Scaling Approximate Linear Programming: Langevin Based Sampling for Constraint Violation Learning

BY

VIRGINIA MARCANTE
B.S., Politecnico di Torino, Turin, Italy, 2021

THESIS

Submitted as partial fulfillment of the requirements
for the degree of Master of Science in Computer Science
in the Graduate College of the
University of Illinois at Chicago, 2023

Chicago, Illinois

Defense Committee:

Ian Kash
Selvaprabu Nadarajah
Xinhua Zhang
Sandra Pieraccini, Politecnico di Torino

ACKNOWLEDGMENTS

First and foremost, I extend my deepest gratitude to Dr. Selvaprabu Nadarajah, for his unwavering support, guidance, and mentorship throughout the course of this research. His expertise and dedication have been pivotal in shaping this thesis.

I would also like to express my appreciation to Dr. Ian Kash and Dr. Sandra Pieraccini, my advisors at UIC and PoliTO, respectively. Their insights and perspectives added value to my academic journey.

On a personal note, I extend my deepest gratitude to my family for their unwavering encouragement and belief in me. My friends have been steadfast pillars of support, and I am immensely thankful for them.

To everyone mentioned and to those who have silently contributed in the background, thank you. This thesis is a result of your trust, guidance, and the collaborative spirit we've fostered. It stands as a testament to our shared dedication and effort.

VM

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

UIC	University of Illinois at Chicago
ALP	Approximate Linear Program
ULA	Unadjusted Langevin Algorithm
MDP	Markov Decision Process
SDP	Stochastic Dynamic Program
MH	Metropolis Hastings
EBM	Energy Based Model
VFA	Value Function Approximation
KL	Kullback Leibler
PSMD	Proximal Stochastic Mirror Descent
PIC	Perishable Inventory Control
UB	Upper Bound
LB	Lower Bound

SUMMARY

Approximate Linear Programs (ALPs) are fundamental models for computing value function approximations and bounds for high-dimensional Markov decision processes (MDPs) arising in a wide range of applications. ALP has a manageable number of variables and a large number of constraints. Constraint generation and sampling are two traditional approaches to handle the numerous constraints. The former approach has limited applicability. The latter approach, while broadly applicable, does not guarantee a valid bound on the optimal policy value. Constraint violation learning is a recent approach for solving ALP that combines first-order methods with Metropolis Hastings sampling to provide a general purpose approach that retains the bounding property of ALP and has convergence guarantees. Its use of Metropolis Hastings however limits its scalability. This thesis introduces a novel adaptation of constraint violation learning based on Langevin Dynamics (LD) that allows us to scale the solution of ALPs to higher dimensional applications.

The primary contribution of the thesis is a comprehensive examination of LD for constraint violation learning. This is a challenging setting for LD because sampling distributions are not log-concave in general, which is when LD has well understood theoretical properties. Nevertheless, we find that a log-concave regularization term used within constraint violation learning plays an important role in ensuring the effectiveness of LD. We argue that the core computational cost of LD is dimension independent. Through a comprehensive numerical study on a perishable inventory control application, this thesis demonstrates that LD maintains consistent

running times even as MDP dimension increase, making it a scalable option for solving ALP. In contrast, Metropolis Hastings does not scale as effectively, as expected.

This research extends the scalability of ALP using an LD based sampling approach in constraint violation learning. The exploration of LD for solving ALP is novel, thus bringing together tools from machine learning and operations research. We are also not aware of the use of LD to address the large scale nature of mathematical programs (e.g., linear programs). This thesis takes a first step at exploring this interface and its findings motivate the exploration of LD for large-scale optimization beyond ALP.

CHAPTER 1

INTRODUCTION

In operations research, Approximate Linear Programming is a pivotal technique to approximate high dimensional Markov decision processes (MDPs). Its significance is underscored by its ability to provide solutions to MDPs that are otherwise computationally intractable to solve. However, despite its importance, ALP is often criticized for its lack of scalability, especially when dealing with high-dimensional problems. In recent developments, [1] introduced constraint violation learning as a potential solution to overcome the limitations of traditional approaches such as constraint generation and constraint sampling for solving ALP. Yet, this approach is not without its own limitations. Specifically, it heavily depends on the computation of high-dimensional expectations, which [1] tackle using Metropolis Hastings, a sampling technique that does not scale well with dimension.

Venturing into the realm of machine learning, this thesis introduces a novel approach to address this limitation: the use of Langevin dynamics for computing high dimensional expectations in constraint violation learning. Langevin dynamics, having shown immense promise in machine learning for sampling from high-dimensional distributions, offers a novel avenue to enhance the scalability and efficiency of constraint violation learning in ALP. However, its application in an ALP setting is non trivial because its theoretical properties have been well documented in the case of log-concave distributions, a condition not always met.

Langevin dynamics, with its origins in statistical mechanics, has been a cornerstone in the study of molecular motion. Its ability to traverse complex energy landscapes through the inclusion of stochastic components, makes it an ideal tool for the challenges posed by ALP. By leveraging Langevin dynamics, this research aspires to elevate sampling efficiency, ensuring that the derived solutions are not only optimal but also computationally viable.

As previously mentioned, the distributions from which we sample are not necessarily log-concave. However, in our research we found that by changing the value of the regularization term in constraint violation learning, we can influence the log-concavity of the distribution. Through fine-tuning of the regularization parameter we are able to effectively obtain samples. It is worth noting that this is one of the first works which introduces the use of Langevin dynamics into linear programming. Furthermore, to the best of our knowledge, no other work as studied a distribution whose log-concavity can be modulated by a parameter.

In order to ensure that the samples we obtain are within the state-action domain, we have added a projection step following the Langevin update. This step acts as a safeguard, ensuring that our samples remain consistent with the constraints of our problem space.

The proposed methodology's ability to handle large instances effectively showcases its potential for real-world applications where dimensions can be vast and varied. The crux of this research lies in the development and meticulous evaluation of an algorithm that seamlessly melds Langevin dynamics into the ALP framework. While the theoretical foundation is paramount, the real merit of any method is gauged by its practical application. In this context, the Perishable Inventory Control (PIC) problem is utilized as a testing ground. PIC serves as a rigorous

platform to assess the efficacy, scalability, and adaptability of the Langevin dynamics-infused approach.

The ramifications of this study are significant. The prospects of diminished computational demands and therefore superior scalability have the potential to extend the applicability of ALP. More broadly, melding Langevin dynamics with ALP paves the way for tackling a other large scale optimization challenges across varied sectors. It also presents another area where synergies between machine learning and operations research (specifically optimization) appear to have significant value.

The structure of this thesis is as follows: Chapter 2 provides a literature review, discussing existing methods and their limitations. Chapter 3 covers the background material, offering insights into the foundational concepts that underpin the subsequent chapters. In Chapter 4, the focus shifts to the introduction of constraint violation learning and the detailing of our innovative approach that harnesses Langevin dynamics. Chapter 5 presents the results derived from the proposed methodology, offering both a detailed analysis and a comparative perspective with existing methods. Chapter 6 provides concluding remarks and points towards potential avenues for future exploration in this domain.

CHAPTER 2

RELATED LITERATURE

Sampling techniques have consistently been a focal point of research, evolving with the introduction of numerous methods tailored to address its inherent challenges. In recent times, Langevin dynamics, a technique with origins in physics, has carved a niche for itself within the machine learning community [2] [3] [4] [5] [6]. The technique's adeptness in handling high-dimensional domains hints at its potential versatility, extending beyond its current applications.

The integration of Langevin dynamics into optimization is a more recent endeavor, with limited literature bridging the two [7] [8]. This is somewhat unexpected, considering the pivotal role sampling has always played in optimization, particularly when dealing with high-dimensional distributions. While traditional methods like Metropolis-Hastings have their merits, they often grapple with challenges in high-dimensional spaces. In contrast, Langevin dynamics offers an alternative approach, bypassing some of these inherent difficulties. While there is some use of Langevin dynamics in optimization, to the best of our knowledge, it has not been used to solve large scale math programs.

Our study embarks on this innovative journey, applying Langevin dynamics within the context of constraint violation learning for solving Approximate Linear Programs. Historically, ALPs have been addressed through two primary strategies: row generation [9] [10] [11] and constraint sampling [12] [13] [14] [15]. The former involves solving a relaxed version of the ALP with a limited subset of constraints, while the latter pre-samples state-action pairs by

simulating a heuristic control policy. Constraint violation [1] learning, however, differs from these methods. It identifies and samples constraints that are highly violated and uses them to solve the ALP.

To the best of our knowledge, this is the first study where the log-concavity of the distribution being sampled can be controlled using a parameter. This aspect remains largely uncharted, yet it holds significant importance. Within the domain of Langevin dynamics, the log-concavity directly influences both the convergence rate and the precision of the samples.

In summary, this work delves into the intricate interplay between Langevin dynamics and large scale optimization, particularly within the context of constraint violation learning for Approximate Linear Programs. By introducing a parameterizable log-concavity to the sampled distribution, we open a new dimension of exploration that can significantly impact the convergence and accuracy of Langevin dynamics.

CHAPTER 3

BACKGROUND MATERIAL

3.1 Approximate Linear Programming

3.1.1 Introduction to Markov Decision Processes

A Markov Decision Process (MDP) is a mathematical model that captures the essence of decision-making in stochastic environments. It is a fundamental concept in operations research, artificial intelligence, and many other fields where decisions are made over time under uncertainty.

An MDP is defined by the following components:

- $\mathcal{S}$: denotes the state space.
- $\mathcal{A}$: the action space, which is independent of the state.
- $p(s'|s,a)$: the transition probability density function which represents the probability of moving from state s to state s' under action a.
- $c(s,a)$: denotes the cost of taking action a in state s. This cost is a continuous function spanning both the state and action domains.
- γ: discount factor, which has a value between 0 and 1.
- $\pi : \mathcal{S} \to \mathcal{A}$: maps each state in set $\mathcal{S}$ to an action in set $\mathcal{A}$.

