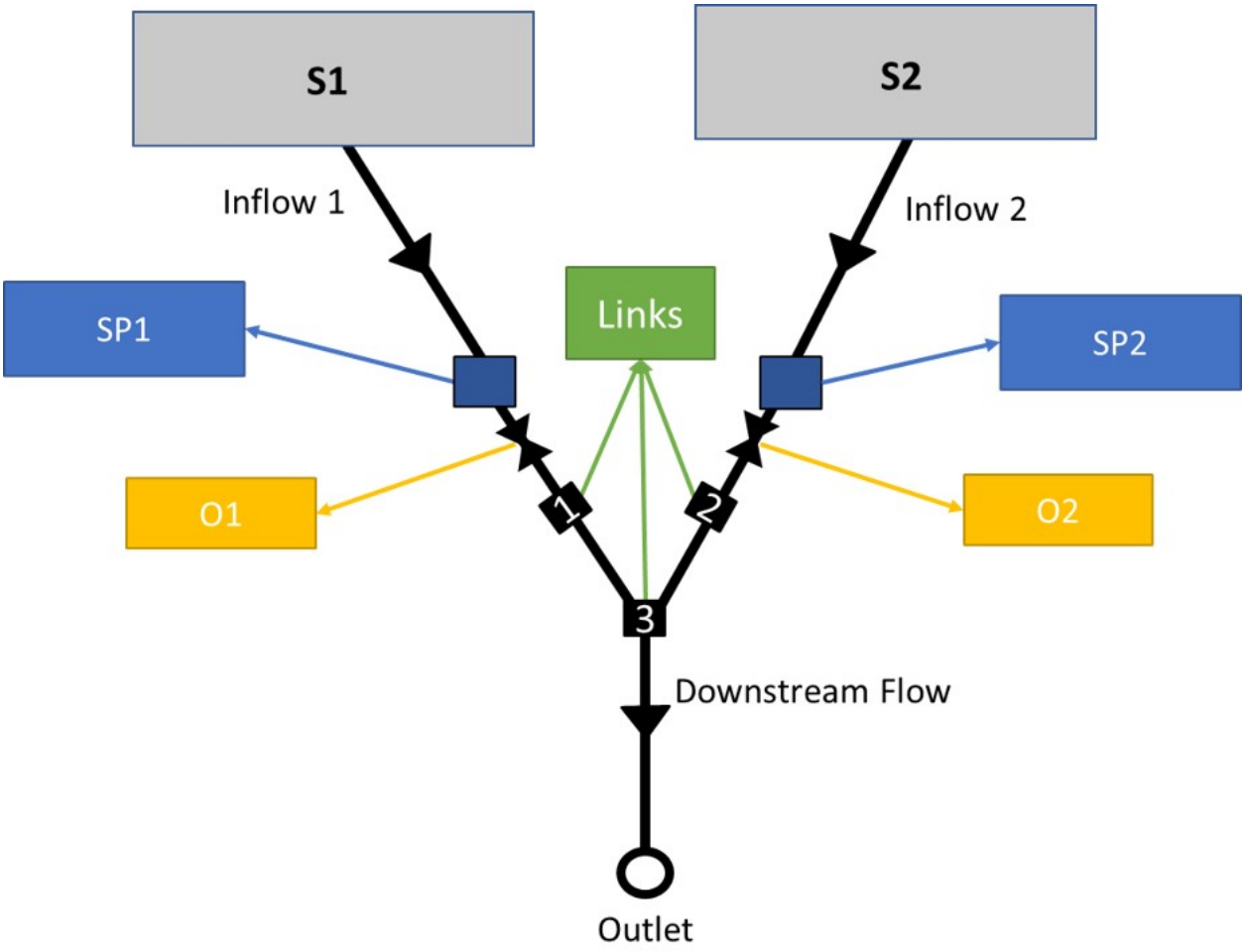

Figure 3.1: Schematic of the stylized stormwater system used in this study.

water is released from the storage ponds. This control over the discharge rate plays a crucial role in managing flood volumes upstream (at the storage ponds) and downstream, optimizing the system's overall performance.

The objectives of this study are: 1) to minimize total upstream flooding, quantified as the sum of overflows at storage ponds 1 and 2; and 2) to minimize total downstream flooding, quantified as flooding at links 1 and 2. These objectives can be formulated mathematically as:

$$\text{Minimize: } O_1 = (Overflow_{\text{SP1}} + Overflow_{\text{SP2}}) \tag{3.11}$$

$$\text{Minimize: } O_2 = (Flooding_{\text{Link 1}} + Flooding_{\text{Link 2}}) \tag{3.12}$$

These objectives are computed over a simulation of historical rainfall from the Norfolk airport.

3.5 COMPUTATIONAL EXPERIMENT

To assess the performance of EMODPS and DDPG, a thorough computational experiment was conducted. For a fair comparison, optimization with DDPG and EMODPS each was done using the same total number of computational service units (SUs). The simulations of the stormwater system were conducted using the U.S. Environmental Protection Agency's Stormwater Management Model (SWMM), version 5. Control rules were implemented using the PySWMM package in Python, as depicted in Figure 3.2. More detailed features of the system such as parameters values, storage ponds depths, orifice size, and length of pipes can be found in the SWMM input files in our Github repository.[1]

The objectives in each optimization are to minimize flooding both upstream and downstream. To achieve this, three state variables were selected to inform actions: the depths at ponds 1 and 2, and a 24-hour perfect forecast of rainfall obtained from observational data. Future work could explore utilizing forecasts with error, as in Saliba et al. (2020), or for different lead times and lengths, but we fix this for simplicity here. The state variables inform the optimizer on how to adapt its policy/policies to mitigate flooding both upstream and downstream. Figure 3.3 displays the state-action dynamics, showing

[1]https://github.com/hosseinkavianih/Tackling-Complexity-EMODPS-vs-DDPG

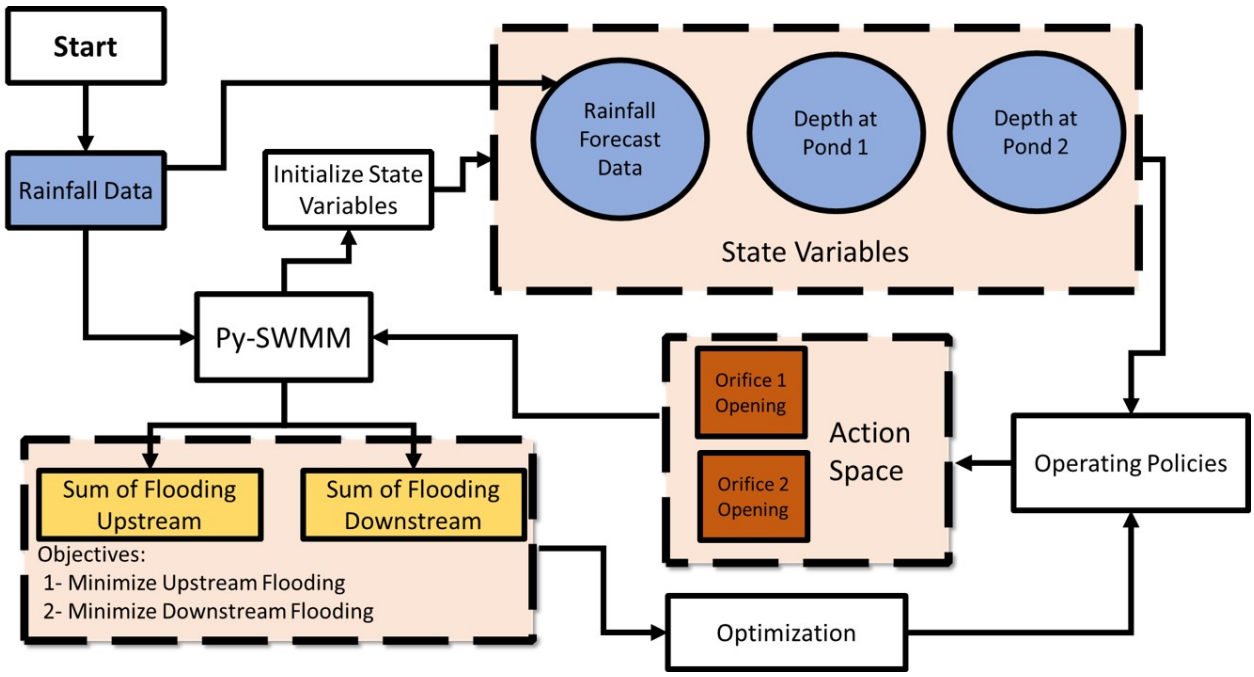

Figure 3.2: Feedback control loop of the optimization process. Rainfall data is passed to PySWMM, which initializes the state variables. The states (pond depths and rainfall forecast) are input to operating polices, which determine the actions to be taken (% of the pond orifices to open). PySWMM executes those actions, updates the state variables, and the process repeats. At each time step (for DDPG) or the end of the simulation (for EMODPS), objectives are computed based on the simulation. Finally, this process is coupled with an optimizer to update the operating policies to reduce simulated flooding.

how the actions and states are updated through interacting with the stormwater system. The actions prescribed by the policies correspond to the percentage of opening of orifices O1 and O2 in each 15-minute time step. Upon executing PySWMM, objective values are computed from the simulation, and the policies are optimized to minimize flooding both upstream and downstream.

To do this, we need to design a proper objective function for both DDPG and EMODPS. The Borg MOEA can take the objectives in Equations (3.11) and (3.12) as they are. The algorithm then iterates through the candidates that minimize these two functions over the course of the simulation horizon. However, DDPG uses a single reward and

Table 3.2: DDPG and EMODPS states, actions, and objectives

States	Actions	Objectives
• Depth at SP1 (m) • Depth at SP2 (m) • 24-hr rainfall forecast (mm/day)	• % openness of O1 • % openness of O2	• Downstream flooding (m^3/hr) • Upstream flooding (m^3/hr)

value function, and therefore cannot derive a Pareto set of alternative solutions in one optimization. Usually, a penalty or reward is supplied to represent the objective value. The value and policy functions are then updated and the next action is taken with the updated policy to further improve the objective. However, since we have two objectives here and need to derive a Pareto front from DDPG to have a fair comparison with EMODPS, we need a reward function that includes both component objectives of upstream and downstream flooding. To address this issue, we develop a DDPG reward function that assigns weights to different objectives. Natarajan & Tadepalli (2005) suggest using fixed weights for multiple objectives, while Abels et al. (2019) propose adaptive weights for different objectives. We used fixed weights assuming a fixed set of preferences over the simulation, but vary them for different optimizations to capture alternative values.

Our reward function was:

$$R(s_t, a_t) = -\omega_1 \frac{\sum \text{Flooding\_Upstream}}{\text{Max Flooding\_Observed}} - \omega_2 \frac{\sum \text{Flooding\_Downstream}}{\text{Max Flooding\_Observed}} \tag{3.13}$$

$$\begin{aligned} &\omega_1 \in (0, 0.05, 0.1, 0.15, \ldots, 0.95, 1) \\ &\omega_2 = 1 - \omega_1. \end{aligned} \tag{3.14}$$

To derive a Pareto front of alternative solutions, we suggest a vector of weights as $<\omega_1, \omega_2>$. By having $\omega_1 = 1-\omega_2$ and varying ω_1 between 0 and 1, we explore all the regions of the Pareto front, from where downstream flooding is penalized more than upstream flooding to the reverse. We selected increments of 0.05 for the weights, resulting in 21 combination of weights and therefore 21 control rules.

Lastly, we divided our rainfall data into two sets: one for training and one for testing. We trained our models on approximately four months of rainfall data, from June to September 1995. This specific period was chosen to include a variety of rainfall events with different intensities and durations. Additionally, considering the computational expenses and the need for hyperparameter tuning, a four-month training duration seemed reasonable. Each of the 21 control rules were trained for 12 GPU hours. Since each GPU hour equates to 3 service units (SU), while each CPU hour equates to 1 SU, the EMODPS optimization was run for 12 hours on 3 nodes with 21 cores/node, resulting in the same computational time.

To determine which algorithm generalizes better to unseen data, for the testing phase, we selected a period spanning roughly four years, from June 1996 to August 1999. This longer testing period was chosen because the policies optimized during training are applied without further optimization during testing, reducing computational demands. Moreover, using a longer testing period allows us to better evaluate the reliability of these solutions over new events.

3.6 RESULTS AND DISCUSSION

3.6.1 Performance of Optimized EMODPS and DDPG Policies

In this section, we present our results from the optimization. Figure 3.4 displays the objective values obtained by optimized solutions from each algorithm on both (a) the training

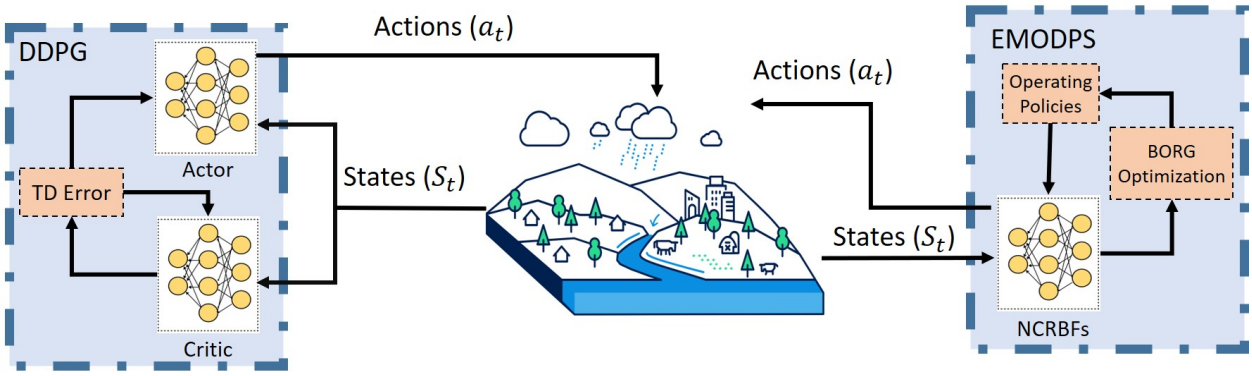

Figure 3.3: Illustration of the simulation-optimization approach to training DDPG and EMODPS networks. In DDPG, the states from the PySWMM environment are passed to the actor and critic networks, which then output the actions to be taken by the environment. The actor and critic networks are then updated by the TD error and the process repeats. In EMODPS, the states are passed to the policy function (here, NCRBFs), which outputs the actions to be taken. An optimization algorithm then optimizes the parameters of the policy function.

set and (b) the test set. For each of these plots, the x-axis represents the upstream flooding, i.e. the average hourly sum of overflows at the two ponds, and the y-axis represents the downstream flooding, i.e. the average hourly sum of flooding at links 1 and 2.

In Figure 3.4a, we see that the EMODPS solutions form a true Pareto set, while the DDPG solutions do not. Because EMODPS uses an MOEA as the solver, all solutions are guaranteed to be non-dominated. On the other hand, not all DDPG solutions are non-dominated since solutions are discovered by independently optimizing control rules with varying weights assigned to the two different objectives rather than with a non-dominated sorting algorithm. Not only are some of the DDPG solutions dominated by other DDPG solutions, but all of them are dominated by the EMODPS solutions. Therefore, EMODPS is more capable of finding policies that can minimize both objectives in this system.

To confirm the EMODPS policies do not overfit to the training set, we re-simulate the optimized policies from each algorithm over the test set. The objective values over

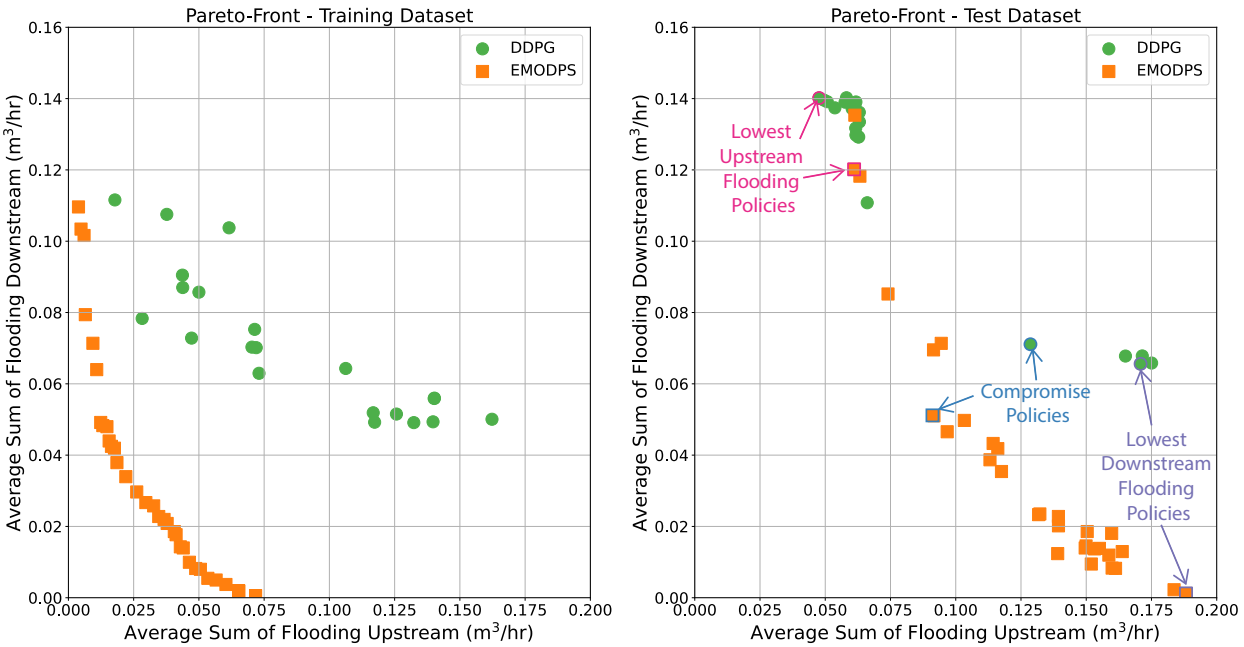

Figure 3.4: (a) Objective values of optimized control policies of EMODPS and DDPG algorithms on the training set. Note the DDPG solutions do not form a Pareto set, as some solutions found for certain sets of weights were dominated by others. (b) Objective values of the optimized control policies in panel (a) when re-simulated on the test set. The best solution on each objective and a compromise solution from each formulation are indicated in panel (b) and selected for further analysis.

the test set are shown in Figure 3.4b. Note that the EMODPS solutions are no longer all non-dominated, as they are being tested on a different time series. Policies that were non-dominated during the training period are not guaranteed to remain non-dominated during the testing period. Degradation occurs with respect to both objective values, but more so for upstream flooding. While all DDPG policies were dominated by at least one EMODPS policy in training, the best DDPG solutions for minimizing upstream flooding become non-dominated with respect to the EMODPS solutions on the test set. However, the best DDPG solutions for minimizing downstream flooding are still greatly dominated by several EMODPS solutions. Additionally, EMODPS continues to offer better compromise solutions that balance both objectives, with DDPG missing this region entirely.

3.6.2 Understanding Optimized EMODPS and DDPG Policies

To understand how the EMODPS control rules are able to achieve such strong performance compared to the DDPG control rules, we select three different solutions from each algorithm to investigate further. These solutions, highlighted in Figure 3.4b, are those with the lowest downstream flooding, the lowest upstream flooding, and a compromise solution. We see from Figure 3.4b that the DDPG solution with the lowest upstream flooding actually does better on this objective than the EMODPS solution with the lowest upstream flooding, although it does increase downstream flooding by a similar amount, making the two policies non-dominated with respect to one another. The DDPG solution with the lowest downstream flooding, however, performs much worse on this objective than the EMODPS solution with the lowest downstream flooding, with only a minor decrease in upstream flooding. Most notably, the EMODPS compromise solution outperforms the DDPG compromise solution on both objectives. To understand how this can be, we plot the time series of states (rainfall forecast and pond depths), actions (orifice opening), and objectives (upstream and downstream flooding).