Both $\mathcal{S}$ and $\mathcal{A}$ are assumed to be continuous, convex, and compact domains. Additionally we operate under the assumption that these spaces are full-dimensional to sidestep potential

problems. Given these elements, we can define the expected discounted cost under policy π over an infinite planning horizon as:

$$V^{\pi}(s) = \mathbb{E}_s^{\pi}\left[\sum_{t=0}^{\infty}\gamma^t c(s_t, \pi(s_t))\right] \tag{3.1}$$

Where s_t represents the state at time t when applying policy π from an initial state $s_0 = s$. The expectation is computed with respect to π and the corresponding transition probability distribution $p(\cdot|s,a)$. The objective is to determine a policy π by minimizing $V^{\pi}(s)$ over the set of all feasible policies Π.

$$V^*(s) = \inf_{\pi\in\Pi} V^{\pi}(s), \quad \forall s \in \mathcal{S}. \tag{3.2}$$

Adopting Assumption 1 from paper [1], we substitute the infimum with a minimum. The optimal policy, π^*, solves the Stochastic Dynamic Program (SDP) given by

$$V^*(s) = \min_{a\in\mathcal{A}}\left[c(s,a) + \gamma\mathbb{E}_p[V^*(s')|s,a]\right], \quad \forall s \in \mathcal{S}. \tag{3.3}$$

Here, $\mathbb{E}_p[V^*(s')|s,a]$ represents the expectation determined by $p(\cdot|s,a)$.

Theoretically, this SDP can be formulated as an infinite linear program, as detailed in [16]:

$$\begin{aligned} &\max_{V}\mathbb{E}_q[V(s)] \\ &s.t.\ V(s) - \gamma\mathbb{E}_p[V(s')|s,a] \le c(s,a), \quad \forall (s,a) \in \mathcal{S}\times\mathcal{A}. \end{aligned} \tag{3.4}$$

In this formulation, $q(\cdot)$ is a continuous probability density function over $\mathcal{S}$. The goal of this linear program is to optimize the expectation of $V(\cdot)$ with respect to $q(\cdot)$. By substituting the action minimization in the original SDP with a series of inequalities, we obtain the constraints.

3.1.2 Approximate Linear Programming for MDPs

Solving the problem as presented in Equation 3.4 is computationally challenging as it is a doubly infinite linear program. This program encompasses a continuous range of decision variables and constraints, each corresponding respectively to a specific state and state-action pair. To address this continuum of variables, we approximate the function $V(\cdot)$ using a set of stochastic basis functions. For a state space $\mathcal{S}$ with dimension d, a basis function is characterized by a coefficient vector $\omega = (\omega_0, \omega_1, \ldots, \omega_d) \in \mathbb{R}^{d+1}$, drawn from a density function $\rho(\omega)$. A powerful class of basis functions that can approximate continuous functions well are referred to as random basis function $\phi(\cdot)$ [17] over the state space given the coefficient ω maps a state s to $\phi(s|\omega) = \phi\left(\omega_0 + \sum_{i=1}^{d} \omega_i s_i\right)$.

Given a collection of B basis functions, associated weights $\theta = (\theta_1, \ldots, \theta_B) \in \mathbb{R}^B$, and a constant $\tau \in \mathbb{R}$, we can represent $V(\cdot)$ as $V(s) \approx \tau + \sum_{b=1}^{B} \theta_b \phi_b(s)$. Utilizing this representation, we derive the following Approximate Linear Programming formulation, denoted as F:

$$\begin{aligned} \max_{(\tau,\theta)\in\mathbb{R}^{B+1}} \quad & \tau + \sum_{b=1}^{B} \theta_b \mathbb{E}_q[\phi_b(s)] \\ \text{s.t.} \quad & (1-\gamma)\tau + \sum_{b=1}^{B} \theta_b(\phi_b(s) - \gamma \mathbb{E}_p[\phi_b(s')|s,a]) \leq c(s,a), \quad \forall (s,a) \in \mathcal{S} \times \mathcal{A}. \end{aligned} \tag{3.5}$$

Given a value function approximation, an associated action can be computed by leveraging the right hand side of Equation 3.3 as

$$\min_{a \in \mathcal{A}} \left\{ c(s,a) + \gamma \sum_{b=1}^{B} \theta_b \mathbb{E}_p[\phi_b(s')|s,a] \right\}. \tag{3.6}$$

Directly solving ALP poses difficulties due to its infinite constraints and the potential complexity in calculating expectations. Row generation [9] [10] [11] and constraint sampling [12] [13] [14] [15] are common solution strategies that address one or both of these challenges.

3.2 Drawing Samples: Metropolis Hastings to Langevin Dynamics

Throughout the 20th century, the task of generating samples from a probability distribution emerged as a pivotal challenge in applied mathematics and computer science. This endeavor has been instrumental in advancing fields ranging from statistical simulations to machine learning. The quest for algorithms that can effectively sample, particularly when confronted with an undefined or partially known probability density function, has been driven by both the theoretical intricacies and the vast array of practical applications it supports. As we delve deeper into complex data-driven domains, the significance of robust sampling techniques continues to grow. In this section, we will explore two renowned methods specifically designed to sample from a probability density function that is known up to a constant term.

3.2.1 Metropolis Hastings

The Metropolis-Hastings (MH) algorithm [18] stands as a cornerstone in the realm of methods designed to generate samples from a probability distribution, especially when direct sampling proves challenging. The primary objective of MH is to sample from a target distribution $p(x)$. In many scenarios, while $p(x)$ remains elusive, we possess knowledge of it up to a normalization constant, meaning we are aware of some function $f(x)$ such that $p(x) \propto f(x)$.

To aid the sampling process, the algorithm introduces a proposal distribution $\nu(x'|x)$, which suggests a potential new state x' based on the current state x. This distribution is typically chosen for its ease of sampling. The algorithm begins with an initial state x_0. From this current

state x, a proposed state x' is drawn from $\nu(x'|x)$. The newly generated state x' is accepted with probability α computed [18] as:

$$\alpha(x, x') = \min\left(1, \frac{f(x')\nu(x|x')}{f(x)\nu(x'|x)}\right)$$

This iterative process is continued for a predetermined number of steps n. The sequence of states x produced by the algorithm forms a Markov chain. After a certain number of iterations, known as the "burn-in" period, this chain is believed to have converged to the target distribution $p(x)$ and can be employed for subsequent analyses. The procedure, which is described in Algorithm 1, is versatile and allows for sampling from multi-dimensional distributions. The efficiency of the MH diminishes as the dimensionality of the target distribution rises [19]. In high-dimensional spaces, the algorithm's acceptance rate drops, leading to correlated successive samples and a "random walk" behavior. This causes the Markov chain to take longer to explore the state space, resulting in slow convergence and inefficient mixing.

Algorithm 1 Metropolis-Hastings

Initial value x_0
for $k = 1$ **to** K **do**
  Sample independently $x' \sim \sqrt{}(\cdot|x_k)$ and $u \sim U(0, 1)$
  if $u < \alpha(x_k, x')$ **then**
    $x_{k+1} \leftarrow x'$
  else
    $x_{k+1} \leftarrow x_k$
  end if
end for

3.2.2 Langevin Dynamics

The Langevin dynamics is an approach used for sampling from a target probability density distribution. The approach originates from a discretized form of a specific stochastic differential equation, commonly referred to as Langevin dynamics. The equation, first proposed by Paul Langevin [20], and later rigorously formulated by J. L. Doob in [21] [22], described the Brownian motion of a particle with friction. The diffusion process in Langevin dynamics is mathematically represented as

$$dL_{\mathrm{LD},t} = -\nabla f(L_{\mathrm{LD},t})\, dt + \sqrt{2}\, dW_t,$$

where W_t is a standard Brownian motion. The initial proposal to employ Langevin dynamics as a method for continuous-time simulation was made by [23] in the realm of pattern theory. Before delving into the details of the algorithm, it is essential to establish the notation.

Consider a probability distribution that is known up to a constant term. The distribution can be expressed as:

$$p_\theta(x) = \frac{e^{-E(x)}}{\int_{x\in X} e^{-E(x)}\, dx} = \frac{e^{-E(x)}}{Z(\theta)} \tag{3.7}$$

Where $Z(\theta)$ serves as the normalization constant term. In this expression, $E(x)$ is referred to as the Energy.

The equation referenced in Equation 3.7 establishes the framework for an Energy-Based Model (EBM). The core principle behind the Langevin dynamics method is to utilize the

Langevin equation for sampling from this EBM. By applying a time discretization to the Langevin equation, we arrive at the following approximation:

$$x_{k+1} = x_k - \frac{\varepsilon}{2}\nabla E(x_k) + \varepsilon\,\xi_k \quad \text{where} \quad \xi_k \sim \mathcal{N}(0,\,1) \tag{3.8}$$

Similarly to Metropolis-Hastings, it is sufficient to know $p_\theta(x)$ up to the normalization constant. This is because the latter disappears when calculating the gradient of the drift term. It is noteworthy that the updates in Equation 3.8 bear a resemblance to those in gradient descent. The key distinction lies in the introduction of a stochastic "noise" term ξ_k. We can interpret the Gaussian noise as a non-vanishing gradient noise which allows to get away from the local minima of the non-convex potentials [24] [25].

Thus, it is reasonable to view the samples as originating from a Gaussian distribution.

$$x_{k+1} \sim \mathcal{N}(x_k - \frac{\varepsilon}{2}\nabla E(x_k),\,\varepsilon)$$

The pseudocode of the Langevin Dynamics algorithm, often referred to as the Unadjusted Langevin Algorithm (ULA) or Langevin Monte Carlo in the literature [26] [27], is detailed in 2.

The main advantages of employing this technique for sampling include its ability to generate multiple samples at once and the lack of a need for an acceptance or rejection step, which makes the process quicker compared to the Metropolis-Hastings method. Furthermore it allows to sample efficiently from high-dimensional distributions. Traditional sampling methods can

Algorithm 2 Langevin Dynamics Sampling

Initial value x_0
for $k = 1$ **to** K **do**
  Sample independently $\xi_k \sim \mathcal{N}(0, I_p)$
  $x_{k+1} \leftarrow x_k - \frac{\varepsilon}{2}\nabla E(x_k) + \varepsilon\,\xi_k$
end for

struggle in high-dimensional spaces due to the curse of dimensionality, where the volume of the space grows exponentially with the number of dimensions. Langevin dynamics can navigate such spaces more effectively.

Welling and Teh [28] have examined the behavior of the ULA when employing diminishing step sizes. Their findings indicate that ULA weakly converges towards the intended distribution. Nevertheless, in the broader context of machine learning applications, fixed step sizes are commonly utilized instead of diminishing ones, as evidenced by multiple studies [29] [30] [31] [32] [33].

In cases where the target distribution is log-concave, which corresponds to scenarios where the energy function is both strongly convex and smooth, a convergence rate bound of $\mathcal{O}(d\epsilon^{-1})$ has been established [34]. A smooth function is defined as one with a Lipschitz-continuous gradient. Higher-order smoothness in the energy function can further enhance the rate of convergence [35]. Confirming the convergence properties of the Langevin Dynamics become significantly more intricate when the target distribution lacks log-concavity.

It is worth noting that despite the well-established theoretical guarantees for convergence when the target distribution is log-concave, many machine learning applications [32] [36] [37]

do not explicitly consider the log-concavity of the distribution they are sampling from. Without log-concavity samples might get trapped in a local minimum failing to explore others or take longer to move between regions of low energy, thus obtaining biased samples that do not represent the true distribution. Ignoring the log-concavity of the target distribution could compromise both the performance and reliability of machine learning models, making it vital for practitioners to understand the assumptions behind their chosen sampling algorithms.

Recent studies have determined the rate of convergence for sampling from non-log-concave distributions under specific conditions. These conditions include the fulfillment of functional inequalities like the log-Sobolev or Poincaré inequalities [38] [39] [40] [41], or they focus on manageable categories of non-log-concave sampling that meet particular tail-growth stipulations [42] [43] [44] [27]. One takeaway from these studies is clear: the rate of convergence deteriorates significantly when dealing with non-log-concave distributions.

CHAPTER 4

LANGEVIN DYNAMICS FOR CONSTRAINT VIOLATION LEARNING

In this chapter, the focus will be on the algorithmic approach we have adopted to address the ALP challenge and the novel contributions we have made to resolving this problem.

4.1 ALP Reformulation

We will begin by discussing how the authors in [1] derived a primal-dual reformulation of Approximate Linear Programs. Specifically, we replace the constraints in Equation 3.5 with a single constraint that serves as an upper bound for the value of τ. This new constraint is defined as follows:

$$\tau \leq \bar{\tau}(\theta) := \frac{1}{1-\gamma} \min_{(s,a)\in\mathcal{S}\times\mathcal{A}} \left\{ c(s,a) - \sum_{b=1}^{B} \theta_b \left(\phi_b(s) - \gamma \mathbb{E}_p \left[\phi_b(s') | s, a \right] \right) \right\} \tag{4.1}$$

The objective function of the ALP aims to maximize τ so the inequality in Equation 4.1 is expected to be an equality at the optimal solution. By replacing τ in the ALP objective from Equation 3.5 with $\bar{\tau}(\theta)$, we arrive at the following saddle-point formulation:

$$F = \max_{\theta\in\Theta} \min_{(s,a)\in\mathcal{S}\times\mathcal{A}} f(\theta, s, a) \tag{4.2}$$

where $f(\theta, s, a) := \frac{1}{1-\gamma} \left\{ c(s,a) - \sum_{b=1}^{B} \theta_b \left(\phi_b(s) - \gamma \mathbb{E}_p[\phi_b(s')|s,a] \right) \right\} + \sum_{b=1}^{B} \theta_b \mathbb{E}_q[\phi_b(s)]$ and Θ is a compact subset of $\mathbb{R}^B$ that contains the optimal ALP solution within its interior.

In Equation 4.2, the objective function is linear with respect to the value function approximation (VFA) weight vector θ, which is maximized. On the other hand, the minimization over the state-action space aims to identify the most violated constraint, a task that could potentially be nonconvex. However, according to Proposition 1 in [1], this nonconvex minimization can be substituted by an infinite-dimensional linear infimum over all continuous probability density functions defined over the state-action space. This set of functions is denoted by $\mathcal{Y}$, defined as $\mathcal{Y} := \left\{ y : \mathcal{S} \times \mathcal{A} \to \mathbb{R}_{++} \;\middle|\; \int_{\mathcal{S}\times\mathcal{A}} y(s,a)\, d(s,a) = 1 \right\}$.

Thus, the saddle-point formulation becomes:

$$F = \max_{\theta\in\Theta} \inf_{y\in\mathcal{Y}} \mathbb{E}_y[f(\theta, s, a)]. \tag{4.3}$$

To convert the infimum back to a minimum, a Kullback-Leibler (KL) regularization term $D(y, p_u)$ is introduced, defined as $D(y, p_u) = \mathbb{E}_y \left[\log \frac{y(s,a)}{\bar{p}}\right]$. Here, y is a continuous probability density function, and p_u is a uniform probability density function over $\mathcal{S} \times \mathcal{A}$ with a constant density value $\bar{p}$, calculated as $\bar{p} = \frac{1}{\left(\int_{\mathcal{S}\times\mathcal{A}} 1\, d(s,a)\right)}$.

The modified objective function then becomes:

$$F(\lambda) := \max_{\theta\in\Theta} \min_{y\in\mathcal{Y}} \left[\mathbb{E}_y[f(\theta, s, a)] + \lambda D(y, p_u)\right] \tag{4.4}$$

Since the term $\mathbb{E}_y[f(\theta, s, a)]$ is linear in both y and θ, and the KL divergence term is convex in y, the objective function in Equation 4.4 now represents a convex saddle-point problem. The inclusion of the Kullback-Leibler divergence term in the objective function serves to ensure

that the distribution y remains similar to p_u, the uniform distribution over the state-action space. Given that p_u is a uniform distribution, it is inherently log-concave. Consequently, the KL divergence term acts as a regularizing force that nudges y towards log-concavity. The parameter λ modulates the influence of this KL divergence term on the objective function. Higher values of λ amplify the importance of the KL divergence, thereby making y more log-concave. Conversely, lower values of λ reduce the emphasis on the KL divergence term, allowing other components of the objective function to have a more pronounced impact on y.

According to Proposition 2 in [1], for any $\theta \in \Theta$, there exists a dual minimizer $y^*_{\lambda,\theta} \in \mathcal{Y}$ given by

$$y^*_{\lambda,\theta}(s,a) := \frac{\exp\left(-\frac{f(\theta,s,a)}{\lambda}\right)}{\int_{\mathcal{S}\times\mathcal{A}} \exp\left(-\frac{f(\theta,s,a)}{\lambda}\right)\, d(s,a)} \tag{4.5}$$

Furthermore, for a specified $\alpha > 0$, a sufficiently small $\lambda \in (0,1]$ exists such that $F(\lambda) - F \leq \alpha$. This implies that the saddle-point problem in Equation 4.4 serves as a close approximation to that in Equation 4.2, and consequently to the original ALP, when λ is adequately small. This reformulation eliminates the need to solve finite-dimensional nonconvex optimization problems in Equation 4.2, reducing it to the infinite-dimensional convex optimization problem presented.

4.2 Algorithm

In this section, we outline the algorithm employed to achieve our results. Our approach is inspired by the Proximal Stochastic Mirror Descent (PSMD) Algorithm presented in [1]. The authors of this work devised a primal–dual stochastic mirror-descent technique to compute an α-optimal ALP solution.

Algorithm 3 PSMD

Input: Stopping tolerance TOL, regularization constant $\lambda_0 \in (0, 1]$, and step length $\eta_0 > 0$

1: Set $t = 0$, $\bar{\theta} = \theta_0 = (0, ..., 0) \in \mathbb{R}^B$, $\bar{y} = y_0 = p_u$, and $\bar{\lambda} = \lambda_0$

2: **while** STOP$(t, \bar{\theta}, \bar{y}, \bar{\lambda}$, TOL$)$ = FALSE **do**

3: Perform primal update: Sample state-action pair $(\hat{s}, \hat{a})$ from the density function y_t, generate next stage state $\check{s}$ from the MDP density function $p(\cdot|\hat{s}, \hat{a})$ and set

$$\theta_{t+1} = \arg\min_{\theta \in \Theta} \left[\eta_t \langle \theta_t - \theta, \nabla_\theta \hat{f}(\theta_t, \hat{s}, \hat{a}) \rangle + \frac{1}{2} \|\theta - \theta_t\|_2^2 \right], \tag{4.6}$$

where

$$\hat{f}(\theta_t, \hat{s}, \hat{a}) := \frac{1}{1-\gamma} \left\{ c(\hat{s}, \hat{a}) + \gamma \sum_{b=1}^{B} \theta_b \phi_b(\check{s}) - \sum_{b=1}^{B} \theta_b \phi_b(\hat{s}) \right\} + \sum_{b=1}^{B} \theta_b \mathbb{E}_q \left[\phi_b(s) \right]. \tag{4.7}$$

4: Perform dual update:

$$y_{t+1} \propto y_t^{(1+\eta_t \lambda_t)^{-1}} exp\left(-\eta_t \hat{f}(\theta_t, \cdot, \cdot)/(1 + \eta_t \lambda_t) \right). \tag{4.8}$$

5: Update step length and regularization coefficient:

$$\eta_{t+1} = \eta_0/\sqrt{t+2}; \quad \lambda_{t+1} = \lambda_0/\sqrt{t+2}. \tag{4.9}$$

6: Compute averaged regularization coefficient and primal and dual solutions

$$\bar{\lambda} = \frac{1}{\sum_{t'=0}^{t+1} \eta_t} \sum_{t'=0}^{t+1} \eta_{t'} \lambda_{t'}; \qquad \bar{\theta} = \frac{1}{\sum_{t'=0}^{t+1} \eta_t} \sum_{t'=0}^{t+1} \eta_{t'} \theta_{t'}; \qquad \bar{y} = \frac{1}{\sum_{t'=0}^{t+1} \eta_t} \sum_{t'=0}^{t+1} \eta_{t'} y_{t'} \tag{4.10}$$

7: Update itaration counter: $t = t + 1$
8: **end while**
Output: $\bar{\theta}$ **and** $\bar{y}$

The key distinction between our method and theirs lies in the computation of sample-average approximations, which are used to manage high-dimensional expectations. We will elaborate on this difference in the subsequent sections.

The Proximal Stochastic Mirror Descent algorithm is delineated in 3.

- Step 1: initializes the iteration counter and sets the algorithm's variables, commencing with a uniform distribution across the state-action space for y_0, $\bar{y}$.
- Step 2: initiates a loop that persists until a specified stopping criterion is met.
- Step 3: performs the primal update, where η_t signifies the step size at the t-th iteration. The update can be conceptualized as a gradient ascent operation succeeded by a projection onto the feasible set.
- Step 4: updates the dual variables. A closed-form expression for this update is derived from Equation 4.5. When sampling from y_{t+1}, the calculation of the normalization constant in the denominator can be circumvented if a sampling method compatible with

probability density functions known up to a constant is employed. Thus, only the denominator needs to be explicitly calculated for the update.