Figures 3.5, 3.6, and 3.7 display these time series for the policies minimizing downstream flooding, upstream flooding, and the compromise, respectively, with the selected DDPG policy shown in the left column and the selected EMODPS policy in the right column. In each of these sets of plots, panels (a) and (b) show the state variables: the storage pond depths on the left axis, and the perfect 24-hr rainfall forecast on the right axis from the top down. Panels (c) and (d) show the resulting actions taken in response to these state variables: the % openness of O1 and O2. Finally, panels (e) and (f) show the volume of flooding downstream on the left axis, and upstream on the right axis. For visual clarity, only a 3.5-month period within the four-year testing phase is shown in Figures 3.5-3.7. Specifically, we focus on mid-May through the end of August of 1998, a period

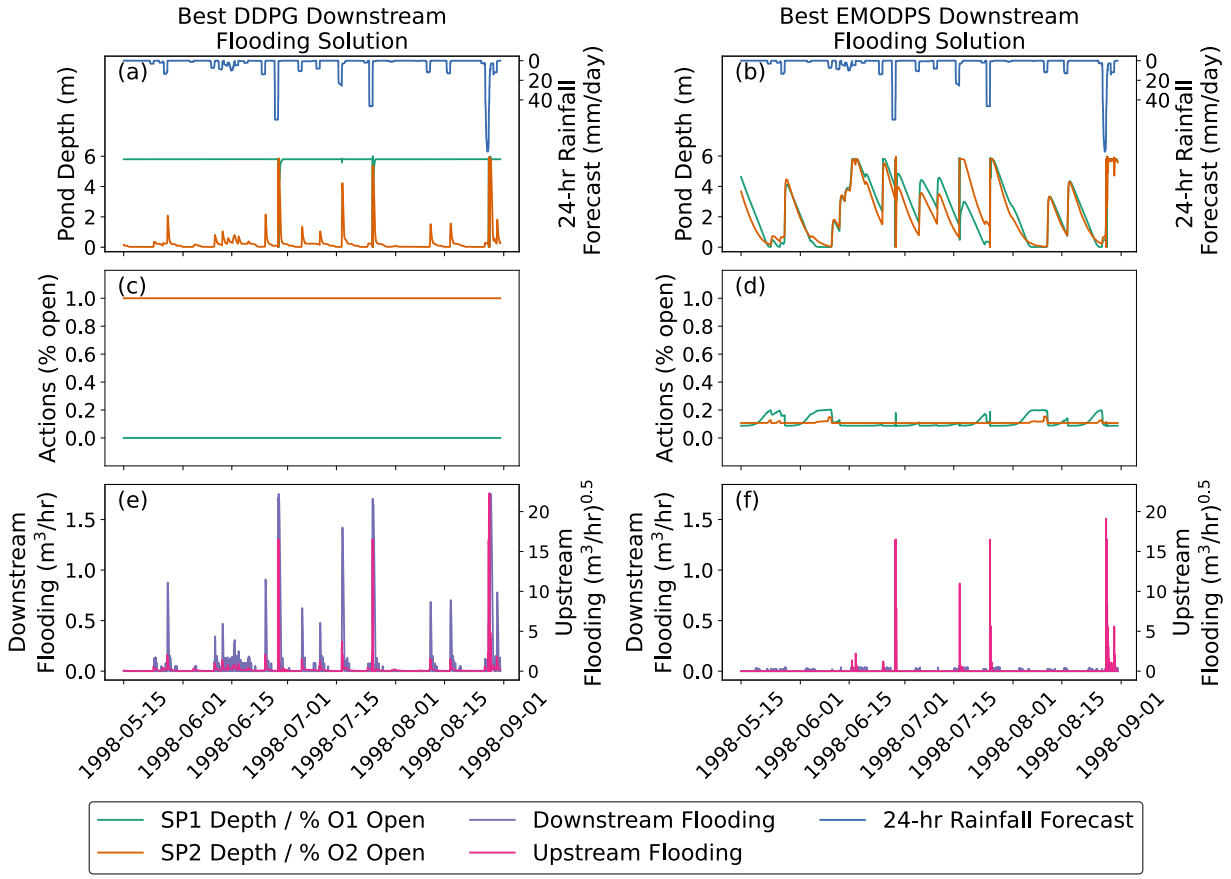

Figure 3.5: Time series of states, actions, and objectives of the DDPG and EMODPS policies with the lowest downstream flooding in the test set over the period of May 15 - Sep 1, 1998. (a,b) Perfect 24-hour rainfall forecast and storage pond depths of SP1 and SP2, (c,d) orifice openness of O1 and O2, and (e,f) flooding downstream and upstream for selected (DDPG, EMODPS) solutions.

that includes a variety of rainfall events with different intensities. Lastly, we performed a square root transformation on the upstream flooding values for visualization purposes. This adjustment was necessary because the magnitudes of upstream flooding are significantly higher than downstream flooding; upstream flooding just occurs less frequently, leading to similar average values.

First, we investigate the solutions that favor the downstream flooding objective. In Figure 3.5c, we see that during this period, the Best DDPG Downstream Flooding Solution

shuts O1 at all times and keeps O2 open at all times. Shutting O1 results in SP1 staying full at 6 m at all times, while opening O2 allows SP2 to drain (Figure 3.5a). However, SP2 is not able to drain water as quickly as it enters; consequently the depth in SP2 increases with each storm, and then empties afterward (Figure 3.5a). The filling of SP2 during the storm slows down outflows from the pond, decreasing downstream flooding (Figure 3.5e). However, if SP2 emptied slower, downstream flooding could be decreased further. The fast emptying of SP2 means it rarely reaches its maximum storage, so overflows from this pond are rare, but significant (see three large spikes in upstream flooding near the end of June, July, and August). There is also relatively frequent, but small upstream flooding during other periods despite lower rainfall because SP1 is always full, leading to overflows.

This approach to coordinating the pond operations is clearly not as effective as it could be. The Best EMODPS Downstream Flooding Solution keeps O1 about 15% open at all times, while alternating O2 between about 15% and 20% open during wet and dry periods, respectively, when the pond is partially full vs. empty (Figure 3.5d). Keeping each orifice partially open allows the ponds to drain faster than if they were fully closed, but slower than if they were fully open. This strikes a balance between goals of draining fast to reduce overtopping and consequent upstream flooding, and draining slowly to reduce downstream flooding. This balance is also achieved within events from changing the percent openness in response to state variables. Increasing the O1 opening when SP1 is empty also allows SP1 to initially fill fast when a new storm comes in, but then reducing the O1 opening during the storm slows outflows during the event.

The effectiveness of this coordination in better balancing between the two objectives is evident in the downstream and upstream flooding time series in Figures 3.5e-f. The Best EMODPS Downstream Flooding Solution barely experiences any downstream flooding

because the ponds effectively fill and slowly drain during each storm event. The ponds do fill a little more frequently for the Best EMODPS Downstream Flooding Solution than for the corresponding DDPG solution, resulting in four significant spikes in upstream flooding from overflows instead of only three. However, the EMODPS policy experiences less frequent small upstream flooding events since SP1 isn't full at all times. As such, the Best EMODPS Downstream Flooding Solution achieves far less downstream flooding with only a marginal increase in upstream flooding compared to the corresponding DDPG solution.

If one instead wants to minimize upstream flooding, Figure 3.6 displays the states, actions and objectives of the solutions from DDPG and EMODPS favoring this objective. Analyzing the Best DDPG Upstream Flooding Solution, we see this policy chooses to keep both orifices fully open at almost all times (Figure 3.6c). This results in the ponds being empty most of the time, except for when they quickly fill during storms. Yet just as quickly as the ponds fill, they drain (Figure 3.6a), resulting in significant downstream flooding. This does minimize the times that the ponds are full, though, resulting in only three spikes in upstream flooding from overflows, similar to the Best DDPG Downstream Flooding Solution, but with less frequent nuisance upstream flooding since SP1 is no longer full at all times.

The Best EMODPS Upstream Flooding Solution performs similarly in terms of flooding, but through very different actions. It keeps O1 open about 40% and O2 about 50% during dry periods, and then increases the openness in response to rainfall forecasts or pond depths, with the magnitude of openness increasing proportional to the magnitude of the forecast/pond depth (Figure 3.6d). Because the orifices are not fully open at all times, they drain slower after storm events, decreasing the magnitude of downstream flooding events for the Best EMODPS Upstream Flooding Solution compared to the Best

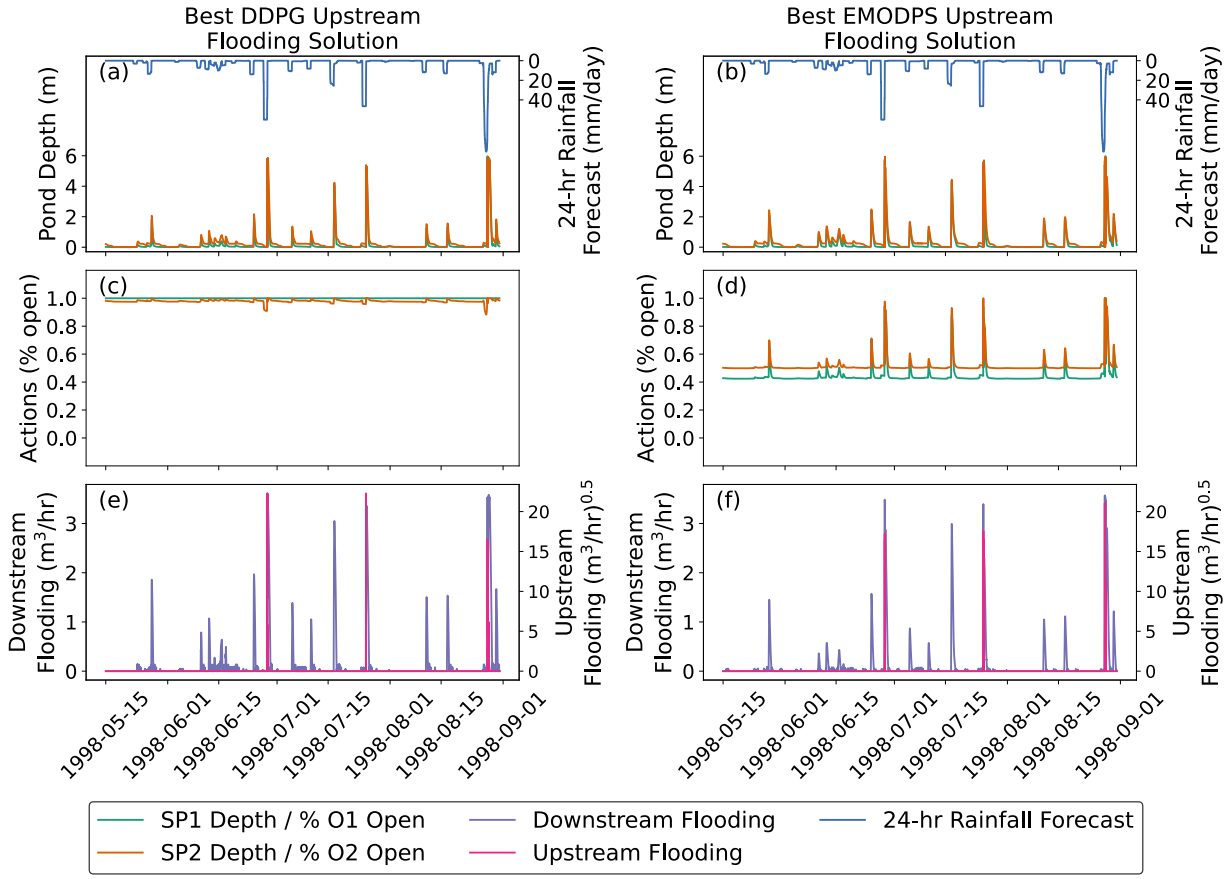

Figure 3.6: Time series of states, actions, and objectives of the DDPG and EMODPS policies with the lowest upstream flooding in the test set over the period of May 15 - Sep 1, 1998. (a,b) Perfect 24-hour rainfall forecast and storage pond depths of SP1 and SP2, (c,d) orifice openness of O1 and O2, and (e,f) flooding downstream and upstream for selected (DDPG, EMODPS) solutions.

DDPG Upstream Flooding Solution (Figure 3.6f vs. 3.6e). However, since the orifices are also not fully closed, the ponds don't fill so fast that they overtop any more frequently or significantly than for the Best DDPG Upstream Solution (Figure 3.6b vs. 3.6a). Consequently, the increase in upstream flooding for the Best EMODPS Upstream Flooding solution is only marginal compared to the Best DDPG Upstream Flooding Solution, with both only experiencing overtopping for three events in this period.

In each of the prior two figures, the EMODPS and DDPG policies were non-dominated

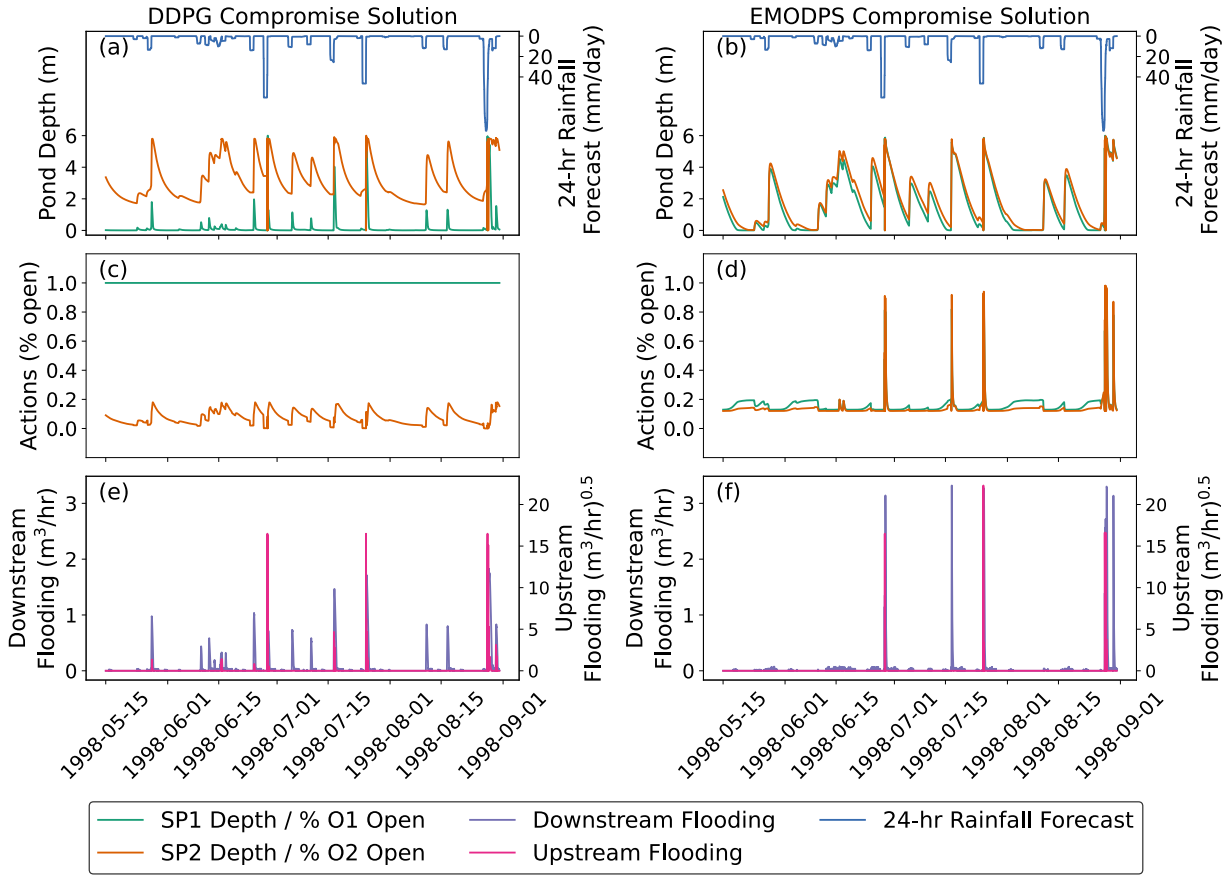

Figure 3.7: Time series of states, actions, and objectives of the DDPG and EMODPS compromise policies in the test set over the period of May 15 - Sep 1, 1998. (a,b) Perfect 24-hour rainfall forecast and storage pond depths of SP1 and SP2, (c,d) orifice openness of O1 and O2, and (e,f) flooding downstream and upstream for selected (DDPG, EMODPS) solutions.

with respect to one another. However, the EMODPS policy often represented a better compromise. For example, the Best Downstream Flooding Solution for EMODPS was much better on that objective but only marginally worse on the Upstream Flooding objective compared to the corresponding DDPG solution. The ability of EMODPS to find better compromise policies is highlighted more strongly by examining policies in the "knee" of each formulation's Pareto set, like those we have labeled Compromise Policies in Figure 3.4.

Looking at the actions of these policies in Figure 3.7, we see that the DDPG Compromise Solution attempts to balance opposing objectives by operating the two ponds in opposing ways. Similar to the Best DDPG Downstream Solution, the DDPG Compromise Solution keeps one orifice open at all times (this time O1 instead of O2), and the other near closed (this time O2 instead of O1). However the near-closed orifice does partially open in response to forecasts or pond depths for the DDPG Compromise solution (Figure 3.7c). This slows how quickly the ponds fill, reducing the magnitude of upstream flooding for the DDPG Compromise Solution during the three overtopping events compared to the Best DDPG Downstream Flooding Solution (Figure 3.7e vs. 3.5e). However, the DDPG Compromise Solution still experiences fairly frequent downstream flooding events.

The EMODPS Compromise Solution, however, greatly decreases the frequency of these events (Figure 3.5f). Similar to the Best EMODPS Downstream Flooding solution, this policy keeps O2 about 15% full during dry periods and O1 about 20% full, dropping that to 15% in response to forecast rainfall or increasing pond depths. However O2 is more responsive to state information for the EMODPS Compromise Solution, similar to the Best EMODPS Upstream Flooding Solution. Consequently, we see that the EMODPS Compromise Solution achieves a balance in performance through a blend of actions between the two extreme policies.

For the EMODPS Compromise Solution, five events in this 3.5-month period trigger the almost complete opening of the orifice in response to the pond being full (Figure 3.5d). Similar to the Best EMODPS Downstream Flooding Solution, the 15-20% opening of the orifices when the ponds are not full allow these ponds to drain slow enough to decrease the frequency of downstream flooding for the EMODPS Compromise compared to the DDPG Compromise (Figure 3.7f vs. 3.7e). However, similar to the Best EMODPS Upstream Flooding Solution, the sudden opening of O2 in response to SP2 reaching capacity

allows this pond to drain faster and reduce the frequency of overtopping events causing Upstream Flooding (four events for the EMODPS Solution in Figure 3.5f compared to three events for the DDPG Solution in Figure 3.5e). Thus EMODPS is able to find a better steady-state openness for dry periods, and better response to state variables during wet periods to compromise across objectives.

3.7 DISCUSSION AND CONCLUSIONS

This work addressed an open question in the literature on MORL: "Is it more effective to use strictly policy-based RL with MOEAs as optimizers or actor-critic-based RL with gradient-based solvers and different weights on multiple objectives?" To answer this question, we compared two prominent RL methods from each category on a stylized stormwater control problem with two conflicting objectives of minimizing downstream and upstream flooding: EMODPS for policy-based MORL, and DDPG for actor-critic-based MORL. While DDPG has been used in the literature to optimize RTC in stormwater systems, to the best of our knowledge, EMODPS has not been utilized in this context, adding another layer of novelty to this work.