- Step 5: reduces both the step length and the regularization coefficient.
- Step 6: calculates moving averages for these coefficients, as well as for the primal and dual variables.

Upon completion, PSMD outputs a Value Function Approximation weight vector $\bar{\theta}$ and a state-action density function $\bar{y}$.

To determine the termination of Algorithm 3, a stopping criterion is essential. A straightforward approach is to set a maximum iteration count, T, chosen heuristically. Once this threshold is reached, the algorithm halts and outputs a solution. However, this method does not yield a lower bound on the optimal policy cost. In [1], a technique for approximating a lower bound was introduced, which we adopted both as a stopping criterion for Algorithm 3 and for estimating the optimality gap.

We begin by introducing two functions, primal and dual:

$$\mathcal{P}(y) = \max_{\theta \in \Theta} \mathbb{E}_y[f(\theta, s, a)] \tag{4.11}$$

$$\mathcal{D}(\theta) = \inf_{y \in \mathcal{Y}} \mathbb{E}_y[f(\theta, s, a)] \tag{4.12}$$

We know that $\mathcal{D}(\theta) \leq F \leq \mathcal{P}(y)$, implying that the difference, $\mathcal{P}(y) - \mathcal{D}(\theta)$, represents the duality gap of the saddle-point problem Equation 4.2.

The function $\mathcal{D}(\bar{\theta})$ offers a lower bound on F. However, its computation is intricate due to the presence of an infimum across the state-action space. To circumvent this, we introduce a regularized version:

$$\min_{y \in \mathcal{Y}} \mathbb{E}_y[f(\bar{\theta}, s, a)] + \bar{\lambda} D(y, p_u) \tag{4.13}$$

where $\bar{\lambda}$ denotes the averaged regularization coefficient determined by PSMD. Mirroring the approach in equation Equation 4.4, we can bypass the direct optimization over the state-action space, given that equation Equation 4.13 inherently has a definitive optimal solution.

The optimal solution of Equation 4.13 is represented by $y^*_{\bar{\lambda},\bar{\theta}}(s, a)$. Consequently, it can be reformulated as:

$$\mathbb{E}_{y^*_{\bar{\lambda},\bar{\theta}}}[f(\bar{\theta}, s, a)] + \bar{\lambda} D(y^*_{\bar{\lambda},\bar{\theta}}, p_u) \tag{4.14}$$

While this expression serves as an upper bound for $\mathcal{D}(\bar{\theta})$, Lemma 1 from [1] enables us to transform it into a lower bound. Specifically, we obtain:

$$\mathcal{D}(\bar{\theta}) \geq \mathbb{E}_{y^*_{\bar{\lambda},\bar{\theta}}}[f(\bar{\theta}, s, a)] + \bar{\lambda} D(y^*_{\bar{\lambda},\bar{\theta}}, p_u) + \bar{\lambda}\bar{C} + n\bar{\lambda}\log(\bar{\lambda}) \tag{4.15}$$

where $\bar{C}$ is defined as:

$$\bar{C} := \log(\bar{p}) - L(R + Q_{\mathcal{S}\times\mathcal{A}}) + n\log(R) - \log\left(\frac{\Gamma\left(\frac{n}{2}+1\right)}{\pi^{\frac{n}{2}}}\right) \tag{4.16}$$

In this expression, R represents the radius of the largest ball contained within $\mathcal{S} \times \mathcal{A}$; $Q_{\mathcal{S}\times\mathcal{A}}$ denotes the diameter of the set $\mathcal{S}\times\mathcal{A}$; $\Gamma(z)$ is the standard gamma function given by $\int_0^\infty x^{z-1}e^{-x}dx$ for $z > 0$; and L is the Lipschitz constant associated with $f(.,.,.)$[5].

Theorem 2 from [1] allows us to simplify the expression in Equation 4.15 by omitting the nonnegative term, yet retaining a valid lower bound:

$$l(\bar{\theta}) := \mathbb{E}_{y^*_{\bar{\lambda},\bar{\theta}}}[f(\bar{\theta}, s, a)] + \bar{\lambda}\bar{C} + n\bar{\lambda}\log(\bar{\lambda}) \tag{4.17}$$

Given that $\mathcal{D}(\bar{\theta})$ is an underestimate of F, which itself provides a lower bound on the optimal policy cost, it follows that $l(\bar{\theta})$ also establishes a lower bound on the policy cost.

With this lower bound in hand, we introduce an additional stopping criterion for our algorithm. We first simulate the ALP policy, which concludes the PSMD associated with the current $\bar{\theta}$, to derive an upper bound. Subsequently, we compute the lower bound $l(\bar{\theta})$ and halt the algorithm when the inequality $\mathcal{P}(\bar{y}) - l(\bar{\theta}) \leq \alpha$ is satisfied for a predetermined $\alpha > 0$. Moreover, the derived upper and lower bounds facilitate the computation of an optimality gap.

4.3 Implementation

In this section, we delve into the intricacies of the PSMD implementation.

The primary inputs for PSMD encompass a stopping tolerance TOL, an initial regularization coefficient λ_0, and an initial step length η_0. Typically, the choice of TOL is influenced by the acceptable time allocation of the user. The tuning of these parameters will be elaborated upon

in the results section. Once the optimal parameters are identified, they are employed to run PSMD to completion.

In the third step of Algorithm 3, in order to handle the potentially high-dimensional expectations in the term $\mathbb{E}_{y_t}\nabla_\theta f(\theta_t, s, a)$ we approximate the expectation with respect to y_t, using an unbiased point estimate $\nabla_\theta f(\theta_t, \hat{s}, \hat{a})$, using the sample pair $(\hat{s}, \hat{a})$ drawn from the density function y_t. Additionally, the function $f(\theta_t, \hat{s}, \hat{a})$ is substituted with an unbiased point estimate $\hat{f}(\theta_t, \hat{s}, \hat{a})$. This is done by replacing the expectation $\mathbb{E}_p[\phi_b(s)|\hat{s}, \hat{a}]$ in $f(\theta_t, \hat{s}, \hat{a})$, for the definition of $f(\theta_t, \hat{s}, \hat{a})$ see Equation 4.7, with a sample average approximation $\phi_b(\check{s})$ based on a single state $\check{s}$ drawn from the MDP transition density function $p(\cdot|\hat{s}, \hat{a})$.

While these unbiased approximations are theoretically sound, they might exhibit high variance in practice, potentially affecting the algorithm's empirical performance. To mitigate this, low-variance versions of these sample average approximations can be crafted using multiple samples. For instance, the term $\hat{f}(\theta, \hat{s}, \hat{a})$ can be replaced with a version based on multiple samples, enhancing the stability of the primal update.

In essence, by generating multiple samples and leveraging them, we can derive a low-variance stochastic gradient, enhancing the robustness of the algorithm.

In our implementation, we replace the term $\hat{f}(\theta, \hat{s}, \hat{a})$ with $\hat{f}_N(\theta, \hat{s}, \hat{a})$ which relies on N samples denoted as s_n where $n = 1, \ldots, N$, drawn from $p(\cdot|\hat{s}, \hat{a})$

$$\hat{f}^N(\theta, \hat{s}, \hat{a}) := \frac{1}{1-\gamma}\left\{c(\hat{s}, \hat{a}) + \gamma \sum_{b=1}^{B} \theta_b \left(\frac{1}{N}\sum_{n=1}^{N} \phi_b(s_n)\right) - \sum_{b=1}^{B} \theta_b \phi_b(\hat{s})\right\} + \sum_{b=1}^{B} \theta_b \mathbb{E}_q[\phi_b(s)].$$

We can obtain a low-variance stochastic gradient $\nabla_\theta \hat{f}(\theta, s, a)$, by using $\hat{f}^N(\theta_t, s, a)$ and generating a set of H samples $\{(\hat{s}_h, \hat{a}_h)\}_{h=1}^{H}$ from y_t. This results in a more stable gradient, given by $\frac{1}{H}\sum_{h=1}^{H} \nabla \hat{f}(\theta_t, \hat{s}_h, \hat{a}_h)$, which can be employed in the primal update Equation 4.6 as a substitute for $\nabla \hat{f}(\theta_t, \hat{s}, \hat{a})$.

This gradient is constructed based on samples from y_t and $p(\cdot|\hat{s}, \hat{a})$ which are respectively the state-action distribution and the MDP state transition distribution. Sampling becomes straightforward when the density function of the distribution is well-defined.

Given that the distribution y_t is an Energy Based Model, Langevin dynamics emerges as a natural choice for sampling.

The main advantage of Langevin dynamics is that its inherent structure does not directly depend on the problem's dimensionality. This means that its execution time might remain relatively stable, irrespective of the dimensionality of the distribution from which we are sampling. One reason for this is that it continuously updates samples without needing an accept-reject step, which often becomes a bottleneck in high-dimensional spaces. On the other hand, the Metropolis-Hastings algorithm struggles as dimensionality increases. With MH, as the data grows, it gets harder to find and accept new samples. This can result in the so-called "random

walk" behavior, where the Markov chain takes a long time to explore the entire state space. As the dimensionality grows, the space that needs to be explored expands exponentially, leading to an expected exponential increase in the MH algorithm's runtime.

While we base our approach to sample from y_t on the algorithm outlined in Section 3.2.2, we introduce a variation to better suit our requirements. Considering the continuous, convex, and compact nature of the State-Action set, it is essential to incorporate a projection step onto the feasible set, ensuring that the samples are in $\mathcal{S} \times \mathcal{A}$.

The algorithm, as described in 2, is modified as follows:

Algorithm 4 Projected Langevin Dynamics

Initial value x_0

for $k = 1$ **to** K **do**

Sample independently $\xi_k \sim \mathcal{N}(0, I_p)$

$x_{k+1} \leftarrow \Pi_{\mathcal{S}\times\mathcal{A}} \left(x_k - \frac{\varepsilon}{2} \nabla E(x_k) + \varepsilon\, \xi_k \right)$

end for

Here, $\Pi_{\mathcal{S}\times\mathcal{A}}(x)$ denotes the projection of x onto the set $\mathcal{S} \times \mathcal{A}$.