Our study revealed the strengths and weaknesses of these methods and their effectiveness in deriving real-time control rules for conflicting objectives. We found that, while both methods were strong in optimizing rules for one of the two objectives (upstream flooding), EMODPS was significantly more capable of deriving effective solutions across all regions of the Pareto front, and in particular, in generating robust compromise solutions for the studied stormwater system. This was true in both training and testing periods, with EMODPS finding a better converged and more diverse Pareto set, with fewer gaps along the front. The reward function formulated for the DDPG method was designed to assign weights to two conflicting objectives, allowing it to continuously tran-

sition from favoring the upstream flooding objective to favoring the downstream flooding objective, and exploring trade-offs in between. However, the Pareto set on the test set clustered in two regions favoring one or the other objective, with few compromise solutions. As such, the weighting method was not as effective as utilizing an MOEA with EMODPS.

It is important to note that in the initial experiments with DDPG, the solutions were even less diverse than those presented in this study. The improved diversity was achieved through a thorough hyperparameter tuning. Hyperparameter tuning for DDPG involves tuning two sets of hyperparameters:

- RL hyperparameters: actor and critic learning rates, discount factor, batch size, exploration noise, and replay buffer size.
- Deep learning hyperparameters: number of hidden layers, hidden units, activation functions, optimizer, and regularization parameters.

EMODPS requires far less hyperparameter tuning, simply requiring one to decide on the functional form of the policy and the number of basis functions.

Finally, the comparison in this paper also provided valuable insights into how to best control stormwater ponds to favor reduction in downstream flooding, upstream flooding, or a compromise between the two. Downstream flood reduction was best achieved by keeping orifices partially open (in this system at about 15-20%) to balance between slowly draining ponds to reduce downstream flooding, and slowly filling them to reduce the probability of overtopping and causing upstream flooding. Other systems will likely settle on different values depending on the storage volumes and inflow rates, but the insights of partially opening the orifices likely remain. If one wants to minimize upstream flooding, keeping the ponds fully open is most effective for slowly filling and quickly draining the ponds to reduce overtopping. However, effective compromise solutions can

be achieved by balancing the behavior of downstream-favoring and upstream-favoring solutions by maintaining partial openness at most times to favor downstream flood reduction, until a forecast or high storage depth triggers complete openness to favor upstream flood reduction.

In conclusion, the insights from this paper are useful for informing 1) the choice of RL method to find a Pareto set of alternative multi-objective control rules, and 2) the design of stormwater control rules for flooding in different parts of a stormwater system. While policy-based methods with MOEAs were found to outperform actor-critic methods with gradient-based solvers in this study, future work should investigate whether those conclusions hold for problems with more than two objectives. We hypothesize that the benefits of EMODPS over DDPG will only increase for such problems, as the non-dominated sorting of MOEAs should mitigate the curse of dimensionality associated with optimizing for multiple objectives using the weighting approach. Finally, it is important to note that both the training and testing in this study utilized the same simulation model. Of course, no simulation model perfectly represents the true system. Future work should investigate how to best design control rules given uncertainty in the true representation of the system, accounting for both parametric and structural uncertainty, as well as model error.

3.8 CODE AND DATA

Code development history for DDPG, EMODPS, and post-processing may be found on our GitHub repository.[2]

[2]https://github.com/hosseinkavianih/Tackling-Complexity-EMODPS-vs-DDPG

CHAPTER 4

Designing Stormwater Control Rules Under Parametric Uncertainty

4.1 ABSTRACT

Robust optimization methods have been utilized in the literature to optimize real-time control rules (RTCs) in stormwater systems. However, previous studies have focused on characterizing climate change or future development uncertainties and neglected parameter uncertainty in their optimization strategies. This study addresses this gap by designing a robust optimization framework for a stylized urban stormwater system aimed at mitigating flooding. We first leverage Differential Evolution Adaptive Metropolis (DREAM) to characterize parameter uncertainty, using prior information of parameters, a Student-t distribution as the likelihood function, and a synthetic truth to evaluate MCMC model convergence. We then compare three optimization strategies: 1) the traditional Maximum A-posteriori (MAP) approach for designing RTCs to optimize performance on the most likely parameterization, 2) Multi-Objective Robust Optimization (MORO) for designing RTCs to optimize performance over a posterior-weighted average of multiple likely parameterizations, and 3) Min-Max optimization for designing RTCs to optimize performance in the worst-case of the likely parameterizations. Comparing these strategies against optimization to the synthetic true parameterization, MORO emerged as the most effective in achieving a compromise policy for minimizing flooding objectives in multiple locations, while Min-Max optimization reported objective values that were closer to their values on the synthetic truth. The insights from this study can help decision-makers incorporate robust optimization strategies into their systems, improving the effectiveness of stormwater management.

4.2 INTRODUCTION

Decision-making for engineering systems is often informed by simulation models that abound with uncertainties (Chankong & Haimes, 2008; Coello, 2018; Koutsoyiannis & Economou, 2003). Srikrishnan et al. (2022) categorize uncertainties in simulation models into three types: structural, parametric, and sampling. Structural uncertainty refers to uncertainty in the mathematical or rule-based representation of the model, parametric uncertainty refers to uncertainty in the values of internal model parameters, and sampling uncertainty refers to stochastic variability in exogenous forcings to the model. These uncertainties can significantly impact the outcomes and effectiveness of decisions, making it crucial to account for them in the decision-making process (Marchau et al., 2019; Lempert et al., 2013; Giuliani & Castelletti, 2016). This consideration becomes especially significant when optimizing engineering designs for complex systems with multiple conflicting objectives (Quinn et al., 2017).

Robust optimization encompasses various methods aimed at protecting decision-makers from such inherent uncertainties Wasko et al. (2021); Dittrich et al. (2016). Robust optimization methods are typically classified based on their level of risk aversion McPhail et al. (2018). For example, min-max optimization is frequently used to design for the worst-case scenario across possible outcomes (Wald, 1949). More risk-neutral approaches instead design systems to work best in expectation (Bartholomew & Kwakkel, 2020; Shavazipour et al., 2021).

Many studies have employed robust optimization to address uncertainties in diverse fields such as logistics and supply chain management (Cacchiani et al., 2020), energy systems (Shen et al., 2020), and water resources systems (Quinn et al., 2017). Our study focuses on stormwater systems, optimizing their control rules to mitigate flooding. In the literature, several studies have characterized uncertainties in stormwater systems,

primarily addressing those due to climate change or future urban development (Dotto et al., 2012, 2014; Kleidorfer et al., 2009). To enhance decision making under these conditions, Bahrami et al. (2019) developed a stormwater management system using genetic algorithms to design control rules robust to future development uncertainties. Yu et al. (2022) leveraged a stochastic system for urban stormwater management measures, making them robust to future precipitation data from Global Circulation Models (GCMs). Oh & Bartos (2023) developed a model-predictive control (MPC) algorithm that is robust to uncertainties in both pollution forecasts and water quality measurements.

However, in addition to such sampling uncertainty, stormwater models often suffer from substantial parametric uncertainty due to the lack of sufficient sensing data for calibration. While many papers address climate change and model uncertainty in their stormwater decision-making processes (Kleidorfer et al., 2009; Dotto et al., 2012), none focus on optimizing control rules for stormwater systems that are robust to parameter uncertainty. This is concerning because designing based on a single assumed parameter set may result in significant over- or under-investment. For example, prior work by Smith et al. (2024) found that reforestation plans for flood control underestimate the needed investment for a reduction of 20% in high flows by 18% when designing to the most likely parameter set, but only by 11% when using robust optimization across posterior-weighted parameter sets. This was the first study to use formal likelihood measures to quantify parametric uncertainty for robust stormwater design, and to quantify its benefits by comparing to performance optimized to a synthetic true parameter set. The few prior studies in which parameter uncertainty had been considered in stormwater system design (Jia & Culver, 2006; Jiang et al., 2017; Xu et al., 2020) used informal Generalized Likelihood Uncertainty Estimation (GLUE) (Beven & Binley, 1992) which can result in great over- or under-estimation of uncertainty (Stedinger et al., 2008; Montanari, 2005),

leading to the same design concerns of over or under-investment.

In this study, we seek to address a gap in the literature by characterizing parameter uncertainty in an urban stormwater model and then designing control rules to mitigate flooding that are robust to that uncertainty. The characterization process involves adjusting model parameters to fit observed data and improve the models predictive performance. However, this process is not without its challenges. Oftentimes, multiple parameter combinations can yield similar performance, called "equifinality" (Beven & Binley, 1992, 2014). These different possible parameter values may have different design implications. Consequently, it is important to quantify and design for this uncertainty. To achieve this, we leverage Markov Chain Monte Carlo (MCMC), a prominent Bayesian calibration method that has been shown in the literature to be a powerful statistical tool for modeling parameter uncertainty.

In order to evaluate the effectiveness of our robust optimization approach, we set synthetic true values for model parameters so that we can determine how well the robust optimization approaches perform in reality, something that cannot be assessed in the real world. However, being able to assess performance in a synthetic case where we know the truth provides evidence for which approaches are most likely to generalize to such real-world cases. Using the synthetic true parameter values, we simulate "observed" data from our physical model and then calibrate the model to try to fit to these observations using MCMC. We then select a diverse set of likely parameter sets from the calibration to which we optimize controls rules using robust optimization. Following Smith et al. (2024), we compare two different robust optimization schemes: Multi-Objective Robust Optimization (MORO) (Kwakkel et al., 2015; Bartholomew & Kwakkel, 2020) and Min-Max optimization (Wald, 1949). For comparison, we also optimize to the parameter set with the Maximum A-posteriori Probability (MAP), the more traditional approach that

ignores parameter uncertainty. MAP provides the most likely parameter set values derived through the Bayesian calibration step, but there is a concern about what happens if the likely parameter set is not the synthetic truth. We hypothesize that MORO and Min-Max will be more likely to perform well on the synthetic truth because they account for multiple likely parameterizations (Kavianihamedani et al., 2024; Laloy & Vrugt, 2012).

Our study is organized as follows. Our methods are described in Section 4.3. Section 4.3.2 describes the Bayesian calibration process, Section 4.3.1 outlines the case study, and Section 4.3.3 highlights the robust optimization methods we performed for the comparison. We display the results of the optimization experiments and the analysis in Section 4.4. Finally, we discuss our conclusions about which optimization strategy provides better trade-offs in our stormwater system, and note areas for future work in Section 4.5.

4.3 METHODS

In this section, we describe our methodology and experiments to design robust control rules for a stormwater system. First, we introduce the case study and objectives of robust optimization, then we introduce Bayesian calibration method for calibrating the hydrological model parameters. Lastly, we explain the different optimization formulations and how we select parameterizations to implement our experiments.

4.3.1 Case Study

We propose to calibrate the same stylistic SWMM model in chapter 3 shown in Figure 3.1a. The optimization objectives of this case study are also the same as before: minimize total upstream and downstream flooding.

4.3.2 Bayesian Calibration

We leverage Bayesian calibration as a powerful statistical technique to estimate the probability distribution of SWMM model parameters based on the similarity of their simulated ponds depths to those simulated under the synthetic truth. This method enables us to use prior knowledge of parameter values in similar systems and data simulated under the synthetic truth to form a posterior joint distribution of parameters that reflects updated priors. The following subsections detail the selection of parameters for calibration, the formulation of the likelihood function, and the algorithm employed for Bayesian calibration.

4.3.2.1 Parameter Selection

We choose a subset of SWMM model parameters to calibrate based on their sensitivity, as discussed in the literature. These parameters, listed in Table 4.1, play pivotal roles in influencing the model's behavior and performance.

To establish the parameter priors for calibration, we refer to the uncertainty levels recommended in "Rules for Responsible Modeling" (James, 2003). We apply an uncertainty level within this range as a multiplier on a base value (that used in chapter 3) of each parameter to obtain the calibration range, i.e. calibration range = base value $\pm$ base value $\times$ uncertainty level. We set the synthetic true parameter values to be the same as the base value parameters used in the chapter 3, meaning the synthetic truth in this case is always the median of the calibration range. The synthetic truth is used in the SWMM model to simulate the "observed" output that we seek to calibrate to. The synthetic truth, uncertainty level, and resulting calibration range are listed in Table 4.1.

Table 4.1: SWMM model parameters to be calibrated.

Parameter	Description	Synthetic True Value	Uncertainty Level in Literature	Uncertainty Level Applied	Parameter Calibration Range
Width (m)	Width of overland flow	1000	50 - 100%	50%	500 - 1500
Slope (%)	Gradient of subcatchments	0.5	10 - 100%	55%	0.225 - 0.775
N Imperv ($s/m^{1/3}$)	Manning's roughness coefficients for impervious surfaces	0.01	10 - 25%	17.5%	0.00825 - 0.01175
N perv ($s/m^{1/3}$)	Manning's roughness coefficients for pervious surfaces	0.16	50 - 100%	75%	0.04 - 0.28
Minimum Infiltration Rate (mm/hr)	Equivalent to soil's saturated hydraulic conductivity	0.5	25 - 50 %	37.5%	0.3125 - 0.6875
Decay constant (hr^{-1})	Infiltration rate decay constant for Horton curve	4	25 - 50 %	37.5%	2.5 - 5.5
Drying time (days)	Time for fully saturated soil to dry	7	50 - 100 %	75%	1.75 - 12.25

4.3.2.2 Likelihood Function

Bayesian calibration requires estimation of the likelihood of each parameter set. To determine an appropriate likelihood function, we generated a Latin hypercube sample of 100 parameter combinations over their calibration range. We then ran SWMM for these parameter sets with the orifices 20% open at all times. For each simulation, we computed the residuals, ϵ, between simulated and "observed" depths in storage ponds 1 and 2, where

"observed depths" are defined as those simulated by the model with the synthetic true parameter values. For each simulation, we then computed the likelihood of the residuals assuming different distributions: normal, Student t, Cauchy, and asymmetric Laplace. For each distribution, we made Q-Q plots of the empirical vs. fitted quantiles (QQ-plots) for each pond, and plots of the log-likelihood vs. Nash-Sutcliffe Efficiency (NSE) (McCuen et al., 2006), between simulated and observed depths at each pond. The best distribution was then chosen subjectively based on which QQ-plots most closely matched and which log-likelihoods were most strongly positively related to the performance metrics. QQ-plots are shown for one of the Latin hypercube samples of parameter values with the ponds 20% open in Appendix B Figure B.1, and of NSE vs. log-likelihood in Appendix B Figure B.2. Based on this comparison, the Student-t distribution was chosen. Equations 4.1-4.3 display the final form of the likelihood function.

$$\log L = \sum_{t=1}^{T} \log\left(f_1(x_{1,t})\right) + \sum_{t=1}^{T} \log\left(f_2(x_{2,t})\right) \tag{4.1}$$

$$f_i(x_{i,t}) = \frac{\Gamma(\frac{\nu_i+1}{2})}{\sqrt{\pi\nu_i}\Gamma(\frac{\nu_i}{2})}\left(1 + \frac{x_{i,t}^2}{\nu_i}\right)^{-(\nu_i+1)/2} \tag{4.2}$$

$$x_{i,t} = \frac{\epsilon_{i,t} - \mu_i}{\sigma_i} \tag{4.3}$$

where $\epsilon_{i,t}$ are the residuals at time t for pond i, T is the number of time steps in the simulation, and ν_i, μ_i, and σ_i are parameters of the Student-t distribution for the pond i residuals.

4.3.2.3 *Algorithm*

To perform Bayesian calibration, we need an effective search algorithm to derive the joint posterior of the model parameters given their prior ranges and likelihood function. Among different statistical methods, MCMC is a robust candidate for this task. In chapter 2, we observed that the DiffeRential Evolution Adaptive Metropolis algorithm (DREAM) was the most reliable and effective MCMC algorithm tested on both test problems. Therefore, we chose it as our MCMC algorithm. We ran DREAM using the PyDREAM package in Python for 100,000 iterations and 15 chains. The parameters of the Student-t distribution were not included in the Bayesian estimation, only the 7 parameters listed in Table 4.1 for each of the 2 SWMM subcatchments, resulting in 14 calibration parameters. For each MCMC sample, the Student-t distribution parameters were simply estimated using Maximum Likelihood Estimation as part of the `scipy.stats` library in Python.

4.3.3 Robust Optimization

After performing Bayesian calibration and deriving the joint posterior of the 14 SWMM parameters, we introduce robust optimization methods to design control rules that generalize well across different likely SWMM parameter sets.

4.3.3.1 *Optimization Methods*

In this study, we optimize stormwater control rules using three multi-objective optimization strategies. The first strategy optimizes control rules using the SWMM parameter set with the maximum a posteriori (MAP) probability from the Bayesian calibration. The other two strategies select a diverse set of likely parameter sets from the calibration and optimize performance in expectation across them, or in the worst case. We call these multi-objective robust optimization (MORO) and min-max robust optimization (Min-

Max), respectively. We evaluate the performance of the MAP, MORO, and Min-Max optimizations by re-simulating their optimized control rules on the synthetic true parameterization. For reference, we also optimize control rules to the synthetic truth (Truth) to see what performance levels could be achieved in absence of uncertainty.

Mathematically, the multi-objective optimization of formulation f can be written in the following form:

$$\text{Minimize} \quad |O^f| = |O_1^f, O_2^f| \tag{4.4}$$

where O^f is a vector of two objective functions representing upstream and downstream flooding, but summarized differently across parameterizations for each formulation f. In the Truth formulation O_1^f and O_2^f are the total overflows at ponds 1 and 2, and total flooding at links 1 and 2, respectively, under the synthetic true parameter set. These are quantified by Equations 4.5-4.6:

$$O_1^{Truth} = (Overflow_{\text{SP1},\theta_{Truth}} + Overflow_{\text{SP2},\theta_{Truth}}) \tag{4.5}$$

$$O_2^{Truth} = (Flooding_{\text{Link1},\theta_{Truth}} + Flooding_{\text{Link2},\theta_{Truth}}) \tag{4.6}$$

where θ_{Truth} represents the synthetic true parameter set.

In the MAP optimization method, the objectives O_i^{MAP} are computed in the same way, but replacing θ_{Truth} with θ_{MAP}, the parameter set with the maximum a-posterior probability:

$$\theta_{MAP} = \arg\max_{\theta} P(\theta|\epsilon) = \arg\max_{\theta} \frac{p(\epsilon|\theta)p(\theta)}{\int p(x|\theta)p(\theta)d\theta}. \tag{4.7}$$

Here θ represents a vector of SWMM parameters, ϵ denotes the residuals in Pond 1 and

Pond 2 depths, $P(\theta|\epsilon)$ is the posterior probability of θ given ϵ, and $p(\epsilon|\theta)$ is the likelihood of ϵ given θ. θ_{MAP} represents the mode of the posterior distribution and is often employed as a point estimate in Bayesian inference.

Unlike MAP, MORO seeks to minimize the objectives over a selection of parameter sets with different likelihood values. This method minimizes the likelihood-weighted average of the objective values derived from a set of likely SWMM model parameterizations, including MAP:

$$O_i^{MORO} = \sum_{\theta} \left(\frac{L_\theta}{\sum_\theta L_\theta} O_{i,\theta} \right) \tag{4.8}$$

Here, L_θ is the likelihood of the parameterization θ and $O_{i,\theta}$ is the i-th objective value in that parameterization.

The third and final strategy is min-max robust optimization (Min-Max). This approach aims to minimize the worst objective value across the same set of selected parameter sets used for MORO, ensuring that the solution is robust against the most adverse of these conditions. The min-max objectives can be described as follows:

$$O_i^{MinMax} = \max_{\theta} O_{i,\theta} \tag{4.9}$$

4.3.3.2 Parameterizations Selection

Computation of both O_i^{MORO} and O_i^{MinMax} requires the selection of different parameter sets θ to which we would like our control rules to be robust. Ideally, we would select all parameter sets with a high likelihood, but this may not be computationally tractable. To reduce the number of parameter sets, we can choose a selection among those with high likelihood that capture a diverse set of parameter values, dropping sets that are similar to others, providing little additional value to the robust optimization. To achieve this, we

apply k-means clustering to all unique parameter sets sampled after burn-in to discover clusters with similar characteristics, and then choose only one representative parameter set from each cluster. K-means clustering aims to partition a collection of n parameter sets $\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_n$ into k clusters $\mathcal{C}_1, \mathcal{C}_2, \ldots, \mathcal{C}_k$ such that the within-cluster sum of squares is minimized:

$$\underset{\mathcal{C}}{\operatorname{argmin}} \sum_{i=1}^{k} \sum_{\mathbf{y} \in \mathcal{C}_i} \|\mathbf{y} - \boldsymbol{m}_i\|^2 \tag{4.10}$$

- $\mathcal{C} = \{\mathcal{C}_1, \mathcal{C}_2, \ldots, \mathcal{C}_k\}$ represents the set of clusters.
- $\boldsymbol{m}_i$ is the centroid of cluster $\mathcal{C}_i$.
- $\|\mathbf{y} - \boldsymbol{m}_i\|^2$ is the squared Euclidean distance between parameter set $\mathbf{y}$ and the centroid $\boldsymbol{m}_i$.

The K-means algorithm first chooses k initial centroids $\boldsymbol{m}_1, \boldsymbol{m}_2, \ldots, \boldsymbol{m}_k$. After that, it assigns each parameter set $\mathbf{y}_j$ to the cluster with the nearest centroid (equation 4.11). Lastly, it updates the centroids by calculating the mean of all points assigned to each cluster:

$$\mathcal{C}_i^{(t)} = \left\{ \mathbf{y}_j : \|\mathbf{y}_j - \boldsymbol{m}_i^{(t)}\|^2 \leq \|\mathbf{y}_j - \boldsymbol{m}_l^{(t)}\|^2, \forall l, 1 \leq l \leq k \right\} \tag{4.11}$$

$$\boldsymbol{m}_i^{(t+1)} = \frac{1}{|\mathcal{C}_i^{(t)}|} \sum_{\mathbf{y}_j \in \mathcal{C}_i^{(t)}} \mathbf{y}_j \tag{4.12}$$

To determine the appropriate number of clusters, we apply the elbow method. We run the K-means clustering algorithm for a range of cluster numbers, and for each k, calculate the SSE (equation 4.10). We then plot the SSE values against the number of clusters k. The

optimal number of clusters is identified at the point where the SSE begins to decrease at a slower rate, forming an "elbow" in the plot. After specifying the number of clusters, we identify the parameterization closest to the centroid of each cluster and designate it as one of the likely parameter sets, except in the case that the cluster includes the MAP parameterization. For that cluster, we instead select the MAP parameterization. This ensures that MAP is included as one of the likely parameterizations in both the MORO and MAP strategies.

4.3.3.3 Computational Implementation

For each optimization, we designed control rules for the two ponds using Evolutionary Multi-Objective Direct Policy Search (EMODPS) (Giuliani et al., 2016) with the Borg multi-objective evolutionary algorithm (Hadka & Reed, 2013) as the solver, as this approach displayed superior performance in finding a diverse Pareto-front compared to Deep Deterministic Policy Gradients (DDPG) (Lillicrap et al., 2015) in chapter 3. We used the control policy formulation with Non-Convex Radial Basis Functions (NCRBFs) described by Equation 3.1. For fairness, each optimization was performed using the same computational resources: 3 nodes with 21 cores/node for 10 hours, totaling 630 core-hours. Note, since the objective functions for MORO and MAP require running the SWMM model k times across the k selected parameter sets, they can therefore perform only about $1/k$ as many function evaluations over this time. We assess whether providing additional likely parameterizations to inform the control rules is worth this computational hit by evaluating the performance of the optimized policies from each formulation on the synthetic truth.

4.4 RESULTS AND DISCUSSION

4.4.1 Parameter Calibration Results

In this section, we present our results for the Bayesian calibration of SWMM parameters using the DREAM algorithm. Figure 4.1 displays the trace plots of parameter posterior values across different chains at different iterations of the search. Iteration values range from 10,000 to 100,000, as we discarded the first 10% of the iterations as burn-in. From the total 14 calibrated parameters, we choose 3 to illustrate different convergence scenarios.

Figure 4.1 shows the trace plots for (a) Manning's N of impervious surfaces (N Imperv), (b) max infiltration rate, and (c) soil drying time. For N Imperv, we can see the algorithm chains converge to a single mode, but one that is different from the synthetic truth (Figure 4.1a). For the max infiltration rate, we see some chains converging to a value of 0.49, which is close to the true value of 0.5; however others converge to values between 0.45-0.46, suggesting there may be some equifinality in this parameter, whereby different values can achieve similar performance in simulating pond depths that are close to the synthetic observations. For the drying time, we see different dynamics still, as the posterior values are relatively noisy and do not converge to a single value, suggesting there is great uncertainty in this parameter. This suggests the drying time does not strongly influence the model's ability to simulate values close to the synthetic observations of pond depths. However, that does not necessarily mean control rules won't influence outcomes differently in model runs with different values of initial deficit. These results reinforce the study's motivation to consider multiple likely parameterizations in designing robust robust control rules under parameter uncertainty. If parameters estimated by the calibration are (a) incorrect, (b) equifinal, or (c) highly uncertain, choosing a single set of parameters for design could have severe consequences.

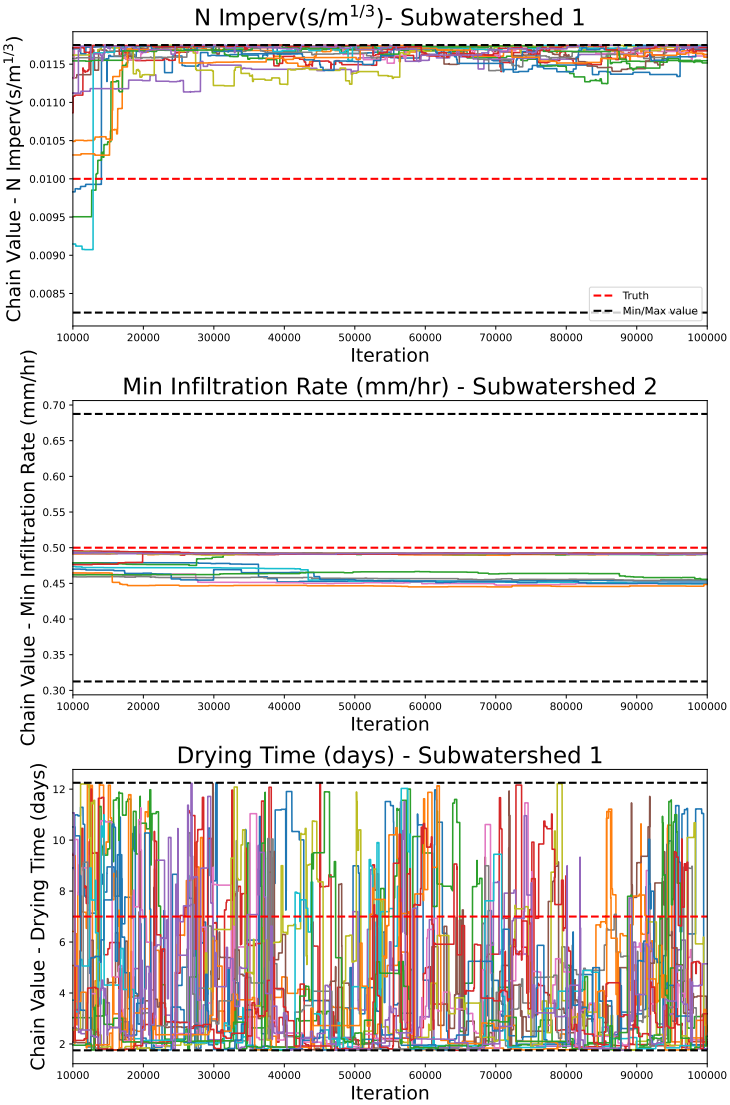

Figure 4.1: Trace plots of (a) N Imperv, (b) Max Infiltration Rate, and (c) Drying Time values sampled by each chain of DREAM over the course of the search after removing burn-in

4.4.2 Parameterization Selection Results

After deriving the posterior values for different parameters, we apply k-means clustering to determine the number of distinct parameterizations to use for the robust optimization formulations. Based on the within-cluster sum of squares (WCSS) values, we choose 4

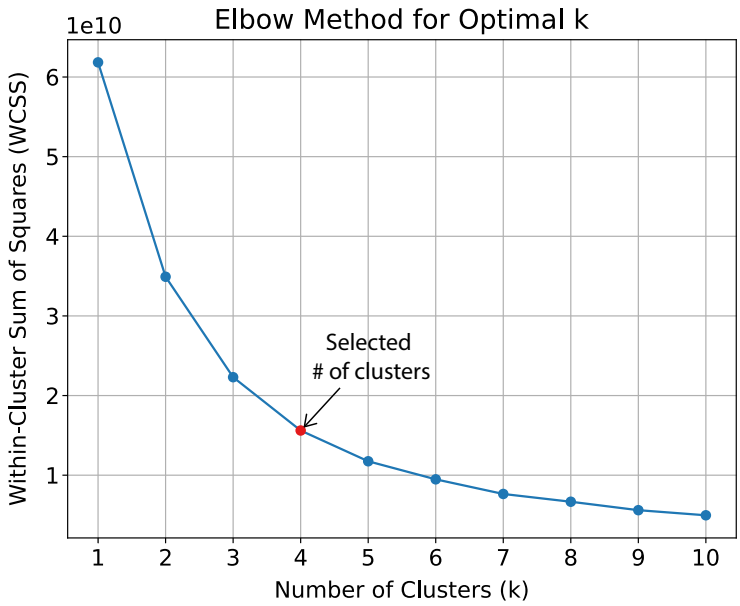

Figure 4.2: K-mean clustering plot - displaying Within-Cluster Sum of Squares (WCSS) vs number of clusters.

clusters as the optimal number (see Figure 4.2). This choice is justified as the WCSS decreases at a slower rate beyond this point. After specifying the number of clusters, we first identify which cluster the MAP parameterization falls within. For the remaining clusters, we find the parameterizations closest to their centroids. We then designate these three parameter sets and the MAP parameterization as the likely parameter sets to be used for the robust optimization formulations (MORO and Min-Max).

4.4.3 Performance of Optimized and Re-simulated Policies With Different Optimization Methods

In this section, we present the results of the three different optimization (MAP, MORO and Min-Max) strategies and the optimization to the synthetic truth. Similar to Figure 3.4 in the previous chapter, Figure 4.3a displays the objective values obtained by the policies of each strategy based on their own formulation of the objective functions. Figure 4.3b

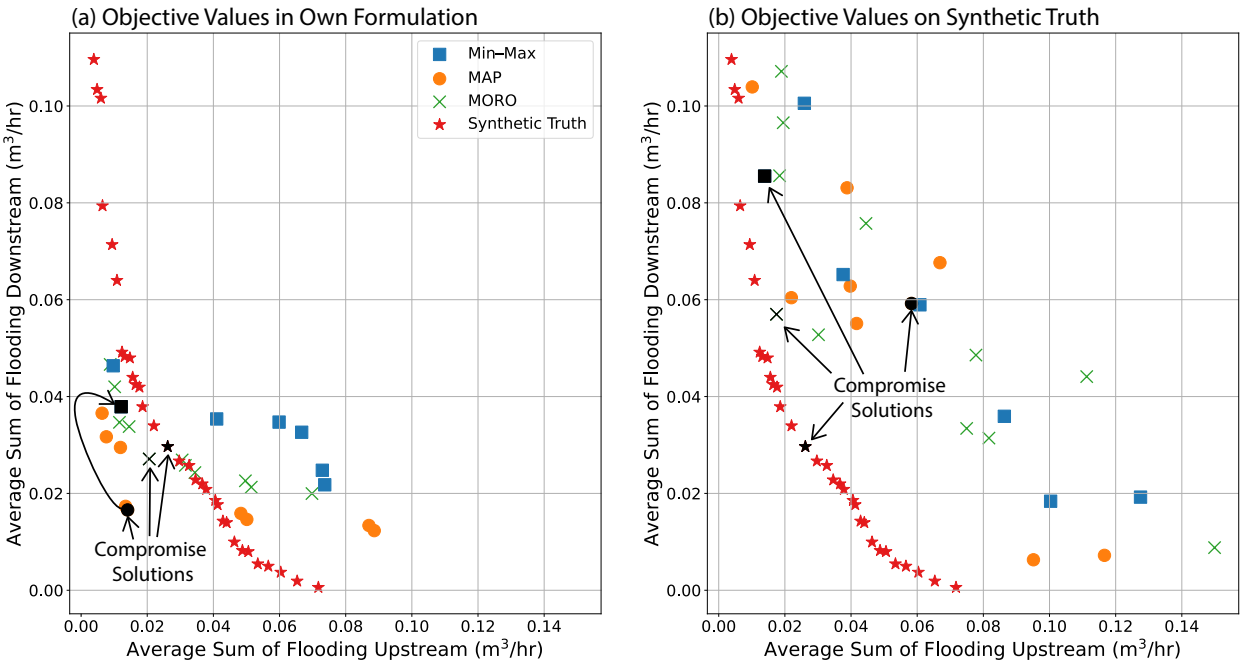

Figure 4.3: (a) Objective values of optimized control policies of MAP, MORO, and Min-Max along with synthetic truth as determined . (b) Objective values of the control policies in panel (a) when re-simulated on the synthetic truth. A compromise solution from each formulation based on their performance in panel (a) is selected for further analysis, and their corresponding re-simulated values on the synthetic truth are shown in panel (b).

shows the objective values when those policies are re-simulated using SWMM with the synthetic true parameter set. These values are of course unchanged for the policies optimized to the synthetic truth (red stars). Note that this Pareto front is the same as the one derived in chapter 3.

In Figure 4.3a, we observe that the three optimization strategies differ considerably from each other in the objective values they obtain. The objective values of the policies optimized using the MAP strategy largely dominate those of both MORO and Min-Max, and even the synthetic truth. However, domination of the synthetic truth suggests these objective values are an overestimation of what they can truly achieve. Between MAP and MORO, the perceived objective values of MORO's solutions are slightly better than

those of Min-Max. This is not surprising given the Min-Max strategy reports the worst objective value achieved across parameterizations, while MORO reports the posterior-weighted average.

To determine which strategy is actually superior, Figure 4.3b shows the performance of the policies optimized to each strategy when re-simulated over the synthetic truth. Here we see that across all formulations, the objective values of the policies generally degrade on the synthetic truth – even the Min-Max policies that reported the worst objective value obtained across the parameterizations selected for optimization. To quantify this degradation, Figure 4.4 shows the kernel density estimate (KDE) of the differences in objective values between the optimized and re-simulated solutions for the different optimization strategies. The closer the peak of the KDE is to zero, the smaller the difference between optimization and re-simulation across policies, indicating the objective values from the optimization more closely reflect what can be achieved in reality. Positive values indicate superior objective values could be achieved than what was found in optimization, while negative values indicate the reverse.

From Figure 4.4, we observe that the peak for Min-Max is closer to zero for both objectives than for MORO and MAP. Not surprisingly, the lower tail of the Min-Max distribution is also less negative on both objectives, indicating it is less likely to underestimate flooding by nature of minimizing the worst case across objectives. Following Min-Max, MORO's peak is next closest to zero on both objectives. However, MORO does have a longer lower tail than MAP on the upstream flooding objective, indicating it is more likely to underestimate upstream flooding, but the reverse is true for downstream flooding.

Overall, these findings suggest engineers should use Min-Max optimization for this problem if they are most concerned with the objective values from the optimization reflecting values close to what can truly be achieved, particularly if they want to reduce the

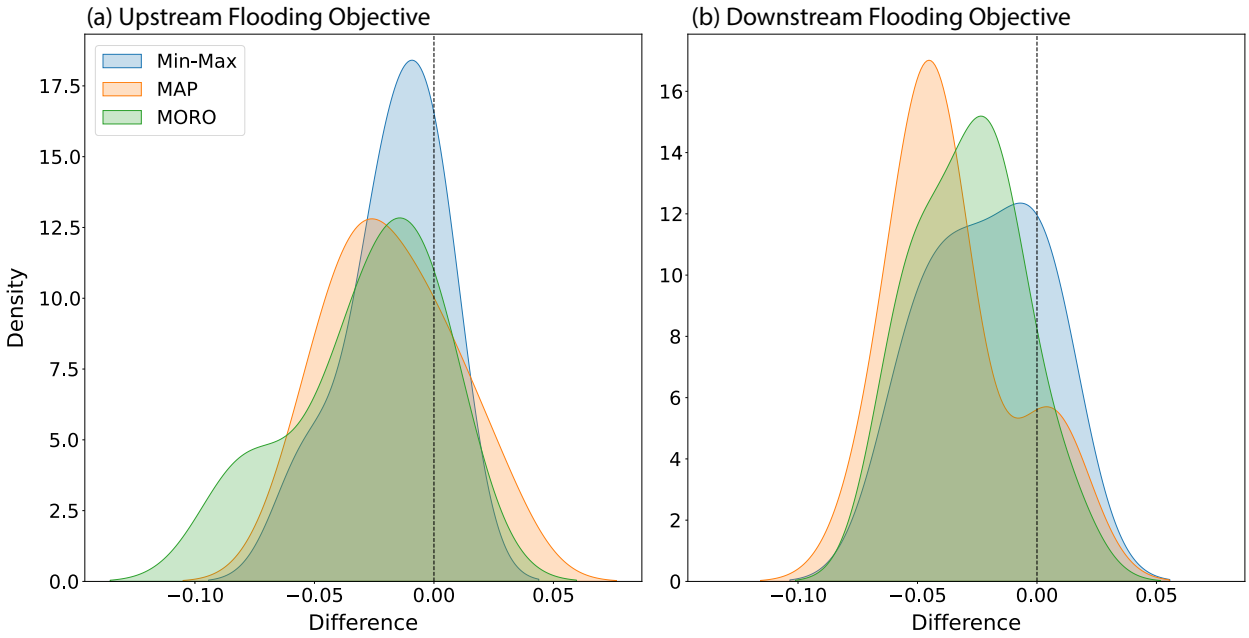

Figure 4.4: KDE plots showing the differences between optimization solutions and their re-simulation over synthetic truth data across MAP, MORO, and Min-Max methods for upstream (a) and downstream (b) flooding objectives

chances of underestimating flooding. However, it is important to note that the Min-Max objective values may be more accurate, but still worse than for the other formulations. As such, this should not be one's only decision criterion.

4.4.4 Understanding Optimized and Re-simulated Policies With Different Optimization Methods

Figures 4.3-4.4 show performance across all policies from each formulation, but stakeholders will ultimately choose only one. Between the three strategies, there is no obvious "winner" in this respect; MAP seems to find the strongest policies for individual objectives, but MORO seems to find the strongest policies in the compromise region Figures 4.3. Assuming stakeholders would like to reduce *both* upstream and downstream flooding, we compare the actions taken by policies that would be selected as compromises

based on the objective values from each strategy's own formulation. Compromise policies were defined as those with the minimum distance to an ideal point with no upstream or downstream flooding. These policies are highlighted in Figure 4.3a, and their true objective values are shown in Figures 4.3b.