For the optimization in Step 3, we can employ a standard convex optimization solver. In our study, we opted for Gurobi [45].

We now delve into the computation of the lower bound. Evaluating this bound entails handling high-dimensional expectations.

The sample average approximation of $l(\bar{\theta})$ is given by:

$$\hat{l}(\bar{\theta}) := \mathbb{E}_{(H',\,N')}\left[\frac{1}{H'}\sum_{h=1}^{H'}\hat{f}^{N'}(\bar{\theta},\hat{s}_h,\hat{a}_h)\right] + \bar{\lambda}\bar{C} + n\bar{\lambda}\log(\bar{\lambda})$$

The N' samples are drawn from $p(\cdot|s,a)$ for every one of the H' samples, which are generated from the density:

$$\hat{y}_{\bar{\lambda},\bar{\theta}}(s,a) \propto \exp\left(-\frac{\hat{f}^{N'}(\bar{\theta},s,a)}{\bar{\lambda}}\right)$$

As with Step 3, we employ Langevin dynamics to obtain state-action samples from $\hat{y}_{\bar{\lambda},\bar{\theta}}(s,a)$.

To compute $\hat{l}(\bar{\theta})$, the constant $\bar{C}$ must be defined. While parameters $\bar{p}$, R, and $Q_{\mathcal{S}\times\mathcal{A}}$ in its definition can be straightforwardly computed given our state-action space's simple structure, in more complex scenarios, $\bar{C}$ can be substituted with a lower bound $\bar{C}_{LB}$, as detailed in [1].

CHAPTER 5

PERISHABLE INVENTORY CONTROL

In this chapter, we discuss the results obtained from applying our algorithm to various Perishable Inventory Control instances. These instances are particularly important for understanding the scalability and applicability of our approach. Initially, we focus on a two-dimensional PIC instance with partially backlogged demand and zero order lead time to provide insights into the algorithm's behavior. Subsequently, we extend our experiments to more complex setting, specifically Perishable Inventory Control with Partial Backlogging and Lead Time instances with three, five, and ten-dimensional state spaces. The results are evaluated based on specific performance metrics and are compared with existing benchmarks to demonstrate the effectiveness of our algorithm.

5.1 Perishable Inventory Control partially backlogged demand and zero order lead time

To evaluate the effectiveness of our approach, we initially chose to apply it to a single-product inventory-control system characterized by partially backlogged demand and zero order lead time. This selection was motivated by prior studies conducted by Nahmias and Smith [46], Rabinowitz et al. [47], and Benjaafar et al. [48], which served as a foundation for our research.

The decision to focus on this particular inventory-control system was driven by several factors. Firstly, the system exhibits a two-dimensional state space, allowing for a comprehensive

analysis of the state-action dynamics. This two-dimensional nature of the system enables us to graphically visualize and interpret the results, facilitating a more intuitive understanding of the observed behaviors.

Furthermore, we can compare the results obtained using Langevin dynamics with those obtained using Metropolis Hastings sampling for the same application previously studied in [1] in order to assess the effectiveness of our approach. This allows us to evaluate the performance and relative advantages of these two sampling techniques within the context of the given application.

5.1.1 MDP Formulation and Instance

In our MDP formulation, the state s is bounded by an upper bound $u_s > 0$ and a lower bound $l_s < 0$, which are respectively 10 and -10. The state represents the on-hand inventory, where negative values indicate backlogged orders.

The action a corresponds to the order quantity, which is constrained within the range $[0, \bar{a}]$, where $\bar{a}$ represents the maximum order size. We assume that orders are placed prior to the realization of stochastic demand G, which follows the distribution P_G, modeled as a truncated normal distribution on the interval $[0, 10]$ with mean 5 and a standard deviation 2.

To determine the next inventory level s', we consider the current inventory s, the order quantity a, and the demand G. The transition function ensures that s' lies within the specified bounds: $s' = \min(\max(s + a - G, l_s), u_s)$.

This function captures two scenarios: if the demand exceeds the inventory, resulting in backlogged orders, the excess demand $(l_s - s - a + G)$ is lost at a cost c_l per unit. If the inventory exceeds the capacity, the surplus units $(s + a - G - u_s)$ are disposed of at a disposal

cost c_d per unit. Together P_G and the non linear state transition function s' infer the transition density $p(s'|s,a)$.

The purchasing, holding, and backlogging costs per unit are denoted as c_p, c_h, and c_b, respectively. The MDP cost function at state s and action a is therefore defined as:

$$c(s,a) = c_p a + c_h \mathbb{E}\left[(s')_+\right] + c_b \mathbb{E}\left[(-s')_+\right] + c_d \mathbb{E}\left[(s+a-G-u_s)_+\right] + c_l \mathbb{E}\left[(l_s - s - a + G)_+\right]$$

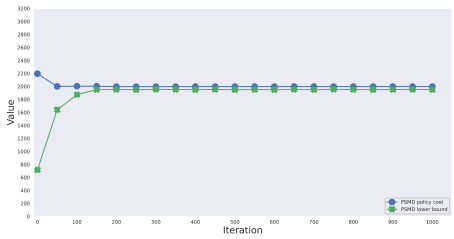
where $(s)_+ = \max\{0, s\}$, the expectations are computed with respect to the distribution P_G, $c_p = 20$, $c_d = 10$, $c_l = 100$, $c_h = 2$, and $c_b = 10$

In this study, we focused on the use of polynomial basis functions, tailored for a one-dimensional state space. These functions are constructed using a linear combination of the constant term one, the linear term s, and the quadratic term s^2.

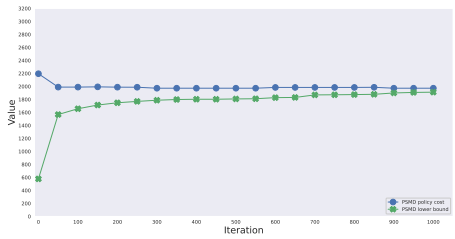
5.1.2 Results

For our specific instance, as set in [1], we set γ to 0.95, H and N are respectively 10 and 50.

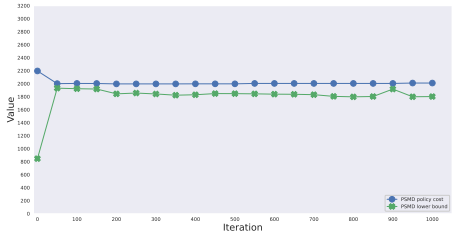
In the context of the Proximal Stochastic Mirror Descent method and Langevin dynamics, the parameter λ plays a crucial role in balancing the trade-off between the objective function and the KL-divergence from a reference distribution in the mirror-descent algorithm. This balance significantly impacts the shape of the distribution from which we sample. When the sampling distribution is log-concave, Langevin dynamics can efficiently explore the state space due to the single-peak, unimodal nature of log-concave distributions.



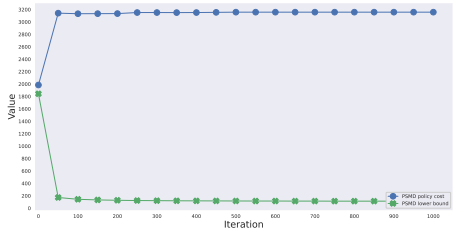
(a) Bounds using $\lambda = 0.1$.



(b) Bounds using $\lambda = 0.01$.



(c) Bounds using $\lambda = 0.001$.



(d) Bounds using $\lambda = 0.0001$.

Figure 1: Comparison between the policy cost and lower bound behaviour using different values of lambda

In the initial stages of our research, we encountered difficulties in achieving convergence with the algorithm. We were using the same parameters as in [1], which chose a λ that was too small to work effectively when sampling with Langevin dynamics. This small λ leaned the balance towards the objective function, potentially making the resulting distribution less log-concave and hindering the efficiency of Langevin dynamics. This led to slow convergence and less accurate samples, as shown in Figure 1.

Conversely, a large λ may lean the balance towards minimizing the KL-divergence from the reference distribution, potentially enhancing the log-concavity of the distribution. While this might improve the efficiency of Langevin dynamics, it could also result in an overly conservative

solution that does not fully exploit the structure of the problem to achieve the optimal objective function value. Therefore, the choice of λ is a critical consideration in the application of the PSMD method and Langevin dynamics, requiring tuning to ensure a balance between the objective function and the divergence from the reference distribution, while also considering the log-concavity of the resulting distribution.

After careful consideration and multiple trials, we decided to adjust the parameters. To fine-tune these parameters, we closely observed the decrease in the policy cost over the first 50 iterations. The detailed decreases are reported in Table I. It's worth noting that negative decreases, in this context, represent an increase in the policy cost. We found that setting λ to 1 and η to 0.1 provided the best results.

TABLE I: TUNING OF λ AND η

η	λ	Decrease in Policy Cost
0.1	0.0001	-1215
	0.001	187
	0.01	198
	0.1	194
	1	221
	10	234
0.01	0.0001	-629
	0.001	-12
	0.01	36
	0.1	80
	1	122
	10	-323

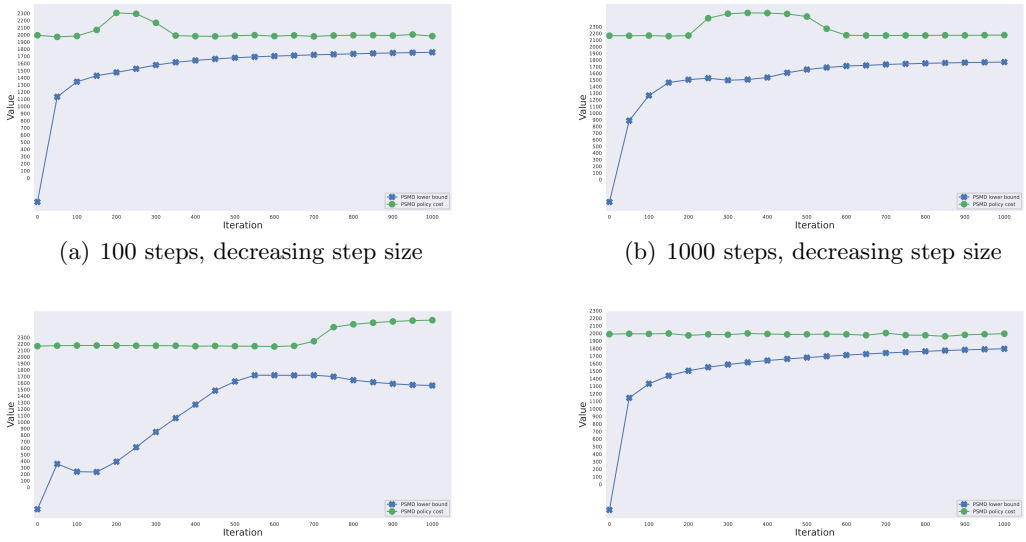
After the successful tuning of the parameters λ and η , our next step was to optimize the number of steps for the Langevin dynamics K and the step length ε in the Langevin dynamics update used for both the PSMD update and the lower bound computation. We conducted a series of experiments for the PSMD update, testing four different configurations:

- 100 steps with a decreasing step size, which started at 0.001
- 1000 steps with a decreasing step size, also starting at 0.001
- 100 steps with a fixed step size of 0.001
- 1000 steps with a fixed step size of 0.001

In these experiments, the step size was decreased in accordance with the square root of the number of Langevin dynamics steps. This approach allowed us to progressively refine the step size, enhancing the precision of the Langevin dynamics. The results of these configurations are illustrated in Figure 2. These figures depict the progression of the lower bound and policy cost for each configuration, providing a clear visual representation of the impact of different step sizes and step counts on the performance of the PSMD update.

In addition to these configurations, we also conducted experiments with larger and smaller starting step sizes. However, these configurations did not yield satisfactory results, producing less accurate samples and slower convergence. As such, they are not included in our discussion.

Upon analyzing the results of our experiments, we found that the most effective configuration was 1000 steps with a fixed step size of 0.001. This configuration provided the best balance between computational efficiency and accuracy of the results, enabling us to achieve a high-



(a) 100 steps, decreasing step size

(b) 1000 steps, decreasing step size

(c) 100 steps, fixed step size

(d) 1000 steps, fixed step size

Figure 2: Policy cost and lower bound changing numbers of step and step size

quality solution with a reasonable computational cost. Following the tuning process for the PSMD update, we applied a similar approach to optimize the parameters for the lower bound computation. We experimented with the same configurations of step sizes and counts, namely 100 steps with a decreasing step size starting at 0.001, 1000 steps with a decreasing step size starting at 0.001, 100 steps with a fixed step size of 0.001, and 1000 steps with a fixed step size of 0.001. The step size was again adjusted in accordance with the square root of the number of Langevin dynamics steps. After a thorough analysis of the results, we found that the most effective configuration for the lower bound computation mirrored that of the PSMD update. Specifically, 1000 steps with a fixed step size of 1e-3 yielded the best balance between

computational efficiency and accuracy of the results. This consistency in optimal parameters across different aspects of our algorithm further validates our approach and provides a clear direction for future experiments.

To evaluate the convergence of our algorithm towards the solution, we employed the optimality gap as a key metric. The optimality gap provides a measure of how close our algorithm's solution is to the best possible solution. It is computed using the formula:

$$\text{Optimality Gap} = \frac{\text{UB} - \text{Best LB}}{\text{Best LB}} \times 100\%$$

Where UB stands for Upper Bound and LB for Lower Bound. This percentage-based metric offers a clear perspective on the relative difference between the algorithm's output and the optimal solution. The choice to use a constant step size significantly improved the performance of the algorithm, after 1000 PSMD iterations, we achieved an optimality gap of 1.75%, in Figure 3 we can see the evolution of the optimality gap with respect to the iterations.

When comparing the results obtained using Langevin Dynamics and Metropolis Hastings, we observed distinct differences. The growth of the lower bound and the decrease in policy cost were more stable when using Langevin Dynamics. This stability is evident in the Figure 4 (b), where the progression of the lower bound and policy cost over iterations is shown.

In contrast, when using Metropolis Hastings, we noticed some abrupt fluctuations in the progression of the lower bound and policy cost as it can be seen in Figure 5, indicating instability.

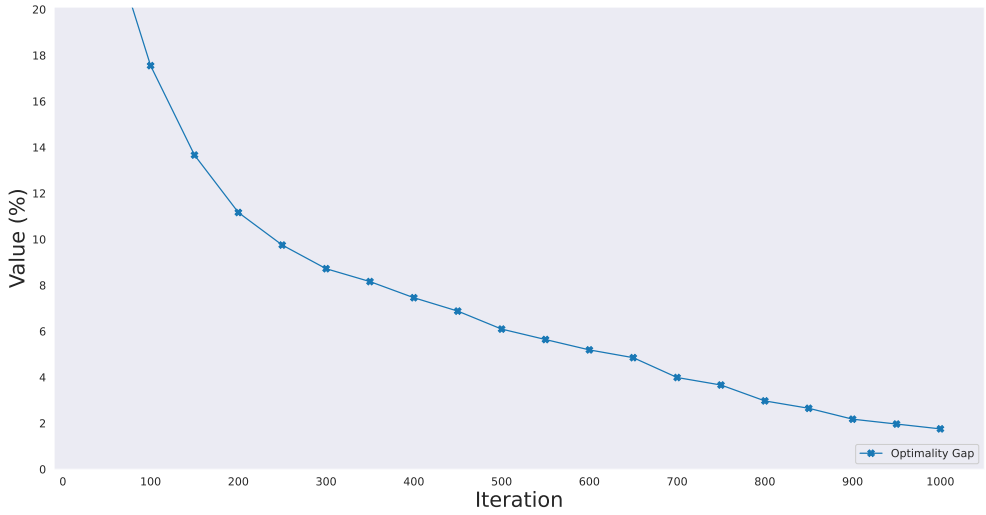
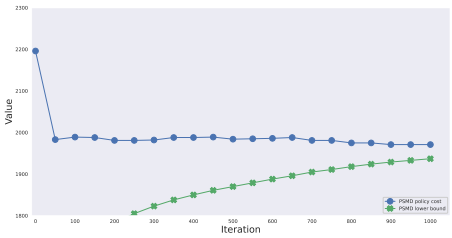


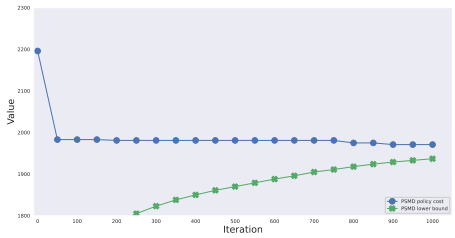
Figure 3: Optimality gap with respect to the iterations using Langevin Dynamics.

However, it is important to note that while Langevin Dynamics provided more stability, it required more iterations for the lower bound to grow as we can clearly see in Figure 4 (b).

In Figure 7, we present a comparative analysis of the points sampled using Langevin Dynamics and Metropolis Hastings. It is evident from the figure that the points sampled with Langevin Dynamics are predominantly concentrated in areas where the energy function is at its lowest. This concentration suggests that Langevin Dynamics is efficient in its sampling process. On the other hand, the points sampled with Metropolis Hastings are more dispersed across the energy function. This dispersion indicates a higher degree of variability in the Metropolis Hastings sampling process.

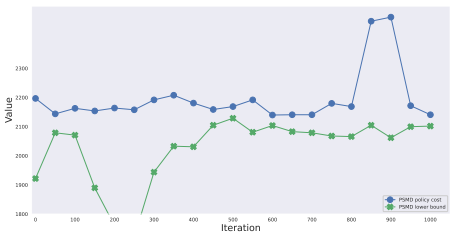


(a) Bounds at current iteration.

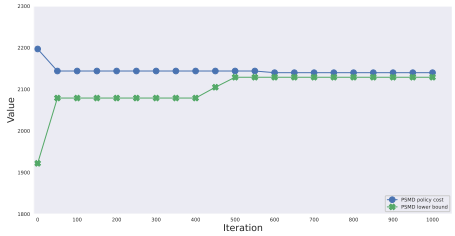


(b) Best historic bounds.

Figure 4: Evolution of the lower and upper bounds over 1000 iterations using Langevin Dynamics.



(a) Bounds at current iteration.



(b) Best historic bounds.

Figure 5: Evolution of the lower and upper bounds over 1000 iterations using Metropolis Hastings.

5.2 Perishable Inventory Control with Partial Backlogging and Lead Time

To assess the scalability of our methodology, we implemented it on Perishable Inventory Control instances featuring partially backlogged demand [49] [15] and lead time [50] [51] [49] [15]. This serves as a robust testbed for evaluating the algorithm's performance across varying