First off, we see that the Min-Max policy identified as the best compromise solution actually strongly favors upstream flooding on the synthetic truth. The MORO and MAP compromise policies, however, continue to balance both objectives under the true parameterization relative to the other policies from their formulation. Between these two solutions, MORO dominates MAP, suggesting benefits of considering multiple parameterizations in the optimization. The MORO Compromise is also the closest of these solutions to that of the synthetic truth, suggesting this formulation is most favorable for selecting a compromise.

Similar to the second study (Chapter 3), we also display the time series of states (rainfall forecast and pond depths), actions (orifice openings), and objectives (upstream and downstream flooding) for the selected compromise solutions in Figure 4.5. In each set of plots, panels (a)-(d) show the state variables: the storage pond depths on the left axis, and the perfect 24-hour rainfall forecast on the right axis coming down from the top. Panels (e)-(h) display the resulting actions taken in response to these state variables: the percentage openness of O1 and O2. Finally, panels (i)-(l) present the volume of flooding, with downstream flooding on the left axis and upstream flooding on the right axis. Time series drawn for this analysis cover the majority of the optimization period from mid-June till September, 1995.

First, we investigate the actions of the Synthetic Truth Compromise to see what actions would be taken to balance both objectives if the true parameterization were known. As we saw in chapter 3, this solution operates both ponds similarly, moving the orifice

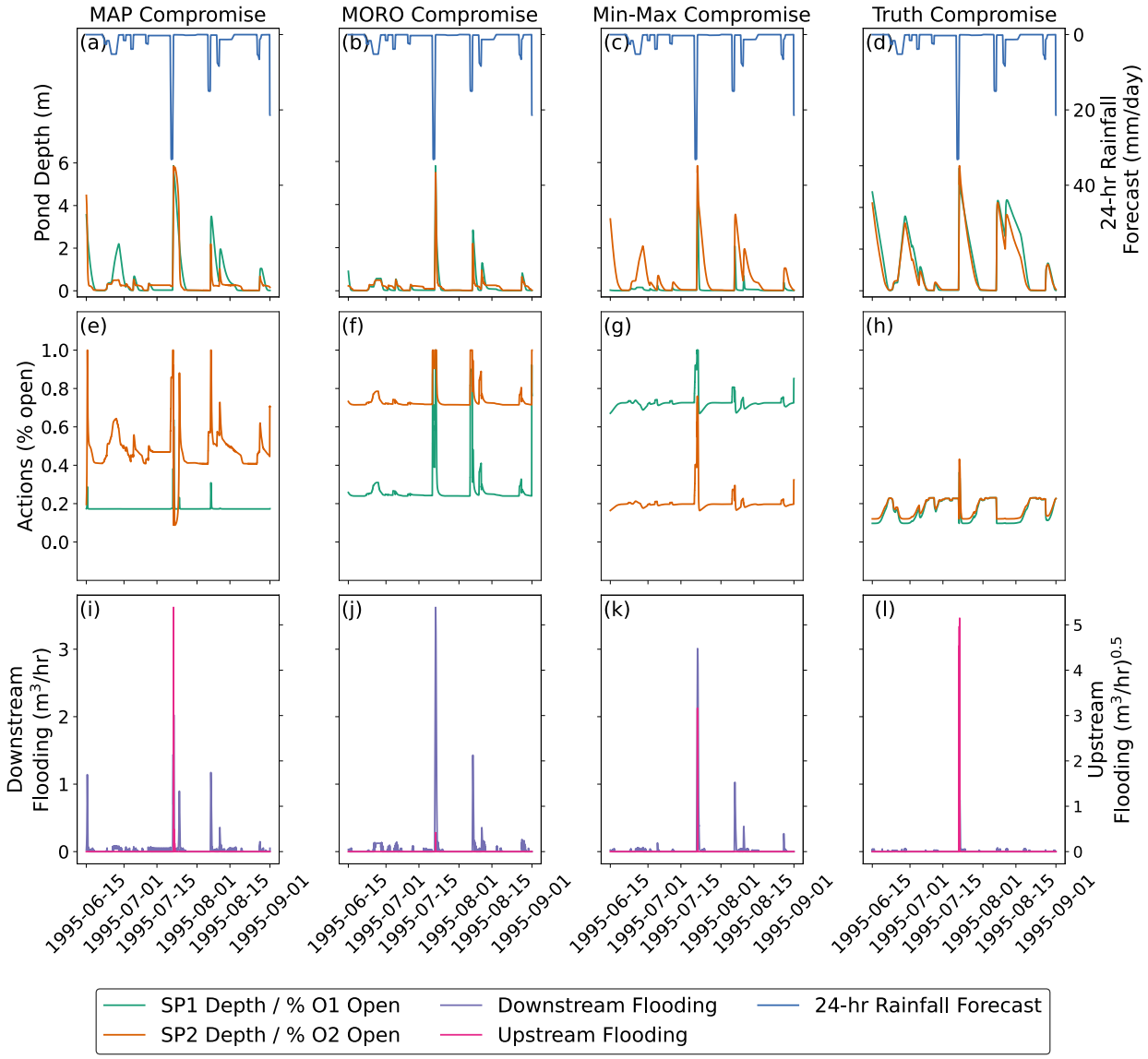

Figure 4.5: Time series of states, actions, and objectives of the MAP, MORO, and Min-Max policies over the period of Jun-Sep 1995. (a=d) Perfect 24-hour rainfall forecast and storage pond depths of SP1 and SP2, (e-h) orifice openness of O1 and O2, and (i-l) upstream and downstream flooding for compromise solutions from MAP, MORO, Min-Max, and the Synthetic Truth

between 15-20% open in each pond in response to changes in the pond depth or forecast (Figure 4.5h), decreasing the opening at the beginning of the storm to reduce downstream flooding, and increasing the opening when full to reduce upstream flooding. This results

in the ponds filling as storms come in (Figure 4.5d), which almost completely prevents downstream flooding (Figure 4.5l). However, only once do the ponds completely fill (Figure 4.5d) causing upstream flooding (Figure 4.5l).

The other three compromise solutions behave differently. Each of them has an "equilibrium" orifice opening that differs for the two ponds to try to balance upstream and downstream flooding. For one of these ponds, it is similarly around 20% for each compromise; however it is higher for the other. The higher equilibrium openness results in faster draining of the corresponding ponds, leading to greater downstream flooding for all of these solutions (Figures 4.5i-l). However, the faster draining generally reduces the chance of filling the ponds and causing overtopping, thereby decreasing Upstream flooding for the MORO and Min-Max Compromise solutions compared to the Truth Compromise (Figures 4.5j-l). This is further facilitated by the orifices opening more to drain the ponds as they fill. For the MORO and Min-Max Compromise Solutions, the opening occurs in unison at each pond (Figures 4.5f-g); however the MAP Compromise generally only changes actions in pond 2, and actually decreases pond openness after first increasing it (Figure 4.5e). This results in greater pond overflow, leading to more upstream flooding for this solution compared to the Truth Compromise despite pond 2 having a higher equilibrium openness that should drain it faster (Figure 4.5i).

These differences highlight the benefits of the robust optimization strategies in identifying compromise policies that can balance both flooding objectives. MORO and Min-Max perform relatively better than MAP, as their actions are more coordinated and dynamic in response to the system's needs. This responsiveness in opening both pond orifices during flood events allows them to drain quickly and reduce the probability of overtopping and causing significant upstream flooding, while maintaining similar downstream flooding from similar equilibrium orifice opening levels.

4.5 CONCLUSIONS AND FUTURE WORK

This work addresses a significant gap in the stormwater literature in designing control rules that account for parametric uncertainty in the stormwater model used for design. While many studies have incorporated uncertainties from climate change or future development scenarios into the design of stormwater systems, none have developed optimization frameworks that are robust to parameter uncertainty. To address this gap, we present two robust optimization approaches, MORO and Min-Max, to optimize real-time control rules that minimize flooding objectives in the studied stormwater system. We compare the performance of both approaches to the traditional MAP optimization approach that does not account for parameter uncertainty based on how each optimization approach performs on a synthetic true parameter set. This allows us to assess which approach is most effective.

The need for robust optimization approaches in designing stormwater control rules is first highlighted in our study by the significant parameter uncertainty in SWMM model parameters discovered through our Bayesian calibration. We find MCMC algorithms sometimes converge to the wrong value, converge to multiple values, or exhibit great uncertainty.This suggests that relying solely on MAP as the most likely parameter set for designing control rules may not the most reliable strategy. Instead, this study underscores the importance of using MORO and Min-Max strategies as robust optimization methods. These approaches incorporate a broader range of likely parameterizations, enabling the design of more robust and reliable control rules compared to the traditional MAP strategy.

Comparing these formulations on our stormwater control problem, we find all solutions tend to underestimate the amount of flooding that would actually occur under the synthetic truth. The worst underestimation of upstream flooding occurred under the MORO formulation, while the worst for downstream flooding occurred for the MAP for-

mulation. The objective values of solutions optimized using the Min-Max strategy were closest to the values achieved on the synthetic truth, providing more accurate estimates of the performance that can be achieved in reality.

However, accuracy is not the only important metric to consider, but performance as well. MAP found solutions that did best on individual objectives when re-evaluated on the synthetic truth, but MORO and Min-Max found better compromise policies. Comparing the time series of states, action values, and flooding objectives for these compromise policies across MAP, MORO, and Min-Max provided valuable insights into the dynamics of these policies. MORO and Min-Max demonstrated more responsive and coordinated actions to state information, showcasing their robustness in adjusting the openness of O1 and O2 to prevent flooding. The MORO compromise solution was most effective, achieved the closest performance to the Synthetic Truth Compromise on both upstream and downstream flooding.

In conclusion, this paper presents two key insights: 1) stormwater models can exhibit great parametric uncertainty, as illustrated by our Bayesian calibration, and 2) using robust optimization to design real-time control rules to be robust to this uncertainty is an effective way to find policies that either 1) provide more accurate objective value estimates (Min-Max optimization) or 2) provide superior compromise performance. These insights could be valuable for decision-makers in stormwater management systems to facilitate the design of control rules under parameter uncertainty. Future work could repeat this experiment using multiple synthetic truth values to ensure the findings themselves are robust across possible true parameterizations. One could also explore including alternative parameters in SWMM beyond those currently addressed in the literature based on a system-specific sensitivity analysis. We hypothesize that this could improve MCMC search performance and convergence. Furthermore, this research opens new avenues

for applying these methodologies to hydrology models beyond SWMM, testing them on more complex models with additional parameters, and scaling them up for larger models.

4.6 CODE AND DATA AVAILABILITY

The code development history for the Bayesian calibration step, as well as the EMODPS optimization strategies (Truth, MAP, MORO, and Min-Max) and post-processing steps, can be found on our GitHub repository.[1]

[1] https://github.com/hosseinkavianih/SWMM-Calibration

CHAPTER 5

Conclusions and Future Work

This dissertation advances our understanding of how to best design stochastic engineered systems to be robust to parametric uncertainty. Three distinct studies are presented that each contribute to this overall mission.

we first advance the diagnostics of MCMC algorithms to visualize their effectiveness, efficiency, reliability, and controllability. This addresses an important gap in the MCMC literature, as existing diagnostics solely focus on diagnosing the effectiveness and efficiency of an individual search process, not on diagnosing its consistency across multiple search processes with different random seeds and hyperparameter configurations. We illustrate the developed diagnostics on three MCMC algorithms - Metropolis Hastings (MH), Adaptive Metropolis (AM), and Differential Evolution Adaptive Metropolis (DREAM) using test problems characterized by high dimensionality and bimodality, characteristics commonly found in engineered systems.

Our published study (Kavianihamedani et al. (2024)) provides key insights into the performance of MCMC algorithms. Most notably, we find DREAM consistently demonstrates superior efficiency and reliability, making it a robust choice for both high-dimensional and multimodal problems. The findings of this paper can inform MCMC algorithm selection for Bayesian inference applications, as well as hyperparameterization of the chosen algorithm. Future work could focus on enhancing the DREAM algorithm by adapting the probability of using its different proposal operators based on their success rates in proposing new chain locations that are accepted. We hypothesize that this adjustment could improve DREAM's controllability and overall search performance.

In the second study, we highlight the importance of finding the Pareto front of optimal solutions to better understand the trade-offs in stormwater management systems. For the

first time, we introduce Evolutionary Multi-Objective Direct Policy Search (EMODPS), a policy-based reinforcement learning (RL) method, to the stormwater literature and compare its performance to Deep Deterministic Policy Gradients (DDPG), an actor-critic RL method that includes both policy optimization and value approximation and has been used to optimize stormwater management rules. We benchmark these two algorithms in optimizing real-time control (RTC) rules for a stylized system consisting of two storage ponds for urban stormwater, each with a bottom orifice to control the outflow rate for flood control. The objectives are to minimize flooding both upstream (at the storage ponds) and downstream.

Findings from the second study demonstrate that EMODPS generates more robust trade-off policies than DDPG with simpler hyperparameter tuning, making it a preferred choice for real-world, multi-objective applications. Insights from this study can guide decision-makers in designing effective stormwater control rules to minimize flooding in stormwater management systems. Future work can investigate whether EMODPS outperforms DDPG with more than two objectives. However, we hypothesize that EMODPS's advantages will increase over DDPG due to its built-in non-dominated sorting of policies, whereas DDPG will face the curse of dimensionality.

Finally, in the last study, we bridge the insights from both the first and second studies to design robust, multi-objective stormwater control rules that address parametric uncertainty in hydrological models. Leveraging the findings from the first paper (Kavianihamedani et al. (2024)), we use DREAM to calibrate the SWMM model parameters and account for their uncertainty. Capitalizing on the second study, we then use EMODPS to design control rules for the system that account for this uncertainty. For that, we introduce two robust optimization methods, MORO and Min-Max, and compare their performance with the traditional MAP method.

The findings of this last study demonstrate the benefits of leveraging robust optimization in stormwater control systems. We observe MORO derives the best compromise policy and closest one to the compromise from the Pareto set optimized to the synthetic truth. We also find Min-Max re-simulated policies over the synthetic true parameter set are closest in performance to the optimized objective values among all three strategies. Insights of this study could be valuable for decision makers in stormwater management systems to improve the design of control rules under parameter uncertainty. For example, stakeholders can use this study's findings to choose which optimization approach to use to account for parameter uncertainty: MORO to find a strong compromise or Min-Max to have a more accurate estimate of performance. Future work could repeat this experiment using multiple synthetic true parameter values to ensure the findings themselves are robust to this assumption.

APPENDIX A

Appendix

A.1 FIGURES

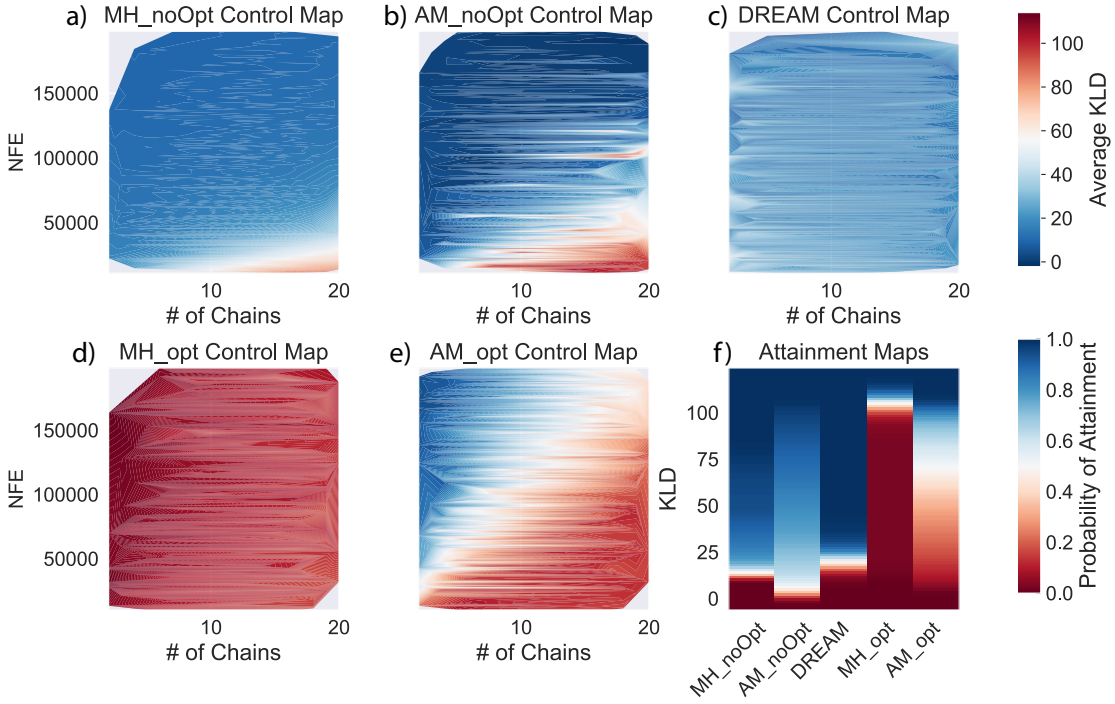

Figure A.1: (a-e) Control maps illustrating the average Kullback-Leibler Divergence (KLD) across random seeds on the 100D MVN test problem as a function of the number of function evaluations (NFE) and number of chains for.(f) Attainment maps illustrating the probability of attaining different KLDs across all seeds and hyperparameters for each algorithm.

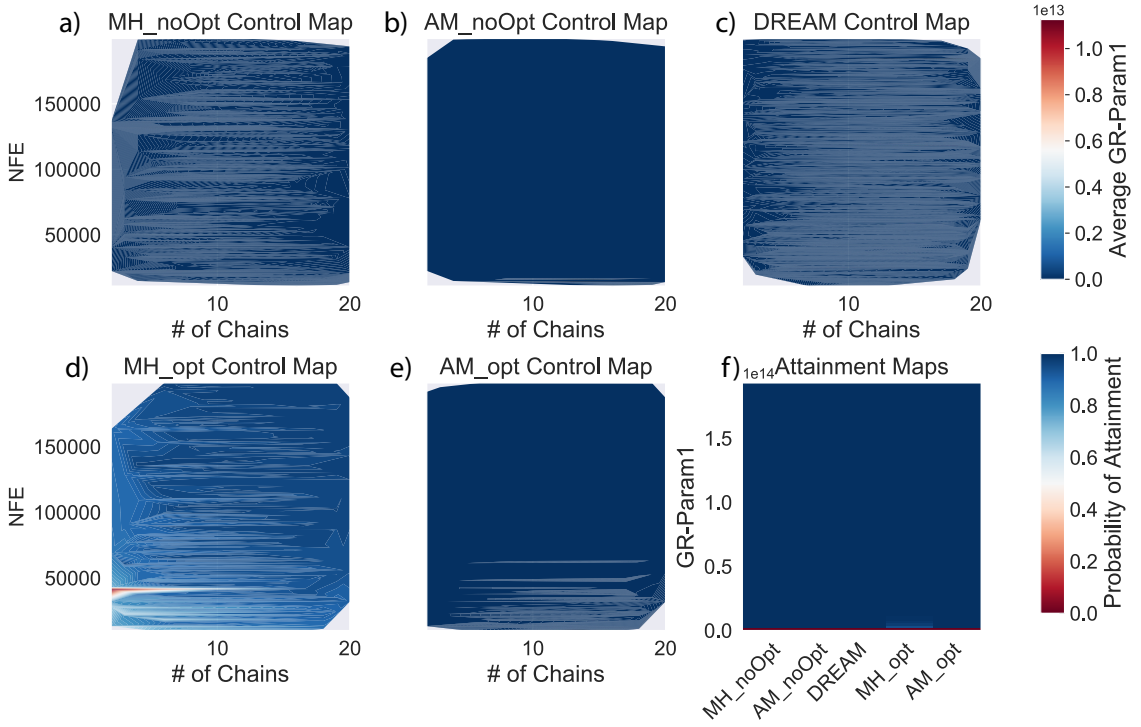

Figure A.2: (a-e) Control maps illustrating the average Gelman-Rubin (GR) diagnostic of the first dimension of the 100D MVN test problem across random seeds as a function of the number of function evaluations (NFE) and number of chains.(f) Attainment maps illustrating the probability of attaining different WDs across all seeds and hyperparameters for each algorithm.

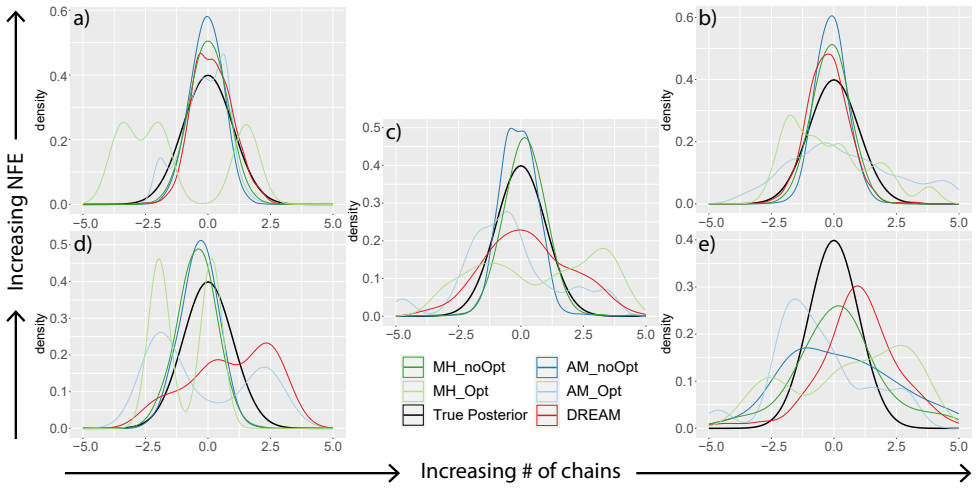

Figure A.3: Posterior marginals of the 1st dimension of the 100D MVN test problem when using the hyperparameter closest to (a) the least number of chains and the most NFE, (b) the highest number of chains and the most NFE, (c) the median number of chains and the median NFE, (d) the least number of chains and the least NFE, and (e) the most number of chains and the least NFE.

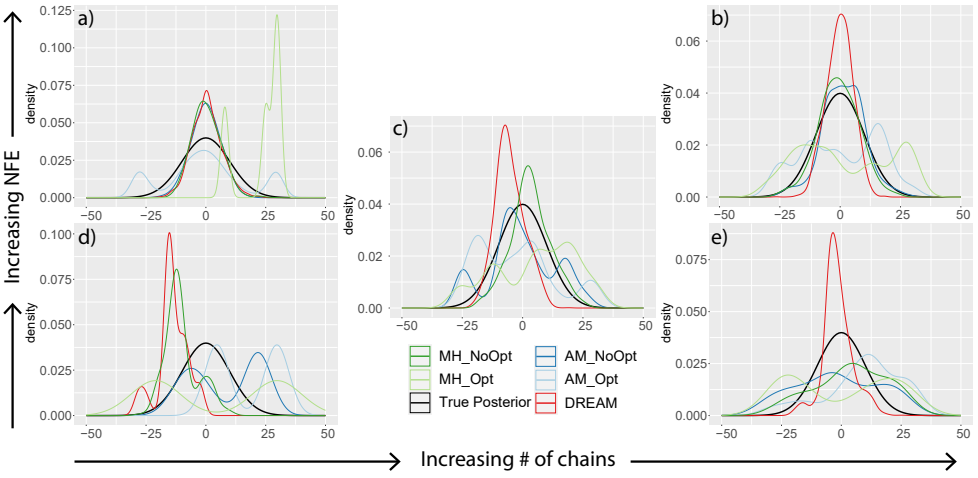

Figure A.4: Posterior marginals of the 100th dimension of the 100D MVN test problem when using the hyperparameter closest to (a) the least number of chains and the most NFE, (b) the highest number of chains and the most NFE, (c) the median number of chains and the median NFE, (d) the least number of chains and the least NFE, and (e) the most number of chains and the least NFE.

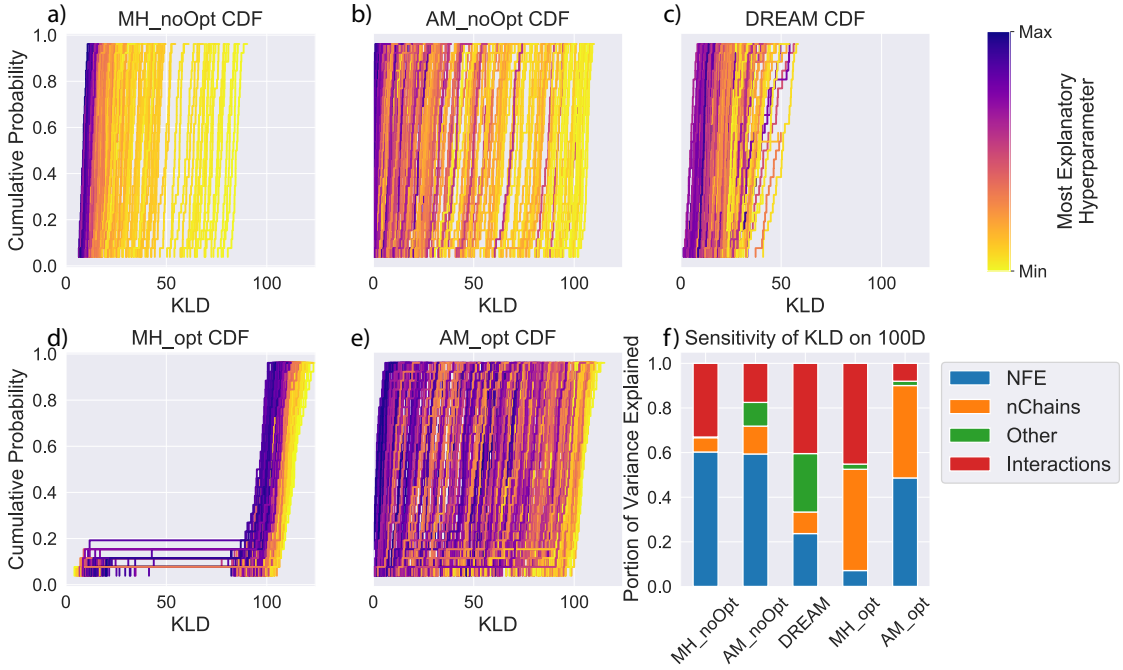

Figure A.5: (a-e) Cumulative distribution functions (CDFs) of Kullback-Leibler Divergence (KLD) across random seeds for each hyperparameter on the 100D MVN test problem. The color of the hyperparameter indicates the value of the parameter to which that algorithm's WD was most sensitive. (f) Decomposition of how much variablity in KLD is explained by each hyperparameter and their interaction for each algorithm.

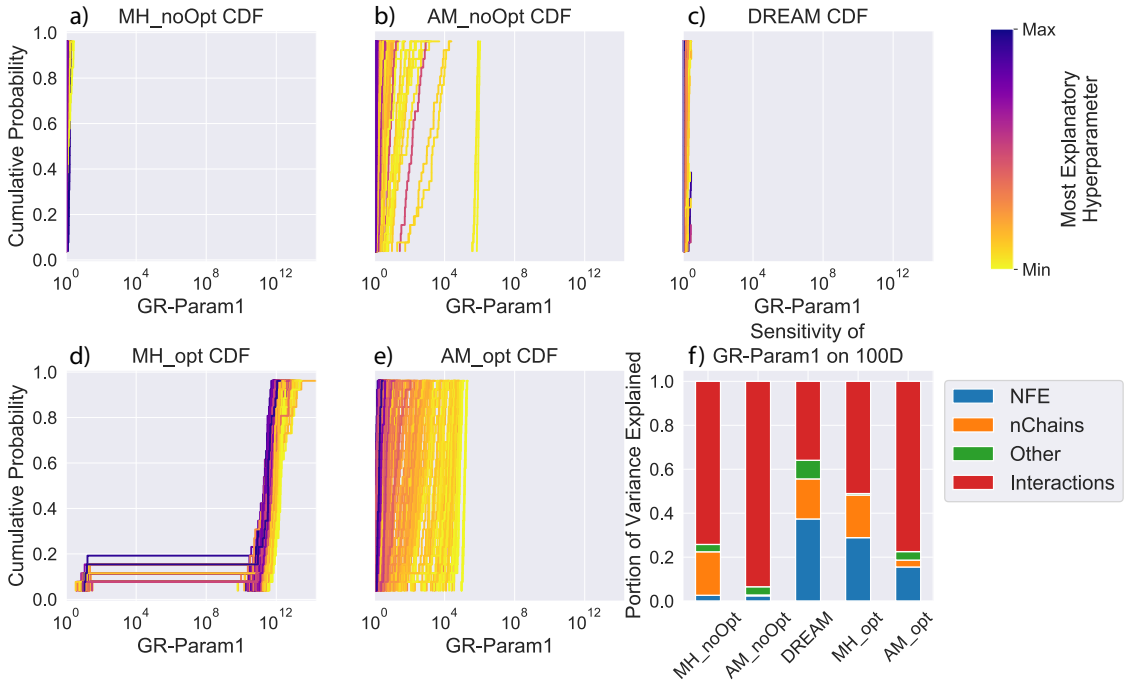

Figure A.6: (a-e) Cumulative distribution functions (CDFs) of the Gelman-Rubin (GR) diagnostic of the first dimension across random seeds for each hyperparameter on the 100D MVN test problem. The color of the hyperparameter indicates the value of the parameter to which that algorithm's WD was most sensitive. (f) Decomposition of how much variablity in GR is explained by each hyperparameter and their interaction for each algorithm.

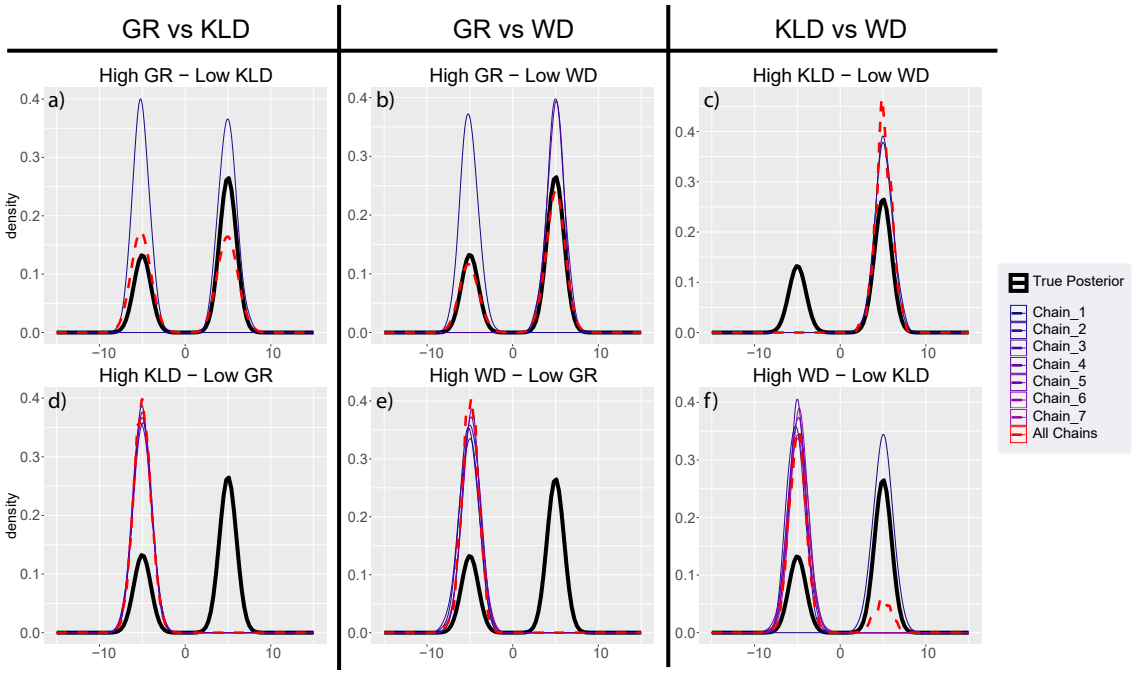

Figure A.7: Comparison of the estimated marginal posterior of the 5th dimension of the 10D bimodal test problem from individual chains and across chains using select hyper-parameter sets with (a) a high GR and low KLD and (d) the reverse; (b) a high GR and low WD and (e) the reverse; and (c) a high KLD and low WD and (f) the reverse.

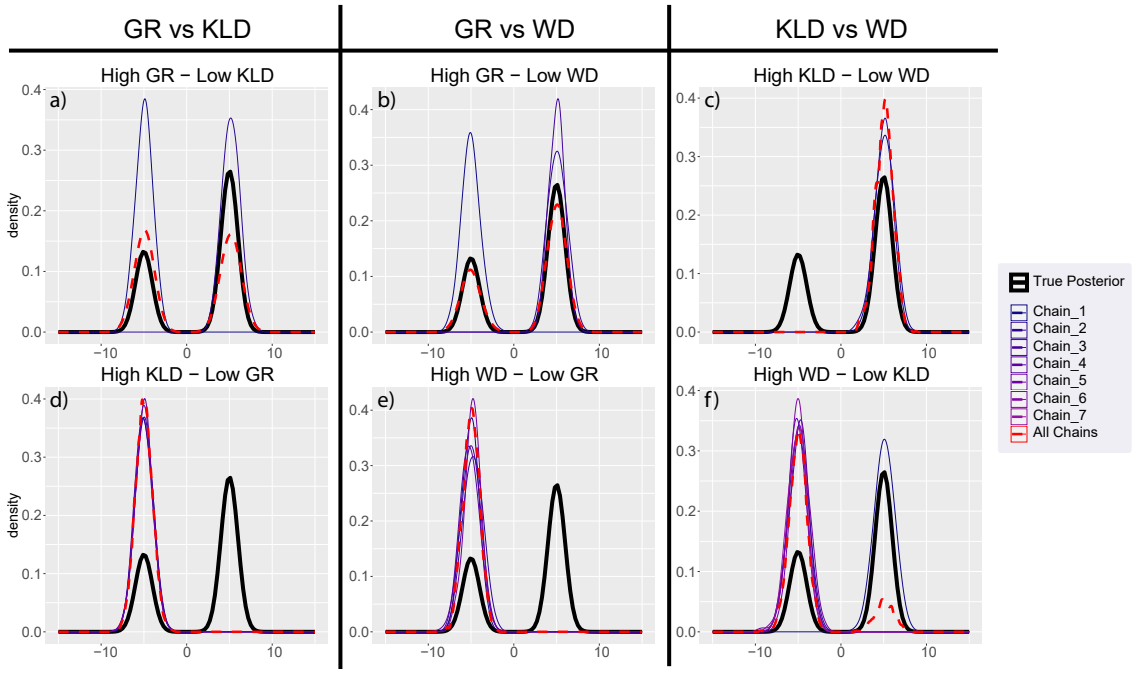

Figure A.8: Comparison of the estimated marginal posterior of the 10th dimension of the 10D bimodal test problem from individual chains and across chains using select hyperparameter sets with (a) a high GR and low KLD and (d) the reverse; (b) a high GR and low WD and (e) the reverse; and (c) a high KLD and low WD and (f) the reverse.

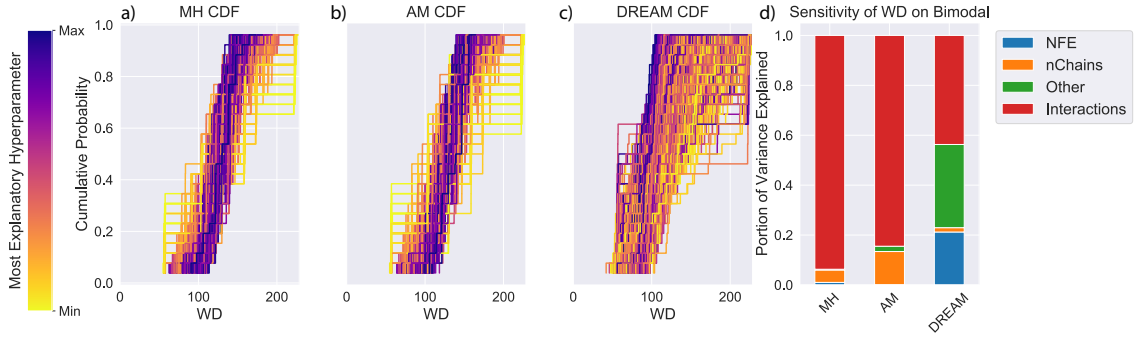

Figure A.9: (a-c) Cumulative distribution functions (CDFs) of Wasserstein distance (WD) across random seeds for each hyperparameter on the 10D bimodal problem. The color of the hyperparameter indicates the value of the parameter to which that algorithm's WD was most sensitive. (d) Decomposition of how much variablity in WD is explained by each hyperparameter and their interaction for each algorithm.

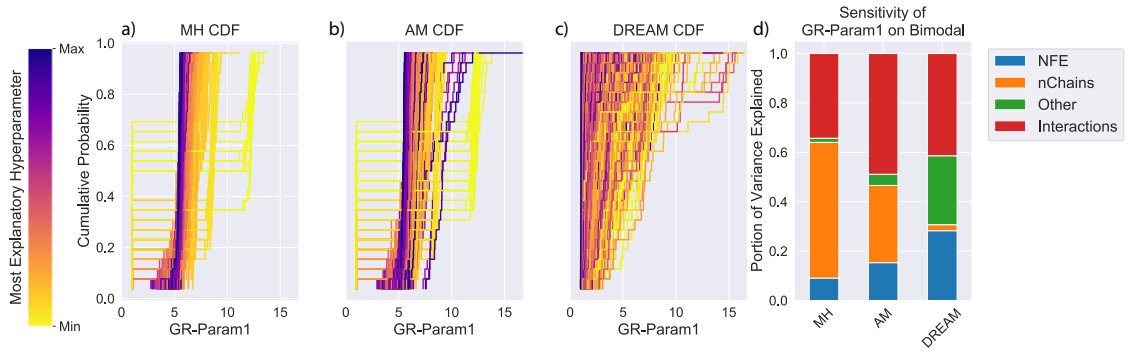

Figure A.10: (a-c) Cumulative distribution functions (CDFs) of the Gelman-Rubin (GR) diagnostic of the first dimension of the 10D bimodal problem across random seeds for each hyperparameter. The color of the hyperparameter indicates the value of the parameter to which that algorithm's WD was most sensitive. (d) Decomposition of how much variablity in WD is explained by each hyperparameter and their interaction for each algorithm.