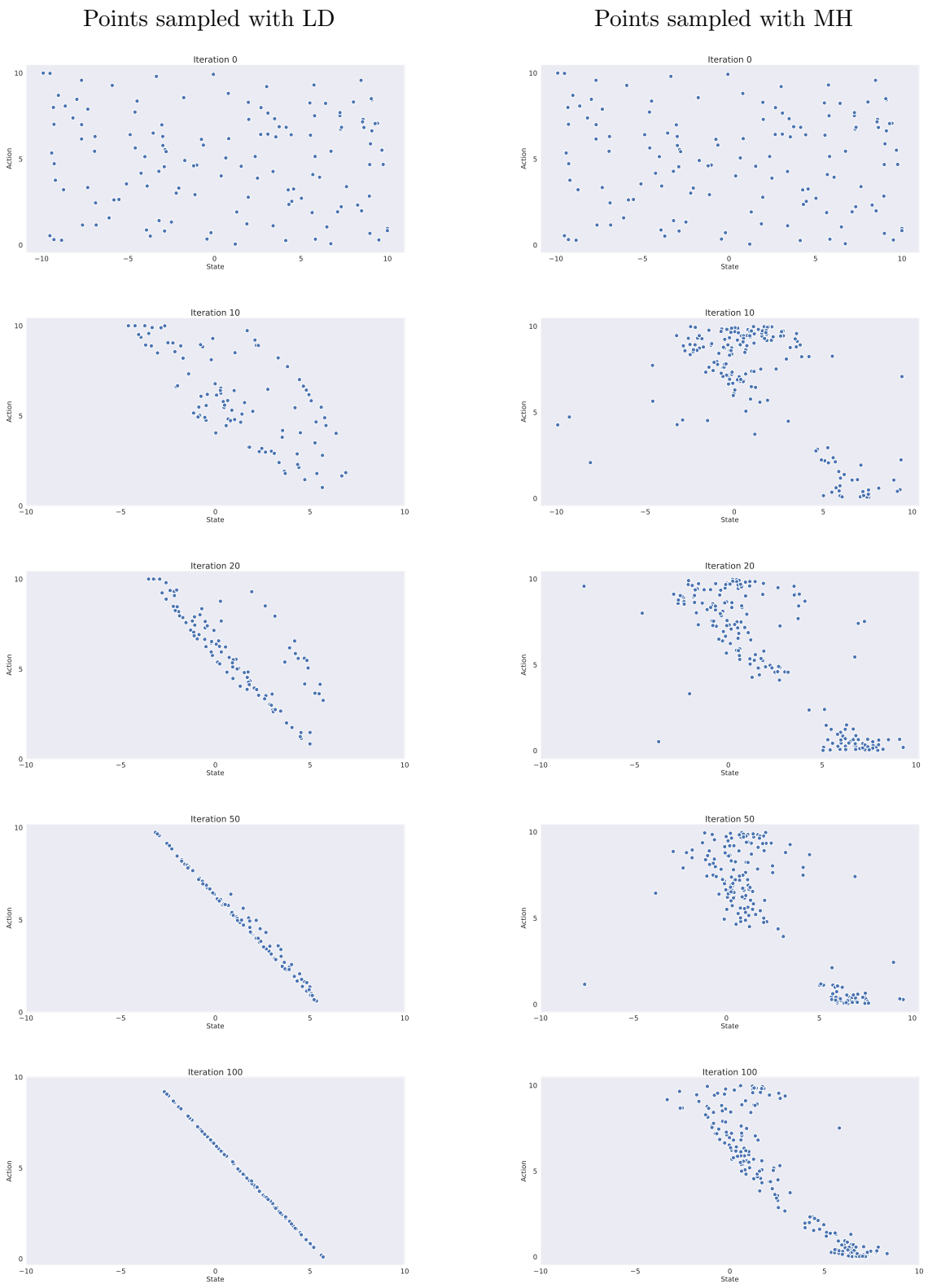


Figure 6: Comparison between the points sampled using Langevin Dynamics and Metropolis Hastings

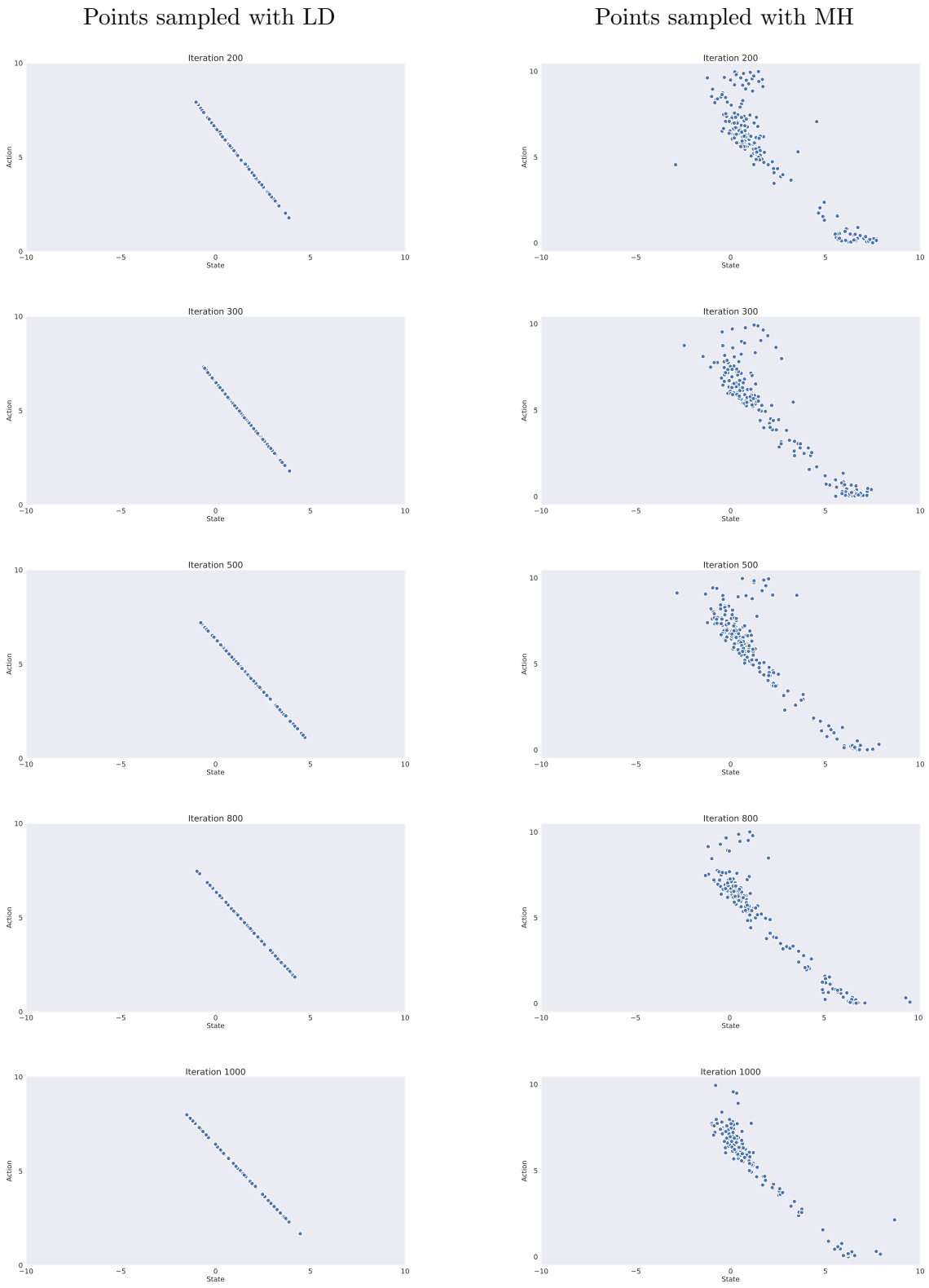


Figure 7: Comparison between the points sampled using Langevin Dynamics and Metropolis Hastings

dimensions. Additionally, we compare the results with Langevin Dynamics with the results presented in previous studies [1] and [52].

5.2.1 MDP Formulation and Instance

In our MDP model, we consider an order lead time of J periods and an item lifetime of I periods from the moment of receipt. The orders scheduled to arrive j periods from now are denoted by q_j, where $1 \leq j \leq J-1$, and the on-hand inventory with i periods of remaining lifetime is represented by z_i, for $0 \leq i \leq I-1$.

The state s is defined as the vector

$$s = (z_0, z_1, \ldots, z_{I-1}, q_1, q_2, \ldots, q_{J-1}) \in \mathbb{R}^{I+J-1}.$$

All inventories, both on-hand and in the pipeline, are required to be non-negative, with the exception of z_0. The value of z_0 can be either non-negative, indicating no backlogged orders and the disposal of any remaining expired items at the end of the current period, or negative, representing the quantity of backlogged orders calculated as $(G - \sum_{i=0}^{I-1} z_i)^+$. Here, G denotes the stochastic demand following distribution P_G.

The order quantity a is restricted to the interval $[0, \bar{a}]$. The lower bound for z_0 is $l_s < 0$, and its upper bound is $u_s = \bar{a}$. Consequently, $z_i \in [0, \bar{a}]$ for $i = 1, \ldots, I-1$ and $q_j \in [0, \bar{a}]$.

Assuming that demand occurs prior to the arrival of an order and is met using a first-in, first-out policy, the MDP transitions to a new state

$$s' = \left(\max \left\{ z_1 - (G - z_0)_+, l_s - \sum_{i=2}^{I-1} z_i \right\}, z_2, \ldots, z_{I-1}, q_1, q_2, \ldots, q_{J-1}, a \right)$$

The demand G follows a truncated normal distribution P_G in the interval $[0, 10]$ with mean 5 and a variable standard deviation σ.

The cost function $c(s, a)$ is defined as:

$$c(s,a) = \gamma^J c_p a + \mathbb{E}\left[c_h \left(\sum_{i=1}^{I-1} z_i - (G - z_0)_+ \right)_+ + c_b \left(G - \sum_{i=0}^{I-1} z_i \right)_+ + c_d (z_0 - G)_+ + c_l \left(l_s + G - \sum_{i=0}^{I-1} z_i \right)_+ \right]$$

where c_p, c_h, c_b, c_d, c_l are respectively the purchasing, holding, backlogging, disposal, lost sales costs. c_p and c_l for this instances are fixed to 20 and 100.

The maximum allowable backlogged orders are set to be equal to the maximum order level, i.e., $l_s = -\bar{a}$.

We focus on using the Fourier basis functions, which are versatile and applicable to any state space. Defined as $\phi(\cdot) = \cos(\cdot)$, their coefficients are determined as $\rho(\omega) : \omega_0 \sim \text{uniform}([-\pi, \pi])$ and $\omega_i \sim \text{normal}(0, \varrho)$ for $i \geq 1$, where ϱ is a user-specified parameter.

We examine 24 different perishable inventory control instances: twelve three-dimensional with $l = J = 2$, six five-dimensional with $l = 2$ and $J = 4$, and the remaining six ten-dimensional with $l = 5$ and $J = 6$. The cost function parameters, discount factor, maximum ordering level $\bar{a}$, and demand standard deviation σ are varied across these instances, with specific values provided in tables Table II, Table III, Table IV.

TABLE II: PARAMETERS FOR THE 3-D INSTANCES WITH $\sigma = 2$ AND $C_L = 100$

Three-dimensional state and action space	γ	c_h	c_d	c_b	$\bar{a}$
Instance 1	0.95	2	5	10	10
Instance 2		2	5	10	50
Instance 3		5	10	8	10
Instance 4		5	10	8	50
Instance 5		2	10	10	10
Instance 6		2	10	10	30
Instance 7	0.99	2	5	10	10
Instance 8		2	5	10	50
Instance 9		5	10	8	10
Instance 10		5	10	8	50
Instance 11		2	10	10	10
Instance 12		2	10	10	30

5.2.2 Results

In this section, we delve into the results obtained for the PIC with Partial Backlogging and Lead Time instances. For all instances under consideration, both H and N, which represent

TABLE III: PARAMETERS FOR THE 5-D INSTANCES WITH $\gamma = 0.95$ AND $C_L = 1000$

Five-dimensional state and action space	c_h	c_d	c_b	σ
Instance 13	1	8	2	5
Instance 14	1	8	2	2
Instance 15	1	2	8	5
Instance 16	1	2	8	2
Instance 17	2	8	5	5
Instance 18	2	8	5	2

TABLE IV: PARAMETERS FOR THE 10-D INSTANCES WITH $\gamma = 0.95$ AND $C_L = 1000$

Ten-dimensional state and action space	c_h	c_d	c_b	σ
Instance 19	1	8	2	5
Instance 20	1	8	2	2
Instance 21	1	2	8	5
Instance 22	1	2	8	2
Instance 23	2	8	5	5
Instance 24	2	8	5	2

the number of samples drawn from $p(\cdot, s, a)$ and y_t, are set to values of 10 and 50, respectively. All instances and configurations utilized in this section are analogous to those detailed in [52], allowing for a direct comparison between our findings.