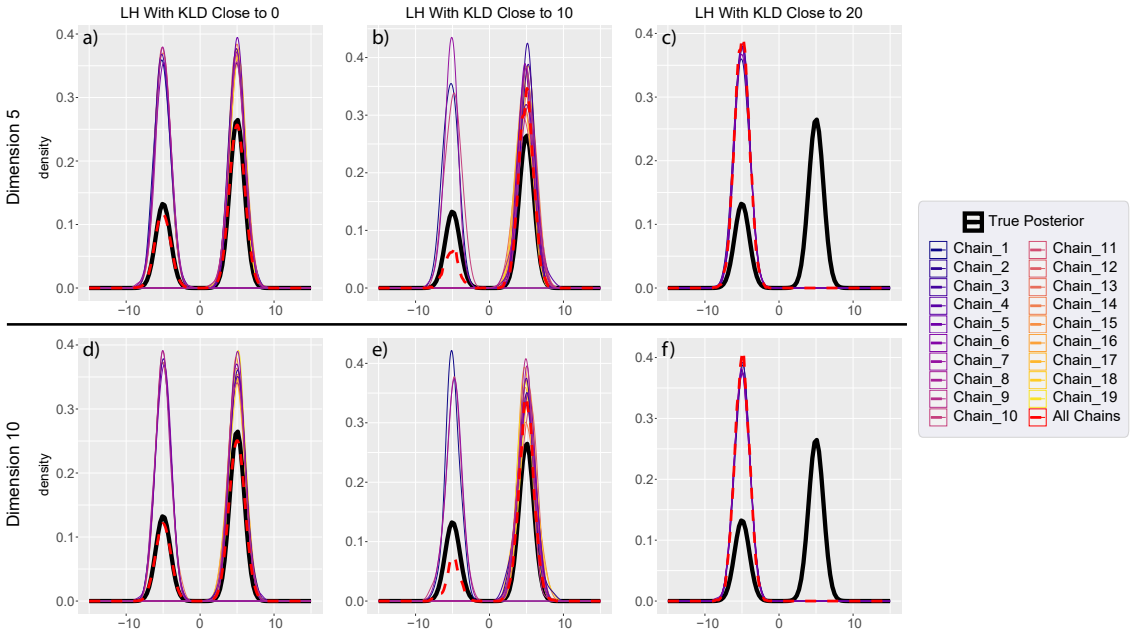

Figure A.11: Comparison of the estimated marginal posterior of (a-c) the 5th dimension of the 10D bimodal test problem and (d-f) the 10th dimension from individual chains and across chains using select hyperparameter sets with (a,d) KLD near 0, (b,e) KLD near 10 and (c,f) KLD near 20.

APPENDIX B

Appendix

B.1 FIGURES

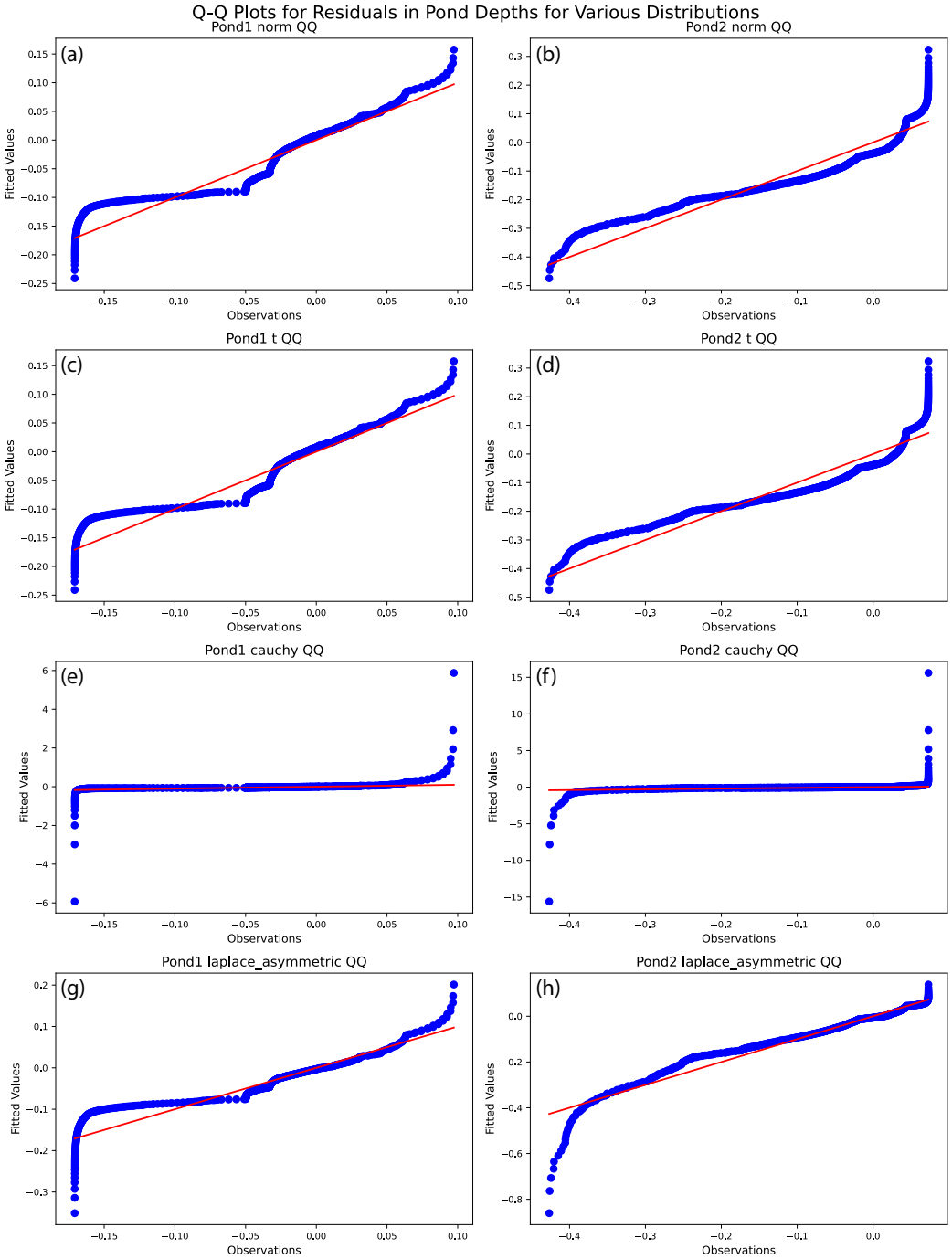

Figure B.1: Quantile-quantile plots of residuals in depths at ponds 1 and 2 simulated by SWMM with one Latin hypercube sample of parameter values over their ranges compared to the synthetic truth when using (a-b) normal distribution, (c-d) Student-t distribution, (e-f) Cauchy distribution, (g-h) Asymmetric Laplace distribution.

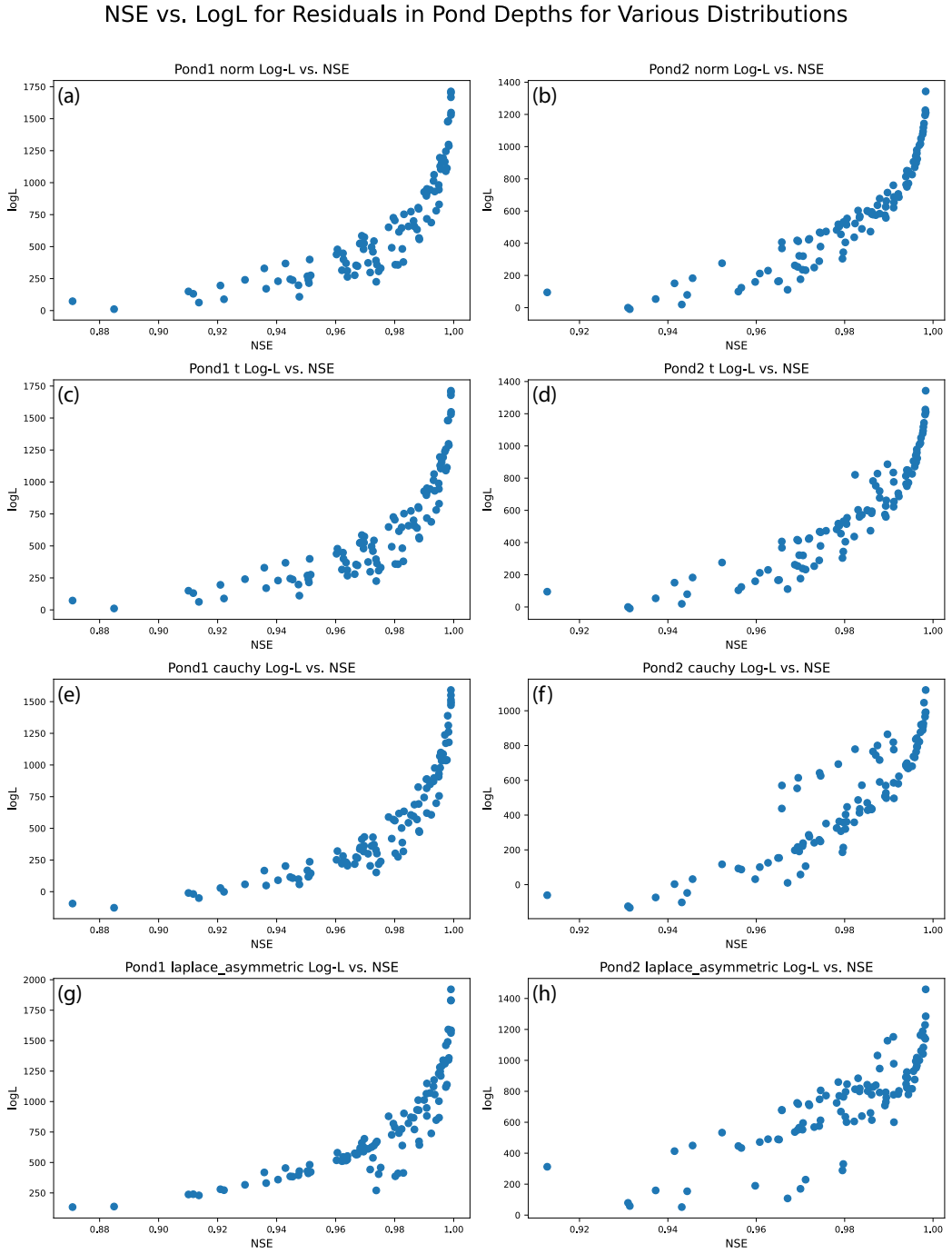

Figure B.2: Log-likelihood of residuals in depths at ponds 1 and 2 vs. Nash-Sutcliffe Efficiency (NSE) between simulations and observations under one Latin hypercube sample of SWMM parameters if using (a-b) normal likelihood function, (c-d) Student-t likelihood function, (e-f) Cauchy likelihood function, (g-h) Asymmetric Laplace likelihood function.

BIBLIOGRAPHY

Abels, A., Roijers, D., Lenaerts, T., Nowé, A., & Steckelmacher, D. (2019). Dynamic weights in multi-objective deep reinforcement learning. In *International conference on machine learning*, (pp. 11–20). PMLR.

Ashraf, N. M., Mostafa, R. R., Sakr, R. H., & Rashad, M. (2021). Optimizing hyperparameters of deep reinforcement learning for autonomous driving based on whale optimization algorithm. *Plos one*, *16*(6), e0252754.

Bahrami, M., Bozorg-Haddad, O., & Loáiciga, H. A. (2019). Optimizing stormwater low-impact development strategies in an urban watershed considering sensitivity and uncertainty. *Environmental monitoring and assessment*, *191*, 1–14.

Bartholomew, E., & Kwakkel, J. H. (2020). On considering robustness in the search phase of robust decision making: A comparison of many-objective robust decision making, multi-scenario many-objective robust decision making, and many objective robust optimization. *Environmental Modelling & Software*, *127*, 104699.

Beven, K., & Binley, A. (1992). The future of distributed models: model calibration and uncertainty prediction. *Hydrological processes*, *6*(3), 279–298.

Beven, K., & Binley, A. (2014). Glue: 20 years on. *Hydrological processes*, *28*(24), 5897–5918.

Blum, A. G., Ferraro, P. J., Archfield, S. A., & Ryberg, K. R. (2020). Causal effect of impervious cover on annual flood magnitude for the united states. *Geophysical Research Letters*, *47*(5), no–no.

Boltz, S., Debreuve, E., & Barlaud, M. (2007). knn-based high-dimensional kullback-leibler distance for tracking. In *Eighth International Workshop on Image Analysis for Multimedia Interactive Services (WIAMIS'07)*, (pp. 16–16). IEEE.

Boltz, S., Debreuve, E., & Barlaud, M. (2009). High-dimensional statistical measure for region-of-interest tracking. *IEEE Transactions on Image Processing*, *18*(6), 1266–1283.

Boone, E. L., Merrick, J. R., & Krachey, M. J. (2014). A hellinger distance approach to mcmc diagnostics. *Journal of Statistical Computation and Simulation*, *84*(4), 833–849.

Bowes, B. D., Tavakoli, A., Wang, C., Heydarian, A., Behl, M., Beling, P. A., & Goodall, J. L. (2021). Flood mitigation in coastal urban catchments using real-time stormwater infrastructure control and reinforcement learning. *Journal of Hydroinformatics*, *23*(3), 529–547.

Bowes, B. D., Wang, C., Ercan, M. B., Culver, T. B., Beling, P. A., & Goodall, J. L. (2022). Reinforcement learning-based real-time control of coastal urban stormwater systems to mitigate flooding and improve water quality. *Environmental Science: Water Research & Technology*, *8*(10), 2065–2086.

Brent, R. P. (1971). An algorithm with guaranteed convergence for finding a zero of a function. *The computer journal*, *14*(4), 422–425.

Brooks, S. P., & Gelman, A. (1998). General methods for monitoring convergence of iterative simulations. *Journal of computational and graphical statistics*, *7*(4), 434–455.

Cacchiani, V., Qi, J., & Yang, L. (2020). Robust optimization models for integrated train stop planning and timetabling with passenger demand uncertainty. *Transportation Research Part B: Methodological*, *136*, 1–29.

Campbell, T., & How, J. P. (2015). Bayesian nonparametric set construction for robust optimization. In *2015 American Control Conference (ACC)*, (pp. 4216–4221). IEEE.

Cannon, A. J. (2018). Multivariate quantile mapping bias correction: an n-dimensional probability density function transform for climate model simulations of multiple variables. *Climate dynamics*, *50*, 31–49.

Castelletti, A., & Soncini-Sessa, R. (2007). Bayesian networks and participatory modelling in water resource management. *Environmental Modelling & Software*, *22*(8), 1075–1088.

Chankong, V., & Haimes, Y. Y. (2008). *Multiobjective decision making: theory and methodology*. Courier Dover Publications.

Chiandussi, G., Codegone, M., Ferrero, S., & Varesio, F. E. (2012). Comparison of multiobjective optimization methodologies for engineering applications. *Computers & Mathematics with Applications*, *63*(5), 912–942.

Choi, J.-H., & Lee, J.-S. (2019). Embracenet: A robust deep learning architecture for multimodal classification. *Information Fusion*, *51*, 259–270.

Coello, C. A. C. (2018). Multi-objective optimization.

Coumou, D., & Rahmstorf, S. (2012). A decade of weather extremes. *Nature climate change*, *2*(7), 491–496.

Cowles, M. K., & Carlin, B. P. (1996). Markov chain monte carlo convergence diagnostics: a comparative review. *Journal of the American Statistical Association*, *91*(434), 883–904.

Cuturi, M. (2013). Sinkhorn distances: Lightspeed computation of optimal transport. *Advances in neural information processing systems*, *26*.

Demographia (????). Urbanization in the united states from 1945. http://demographia.com/db-1945uza.htm. Accessed: 2020-09-07.

Di Matteo, M., Dandy, G. C., & Maier, H. R. (2017). Multiobjective optimization of distributed stormwater harvesting systems. *Journal of Water Resources Planning and Management*, *143*(6), 04017010.

Ding, Z., Huang, Y., Yuan, H., & Dong, H. (2020). Introduction to reinforcement learning. *Deep reinforcement learning: fundamentals, research and applications*, (pp. 47–123).

Dittrich, R., Wreford, A., & Moran, D. (2016). A survey of decision-making approaches for climate change adaptation: Are robust methods the way forward? *Ecological Economics*, *122*, 79–89.

Dixit, A., & Roy, V. (2017). Mcmc diagnostics for higher dimensions using kullback leibler divergence. *Journal of Statistical Computation and Simulation*, *87*(13), 2622–2638.

Dobrushin, R. L. (1970). Prescribing a system of random variables by conditional distributions. *Theory of Probability & Its Applications*, *15*(3), 458–486.

Dotto, C. B., Mannina, G., Kleidorfer, M., Vezzaro, L., Henrichs, M., McCarthy, D. T., Freni, G., Rauch, W., & Deletic, A. (2012). Comparison of different uncertainty techniques in urban stormwater quantity and quality modelling. *Water research*, *46*(8), 2545–2558.

Dotto, C. B. S., Kleidorfer, M., Deletic, A., Rauch, W., & McCarthy, D. T. (2014). Impacts of measured data uncertainty on urban stormwater models. *Journal of hydrology*, *508*, 28–42.

Duan, Y., Chen, X., Houthooft, R., Schulman, J., & Abbeel, P. (2016). Benchmarking deep reinforcement learning for continuous control. In *International conference on machine learning*, (pp. 1329–1338). PMLR.

Durmus, A., & Moulines, É. (2015). Quantitative bounds of convergence for geometrically ergodic markov chain in the wasserstein distance with application to the metropolis adjusted langevin algorithm. *Statistics and Computing*, *25*, 5–19.

Gelman, A., Carlin, J. B., Stern, H. S., Dunson, D. B., Vehtari, A., & Rubin, D. B. (2013). *Bayesian data analysis*. CRC press.

Gelman, A., Roberts, G. O., Gilks, W. R., et al. (1996). Efficient metropolis jumping rules. *Bayesian statistics*, *5*(599-608), 42.

Gelman, A., & Rubin, D. B. (1992). Inference from iterative simulation using multiple sequences. *Statistical science*, *7*(4), 457–472.

Geyer, C. J. (2011). Introduction to markov chain monte carlo. *Handbook of markov chain monte carlo*, *20116022*, 45.

Giles, D. E., et al. (2010). Hermite regression analysis of multi-modal count data. *Economics Bulletin*, *30*(4), 2936–2945.

Gilks, W. R., Roberts, G. O., & George, E. I. (1994). Adaptive direction sampling. *Journal of the Royal Statistical Society: Series D (The Statistician)*, *43*(1), 179–189.

Giuliani, M., & Castelletti, A. (2016). Is robustness really robust? how different definitions of robustness impact decision-making under climate change. *Climatic Change*, *135*, 409–424.

Giuliani, M., Castelletti, A., Pianosi, F., Mason, E., & Reed, P. M. (2016). Curses, tradeoffs, and scalable management: Advancing evolutionary multiobjective direct policy search to improve water reservoir operations. *Journal of Water Resources Planning and Management*, *142*(2), 04015050.

Giuliani, M., Lamontagne, J., Reed, P., & Castelletti, A. (2021). A state-of-the-art review of optimal reservoir control for managing conflicting demands in a changing world. *Water Resources Research*, *57*(12), e2021WR029927.

Giuliani, M., Quinn, J. D., Herman, J. D., Castelletti, A., & Reed, P. M. (2017). Scalable multiobjective control for large-scale water resources systems under uncertainty. *IEEE Transactions on Control Systems Technology*, *26*(4), 1492–1499.

Haario, H., Laine, M., Mira, A., & Saksman, E. (2006). Dram: efficient adaptive mcmc. *Statistics and computing*, *16*, 339–354.

Haario, H., Saksman, E., & Tamminen, J. (2001). An adaptive Metropolis algorithm. *Bernoulli*, *7*(2), 223–242.

Hadka, D., & Reed, P. (2012). Diagnostic assessment of search controls and failure modes in many-objective evolutionary optimization. *Evolutionary computation*, *20*(3), 423–452.

Hadka, D., & Reed, P. (2013). Borg: An auto-adaptive many-objective evolutionary computing framework. *Evolutionary computation*, *21*(2), 231–259.

Hao, J., Yang, T., Tang, H., Bai, C., Liu, J., Meng, Z., Liu, P., & Wang, Z. (2023). Exploration in deep reinforcement learning: From single-agent to multiagent domain. *IEEE Transactions on Neural Networks and Learning Systems*.

Hartig, F., Minunno, F., & Paul, S. (2023). *BayesianTools: General-Purpose MCMC and SMC Samplers and Tools for Bayesian Statistics*. R package version 0.1.8, https://github.com/florianhartig/BayesianTools.

Hastings, W. K. (1970). Monte carlo sampling methods using markov chains and their applications. *Biometrika*, *57*, 97.

Hayes, C. F., Rădulescu, R., Bargiacchi, E., Källström, J., Macfarlane, M., Reymond, M., Verstraeten, T., Zintgraf, L. M., Dazeley, R., Heintz, F., et al. (2022). A practical guide to multi-objective reinforcement learning and planning. *Autonomous Agents and Multi-Agent Systems*, *36*(1), 26.

Heaton, J., Polson, N. G., & Witte, J. H. (2016). Deep learning in finance. *arXiv preprint arXiv:1602.06561*.

Heidelberger, P., & Welch, P. D. (1983). Simulation run length control in the presence of an initial transient. *Operations Research*, *31*(6), 1109–1144.

Herman, J., & Usher, W. (2017). Salib: An open-source python library for sensitivity analysis. *Journal of Open Source Software*, *2*(9), 97.

Herrera, F., Lozano, M., et al. (1996). Adaptation of genetic algorithm parameters based on fuzzy logic controllers. *Genetic Algorithms and Soft Computing*, *8*(1996), 95–125.

Hobbs, B. F., Chankong, V., Hamadeh, W., & Stakhiv, E. Z. (1992). Does choice of multicriteria method matter? an experiment in water resources planning. *Water Resources Research*, *28*(7), 1767–1779.

Hou, Y., Liu, L., Wei, Q., Xu, X., & Chen, C. (2017). A novel ddpg method with prioritized experience replay. In *2017 IEEE international conference on systems, man, and cybernetics (SMC)*, (pp. 316–321). IEEE.

James, W. (2003). Rules for responsible modeling. Tech. rep., CHI Guelph, Ontario.

Jeffreys, H. (1948). Theory of probability, section 3.23.

Jia, Y., & Culver, T. B. (2006). Robust optimization for total maximum daily load allocations. *Water Resources Research*, *42*(2).

Jiang, Q., Su, H., Liu, Y., Zou, R., Ye, R., & Guo, H. (2017). Parameter uncertainty-based pattern identification and optimization for robust decision making on watershed load reduction. *Journal of hydrology*, *547*, 708–717.

Jin, Y., Olhofer, M., & Sendhoff, B. (2001). Dynamic weighted aggregation for evolutionary multi-objective optimization: Why does it work and how. In *Proceedings of the genetic and evolutionary computation conference*, (pp. 1042–1049).

Jones, G. L., & Hobert, J. P. (2001). Honest exploration of intractable probability distributions via markov chain monte carlo. *Statistical Science*, (pp. 312–334).

Kantorovich, L. V. (1960). Mathematical methods of organizing and planning production. *Management science*, *6*(4), 366–422.

Kavianihamedani, H., Quinn, J. D., & Smith, J. D. (2024). New diagnostic assessment of mcmc algorithm effectiveness, efficiency, reliability, and controllability. *IEEE Access*, *12*, 42385–42400.

Kiran, M., & Ozyildirim, M. (2022). Hyperparameter tuning for deep reinforcement learning applications. *arXiv preprint arXiv:2201.11182*.

Kleidorfer, M., Deletic, A., Fletcher, T., & Rauch, W. (2009). Impact of input data uncertainties on urban stormwater model parameters. *Water Science and Technology*, *60*(6), 1545–1554.

Konak, A., Coit, D. W., & Smith, A. E. (2006). Multi-objective optimization using genetic algorithms: A tutorial. *Reliability engineering & system safety*, *91*(9), 992–1007.

Konda, V., & Tsitsiklis, J. (1999). Actor-critic algorithms. *Advances in neural information processing systems*, *12*.

Koutsoyiannis, D., & Economou, A. (2003). Evaluation of the parameterization-simulation-optimization approach for the control of reservoir systems. *Water Resources Research*, *39*(6).

Kullback, S., & Leibler, R. A. (1951). On information and sufficiency. *The annals of mathematical statistics*, *22*(1), 79–86.

Kumar, S., Guntu, R. K., Agarwal, A., Villuri, V. G. K., Pasupuleti, S., Kaushal, D. R., Gosian, A. K., & Bronstert, A. (2022). Multi-objective optimization for stormwater management by green-roofs and infiltration trenches to reduce urban flooding in central delhi. *Journal of Hydrology*, *606*, 127455.

Kwakkel, J. H., Haasnoot, M., & Walker, W. E. (2015). Developing dynamic adaptive policy pathways: a computer-assisted approach for developing adaptive strategies for a deeply uncertain world. *Climatic Change*, *132*, 373–386.

Laloy, E., & Vrugt, J. A. (2012). High-dimensional posterior exploration of hydrologic models using multiple-try dream (zs) and high-performance computing. *Water Resources Research*, *48*(1).

Lempert, R. J. (2003). *Shaping the next one hundred years: new methods for quantitative, long-term policy analysis*. Rand Corporation.

Lempert, R. J., Bryant, B. P., Collins, M. T., Hackbarth, A., LaTourrette, T., Reville, R. T., Popper, S. W., Mijere, C., Groves, D. G., Keller, K., et al. (2013). Making good decisions without predictions: Robust decision making for planning under deep uncertainty.

Li, Z., Wen, X., Lu, Z., & Jing, W. (2022). A ddpg-based transfer learning optimization framework for user association and power control in hetnet. In *2022 IEEE International Conference on Communications Workshops (ICC Workshops)*, (pp. 343–348). IEEE.

Liang, C., & Mahadevan, S. (2015). Bayesian sensitivity analysis and uncertainty integration for robust optimization. *Journal of Aerospace Information Systems*, *12*(1), 189–203.

Liessner, R., Schmitt, J., Dietermann, A., & Bäker, B. (2019). Hyperparameter optimization for deep reinforcement learning in vehicle energy management. In *ICAART (2)*, (pp. 134–144).

Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., & Wierstra, D. (2015). Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*.

Lim, A. E., Shanthikumar, J. G., & Shen, Z. M. (2006). Model uncertainty, robust optimization, and learning. In *Models, Methods, and Applications for Innovative Decision Making*, (pp. 66–94). INFORMS.

Lin, J. G. (2005). On min-norm and min-max methods of multi-objective optimization. *Mathematical programming*, *103*(1), 1–33.

Liu, C., Xu, X., & Hu, D. (2014). Multiobjective reinforcement learning: A comprehensive overview. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, *45*(3), 385–398.

Mandur, J., & Budman, H. (2012). A polynomial-chaos based algorithm for robust optimization in the presence of bayesian uncertainty. *IFAC Proceedings Volumes*, *45*(15), 549–554.

Mandur, J., & Budman, H. (2014). Robust optimization of chemical processes using bayesian description of parametric uncertainty. *Journal of Process Control*, *24*(2), 422–430.

Mann, M. E., & Park, J. (1994). Global-scale modes of surface temperature variability on interannual to century timescales. *Journal of Geophysical Research: Atmospheres*, *99*(D12), 25819–25833.

Marchau, V. A., Walker, W. E., Bloemen, P. J., & Popper, S. W. (2019). *Decision making under deep uncertainty: from theory to practice*. Springer Nature.

McCuen, R. H., Knight, Z., & Cutter, A. G. (2006). Evaluation of the nash–sutcliffe efficiency index. *Journal of hydrologic engineering*, *11*(6), 597–602.

McDonnell, B. E., Ratliff, K., Tryby, M. E., Wu, J. J. X., & Mullapudi, A. (2020). Pyswmm: the python interface to stormwater management model (swmm). *Journal of open source software*, *5*(52), 1.

McKinnon, K. I. (1998). Convergence of the nelder–mead simplex method to a nonstationary point. *SIAM Journal on optimization*, *9*(1), 148–158.

McPhail, C., Maier, H. R., Kwakkel, J. H., Giuliani, M., Castelletti, A., & Westra, S. (2018). Robustness metrics: How are they calculated, when should they be used and why do they give different results? *Earth's Future*, *6*(2), 169–191.

Metropolis, N., Rosenbluth, A. W., Rosenbluth, M. N., Teller, A. H., & Teller, E. (1953). Equation of state calculations by fast computing machines. *The journal of chemical physics*, *21*(6), 1087–1092.

Montanari, A. (2005). Large sample behaviors of the generalized likelihood uncertainty estimation (glue) in assessing the uncertainty of rainfall-runoff simulations. *Water resources research*, *41*(8).

Mullapudi, A., Lewis, M. J., Gruden, C. L., & Kerkez, B. (2020). Deep reinforcement learning for the real time control of stormwater systems. *Advances in water resources*, *140*, 103600.

Mykland, P., Tierney, L., & Yu, B. (1995). Regeneration in markov chain samplers. *Journal of the American Statistical Association*, *90*(429), 233–241.

Nachum, O., Norouzi, M., Xu, K., & Schuurmans, D. (2017). Bridging the gap between value and policy based reinforcement learning. *Advances in neural information processing systems*, *30*.

Nakayama, H., Yun, Y., & Yoon, M. (2009). *Sequential approximate multiobjective optimization using computational intelligence*. Springer Science & Business Media.

Natarajan, S., & Tadepalli, P. (2005). Dynamic preferences in multi-criteria reinforcement learning. In *Proceedings of the 22nd international conference on Machine learning*, (pp. 601–608).

Nelder, J. A., & Mead, R. (1965). A simplex method for function minimization. *The computer journal*, *7*(4), 308–313.

Ning, C., & You, F. (2017). Data-driven adaptive nested robust optimization: general modeling framework and efficient computational algorithm for decision making under uncertainty. *AIChE Journal*, *63*(9), 3790–3817.

Nishihara, R., Murray, I., & Adams, R. P. (2014). Parallel mcmc with generalized elliptical slice sampling. *The Journal of Machine Learning Research*, *15*(1), 2087–2112.

Nowak, D. J., & Walton, J. T. (2005). Projected urban growth (2000–2050) and its estimated impact on the us forest resource. *Journal of Forestry*, *103*(8), 383–389.

Oh, J., & Bartos, M. (2023). Model predictive control of stormwater basins coupled with real-time data assimilation enhances flood and pollution control under uncertainty. *Water Research*, *235*, 119825.

Pareto, V. (1964). *Cours d'économie politique*, vol. 1. Librairie Droz.

Plappert, M., Houthooft, R., Dhariwal, P., Sidor, S., Chen, R. Y., Chen, X., Asfour, T., Abbeel, P., & Andrychowicz, M. (2017). Parameter space noise for exploration. *arXiv preprint arXiv:1706.01905*.

Plischke, E., Borgonovo, E., & Smith, C. L. (2013). Global sensitivity measures from given data. *European Journal of Operational Research*, *226*(3), 536–550.

Quinn, J. D., Reed, P. M., Giuliani, M., & Castelletti, A. (2017). Rival framings: A framework for discovering how problem formulation uncertainties shape risk management trade-offs in water resources systems. *Water Resources Research*, *53*(8), 7208–7233.

Radovanovic, M., Nanopoulos, A., & Ivanovic, M. (2010). Hubs in space: Popular nearest neighbors in high-dimensional data. *Journal of Machine Learning Research*, *11*(sept), 2487–2531.

Reed, P. M., Hadka, D., Herman, J. D., Kasprzyk, J. R., & Kollat, J. B. (2013). Evolutionary multiobjective optimization in water resources: The past, present, and future. *Advances in water resources*, *51*, 438–456.

Reich, J., Kinra, A., Kotzab, H., & Brusset, X. (2021). Strategic global supply chain network design–how decision analysis combining milp and ahp on a pareto front can improve decision-making. *International Journal of Production Research*, *59*(5), 1557–1572.

Ritter, C., & Tanner, M. A. (1992). Facilitating the gibbs sampler: the gibbs stopper and the griddy-gibbs sampler. *Journal of the American Statistical Association*, *87*(419), 861–868.

Robert, C. P., Casella, G., Robert, C. P., & Casella, G. (2004). The metropolishastings algorithm. *Monte Carlo statistical methods*, (pp. 267–320).

Roberts, G. O. (1992). Convergence diagnostics of the gibbs sampler. *Bayesian statistics*, *4*, 775–782.

Roberts, G. O., & Rosenthal, J. S. (2001). Optimal scaling for various metropolis-hastings algorithms. *Statistical science*, *16*(4), 351–367.

Roberts, G. O., & Rosenthal, J. S. (2009). Examples of adaptive mcmc. *Journal of computational and graphical statistics*, *18*(2), 349–367.

Roberts, G. O., & Tweedie, R. L. (1999). Bounds on regeneration times and convergence rates for markov chains. *Stochastic Processes and their applications*, *80*(2), 211–229.

Rosenstein, M. T., & Barto, A. G. (2001). Robot weightlifting by direct policy search. In *International joint conference on artificial intelligence*, vol. 17, (pp. 839–846). Citeseer.

Rosenthal, J. S. (1995). Minorization conditions and convergence rates for markov chain monte carlo. *Journal of the American Statistical Association*, *90*(430), 558–566.

Roy, V. (2020). Convergence diagnostics for markov chain monte carlo. *Annual Review of Statistics and Its Application*, *7*, 387–412.

Saliba, S., Bowes, B., Adams, S., Beling, P., & Goodall, J. (2020). Mitigation of flooding in stormwater systems utilizing imperfect forecasting and sensor data with deep deterministic policy gradient reinforcement learning.

Sauerteig, P., & Worthmann, K. (2020). Towards multiobjective optimization and control of smart grids. *Optimal Control Applications and Methods*, *41*(1), 128–145.

Shavazipour, B., Kwakkel, J. H., & Miettinen, K. (2021). Multi-scenario multi-objective robust optimization under deep uncertainty: A posteriori approach. *Environmental Modelling & Software*, *144*, 105134.

Shen, F., Zhao, L., Du, W., Zhong, W., & Qian, F. (2020). Large-scale industrial energy systems optimization under uncertainty: A data-driven robust optimization approach. *Applied Energy*, *259*, 114199.

Smith, J. D., Quinn, J. D., & Band, L. E. (2024). Comparing robust optimization approaches for addressing hydrologic model uncertainty in infrastructure planning: A green infrastructure example. *Authorea Preprints*.

Srikrishnan, V., Lafferty, D. C., Wong, T. E., Lamontagne, J. R., Quinn, J. D., Sharma, S., Molla, N. J., Herman, J. D., Sriver, R. L., Morris, J. F., et al. (2022). Uncertainty analysis in multi-sector systems: Considerations for risk analysis, projection, and planning for complex systems. *Earth's Future*, *10*(8), e2021EF002644.

Stedinger, J. R., Vogel, R. M., Lee, S. U., & Batchelder, R. (2008). Appraisal of the generalized likelihood uncertainty estimation (glue) method. *Water resources research*, *44*(12).

Storn, R., & Price, K. (1997). Differential evolution-a simple and efficient heuristic for global optimization over continuous spaces. *Journal of global optimization*, *11*(4), 341.

Tabas, S. S., & Samadi, V. (2024). Fill-and-spill: Deep reinforcement learning policy gradient methods for reservoir operation decision and control. *Journal of Water Resources Planning and Management*, *150*(7), 04024022.

Ter Braak, C. J., & Vrugt, J. A. (2008). Differential evolution markov chain with snooker updater and fewer chains. *Statistics and Computing*, *18*, 435–446.

Uhlenbeck, G. E., & Ornstein, L. S. (1930). On the theory of the brownian motion. *Physical review*, *36*(5), 823.

Vallender, S. (1974). Calculation of the wasserstein distance between probability distributions on the line. *Theory of Probability & Its Applications*, *18*(4), 784–786.

Vamplew, P., Dazeley, R., Berry, A., Issabekov, R., & Dekker, E. (2011). Empirical evaluation methods for multiobjective reinforcement learning algorithms. *Machine learning*, *84*, 51–80.

Vamplew, P., Yearwood, J., Dazeley, R., & Berry, A. (2008). On the limitations of scalarisation for multi-objective reinforcement learning of pareto fronts. In *AI 2008: Advances in Artificial Intelligence: 21st Australasian Joint Conference on Artificial Intelligence Auckland, New Zealand, December 1-5, 2008. Proceedings 21*, (pp. 372–378). Springer.

Vats, D., Flegal, J. M., & Jones, G. L. (2019). Multivariate output analysis for markov chain monte carlo. *Biometrika*, *106*(2), 321–337.

Vinyals, O., Ewalds, T., Bartunov, S., Georgiev, P., Vezhnevets, A. S., Yeo, M., Makhzani, A., Küttler, H., Agapiou, J., Schrittwieser, J., et al. (2017). Starcraft ii: A new challenge for reinforcement learning. *arXiv preprint arXiv:1708.04782*.

Vrugt, J. A., Ter Braak, C., Diks, C., Robinson, B. A., Hyman, J. M., & Higdon, D. (2009). Accelerating markov chain monte carlo simulation by differential evolution with self-adaptive randomized subspace sampling. *International journal of nonlinear sciences and numerical simulation*, *10*(3), 273–290.

Wald, A. (1949). Statistical decision functions. *The Annals of Mathematical Statistics*, (pp. 165–205).

Wang, D., Su, J., & Yu, H. (2020). Feature extraction and analysis of natural language processing for deep learning english language. *IEEE Access*, *8*, 46335–46345.

Wasko, C., Westra, S., Nathan, R., Orr, H. G., Villarini, G., Villalobos Herrera, R., & Fowler, H. J. (2021). Incorporating climate change in flood estimation guidance. *Philosophical Transactions of the Royal Society A*, *379*(2195), 20190548.

Watkins, C. J., & Dayan, P. (1992). Q-learning. *Machine learning*, *8*, 279–292.

Xie, Y., Feng, K., Du, M., Wang, Y., & Li, L. (2023). Robust optimization of stamping process based on bayesian estimation. *Journal of Manufacturing Processes*, *101*, 245–258.

Xu, H., Ma, C., Xu, K., Lian, J., & Long, Y. (2020). Staged optimization of urban drainage systems considering climate change and hydrological model uncertainty. *Journal of hydrology*, *587*, 124959.

Yan, Y., Du, H., He, D., & Li, W. (2022). Pareto optimal information flow topology for control of connected autonomous vehicles. *IEEE Transactions on Intelligent Vehicles*, *8*(1), 330–343.

Yan, Z., & Xu, Y. (2020). A multi-agent deep reinforcement learning method for cooperative load frequency control of a multi-area power system. *IEEE Transactions on Power Systems*, *35*(6), 4599–4608.

Yu, B., et al. (1995). Estimating l1 error of kernel estimator: Monitoring convergence of markov samplers. *cahier de recherche, Technical report, Dept. of Statistics, University of California, Berkeley*.

Yu, Y., Chen, L., Xiao, Y., Chang, C.-C., Zhi, X., & Shen, Z. (2022). New framework for assessing urban stormwater management measures in the context of climate change. *Science of the Total Environment*, *813*, 151901.

Zaniolo, M., Giuliani, M., & Castelletti, A. (2021). Neuro-evolutionary direct policy search for multiobjective optimal control. *IEEE Transactions on Neural Networks and Learning Systems*, *33*(10), 5926–5938.

Zatarain Salazar, J., Reed, P. M., Herman, J. D., Giuliani, M., & Castelletti, A. (2016). A diagnostic assessment of evolutionary algorithms for multi-objective surface water reservoir control. *Advances in water resources*, *92*, 172–185.

Zatarain Salazar, J., Reed, P. M., Quinn, J. D., Giuliani, M., & Castelletti, A. (2017). Balancing exploration, uncertainty and computational demands in many objective reservoir optimization. *Advances in water resources*, *109*, 196–210.

Zhang, H., & Yu, T. (2020). Taxonomy of reinforcement learning algorithms. *Deep reinforcement learning: Fundamentals, research and applications*, (pp. 125–133).