We employ Fourier basis functions to represent our value function approximations. The number of these basis functions varies based on the dimensionality of the instances:

- 150 basis functions for the three-dimensional instances,

- 300 for the five-dimensional instances,
- 600 for the ten-dimensional instances.

For the bandwidth parameter, a mixture of 10^{-2} and 10^{-3} is utilized.

With these configurations in place, we proceed to discuss the results and insights derived from the application of our approach to the PIC instances.

Our initial step involved fine-tuning the parameters λ and η. We focused our tuning efforts on three representative instances, each corresponding to a distinct dimensionality of the state-action space: Instance 1, Instance 13, and Instance 19. The optimized values derived from these instances were then consistently applied to all other instances of the same dimensionality. Our evaluation metric was the decline in policy cost observed over the initial 50 iterations of the algorithm. Table V elucidates the policy cost reduction achieved with various λ and η combinations. Based on the table, for the 3D case, $\lambda = 0.001$ and $\eta = 0.1$ emerged as the optimal choice. Meanwhile, for both five and ten-dimensional cases, $\lambda = 0.0001$ and $\eta = 0.1$ were found to be the most effective.

Following the calibration of λ and η, our subsequent focus shifted to fine-tuning the number of Langevin dynamics steps, K, and the step size for the Langevin dynamics update, ε. Contrary to varying step sizes, we opted for constant ones, drawing from the insights gained in Section 5.1.2 which highlighted their superior performance. Our evaluation metric remained consistent, examining the decline in policy cost over the initial 50 iterations. The outcomes of this tuning process are detailed in Table VI.

TABLE V: TUNING OF λ AND η FOR THREE REPRESENTATIVE INSTANCES

η	λ	Decrease in Policy Cost		
		Instance 1	Instance 13	Instance 19
0.1	0.0001	297	843	1354
	0.001	639	594	726
	0.01	332	66	242
	0.1	-58	-51	-58
	1	-101	-113	-337
0.01	0.0001	123	137	268
	0.001	165	138	197
	0.01	84	82	186
	0.1	12	-68	74
	1	-143	-46	49

TABLE VI: TUNING OF K AND ε FOR THREE REPRESENTATIVE INSTANCES

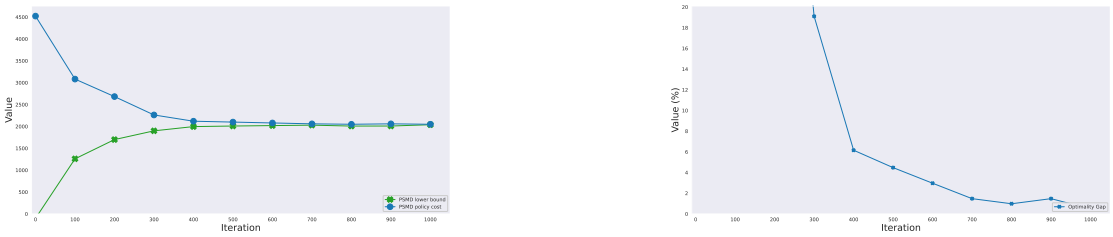
K	ε	Decrease in Policy Cost		
		Instance 1	Instance 13	Instance 19
100	0.01	24	19	-25
	0.001	63	4	46
1000	0.01	118	287	498
	0.001	639	843	1354

Optimal outcomes across all instances were achieved with $K = 1000$ and $\varepsilon = 0.001$. A natural inquiry might be the reason behind not experimenting with larger K values paired with smaller ε. The reason is tied to the sequential nature of Langevin dynamics updates. Given the sequential nature of Langevin dynamics sampling algorithm, elevating the value of K would substantially extend the running time. Given our primary aim to scale approximate linear

programs and the satisfactory results already attained with the current parameters, there is little incentive to venture into greater K values.

Having meticulously fine-tuned our hyperparameters, we proceeded to run our algorithm using this optimal set of parameters. The outcomes, specifically the optimality gap achieved, are detailed in Table VII. For a clearer comparative analysis, we juxtaposed our results with the baseline values from [52], which used FALP.

We observed that Langevin dynamics yielded comparable results even for the larger instances. Figure 8, Figure 9, and Figure 10 illustrate the evolution of the upper bound in relation to the lower bound for the three representative instances, as well as the progression of the optimality gap.



(a) Bounds at current iteration.

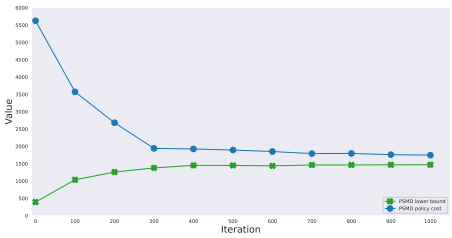
(b) Optimality Gap.

Figure 8: Three dimensional instance evolution of (a) lower and upper bound (b) optimality gap over 1000 iterations.

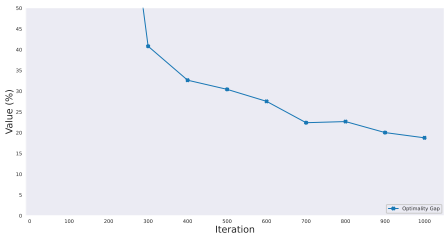
TABLE VII: COMPARISON BETWEEN OPTIMALITY GAP REACHED WITH PSMD AND FALP

Dimensionality of the State-Action space		Optimality Gap	
		PSMD	FALP
3 D	Instance 1	0.5 %	0.1 %
	Instance 2	6.1 %	5.9 %
	Instance 3	0.3 %	0.2 %
	Instance 4	0.5 %	0.2 %
	Instance 5	0.5 %	0.2 %
	Instance 6	1.9 %	1.7 %
	Instance 7	0.6 %	0.2 %
	Instance 8	5.9 %	5.6 %
	Instance 9	0.4 %	0.3 %
	Instance 10	2.2 %	1.5 %
	Instance 11	0.8 %	0.3 %
	Instance 12	1.5 %	1.5 %
5 D	Instance 13	18.8 %	19.6 %
	Instance 14	22.4 %	21.0 %
	Instance 15	17.3 %	15.6 %
	Instance 16	14.7 %	12.1 %
	Instance 17	17.2 %	15.9 %
	Instance 18	18.0 %	16.1 %
10 D	Instance 19	14.1 %	13.0 %
	Instance 20	6.7 %	6.1 %
	Instance 21	12.0 %	11.4 %
	Instance 22	9.6 %	7.0 %
	Instance 23	16.5 %	14.5 %
	Instance 24	10.4 %	9.1 %

In this section, we presented a comprehensive analysis of the results obtained for the PIC with Partial Backlogging and Lead Time instances. The results, as showcased in the tables and figures, indicate that our approach, utilizing Langevin dynamics, is effective in handling even the larger instances. The comparative analysis with FALP further underscores the efficacy of

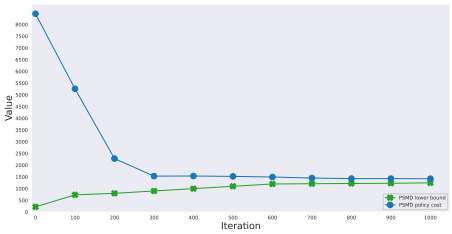


(a) Bounds at current iteration.

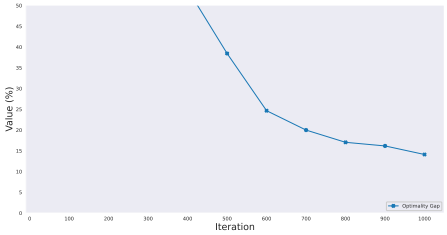


(b) Optimality Gap.

Figure 9: Five dimensional instance evolution of (a) lower and upper bound (b) optimality gap over 1000 iterations.



(a) Bounds at current iteration.



(b) Optimality Gap.

Figure 10: Ten dimensional instance evolution of (a) lower and upper bound (b) optimality gap over 1000 iterations.

our method. As we transition to subsequent sections, these findings provide a solid foundation, reinforcing the potential of our approach in addressing complex optimization challenges in various dimensions.

5.3 Computational Insights: From Efficiency to Scalability

In the preceding section, we delved into the results obtained on the PIC instances, highlighting the efficacy of our algorithm in harnessing Langevin dynamics to achieve encouraging outcomes. While the performance metrics and optimality gaps provided a promising narrative, it is essential to also understand the computational underpinnings that drive these results.

In this concluding section, our focus will be on the computational intricacies of Langevin dynamics. We will discuss the scalability of our approach. As we scale up the problem dimensions, it is crucial to observe how our methodology responds. We will also explore any complications that arise and discuss potential strategies for their mitigation or resolution. By the end of this discourse, we aspire to present a holistic view of our approach, encompassing both its theoretical prowess and its computational practicality.

All computational code for this research was written using PyTorch [53], a widely-used deep learning framework. The decision to use PyTorch was strategic, given its native support for GPU acceleration, which is essential for tensor operations. This feature is especially beneficial when sampling with Langevin dynamics, as tensor-based calculations can be parallelized and executed more rapidly on a GPU compared to CPU-based computations.

For the experiments conducted in this study, a single NVIDIA Tesla T4 GPU was employed. The Tesla T4 is designed for inference workloads and is equipped with 16 GB of GDDR6 memory and 320 Turing Tensor Cores. It provides a peak performance of 8.1 TFLOPS for single-precision tasks, making it a suitable choice for our computational needs. Utilizing this specific GPU ensured that our computations were not only accurate but also efficiently executed.

Langevin dynamics offer a distinct advantage when it comes to sampling from high-dimensional distributions, making them particularly suited for complex problems that span multiple dimensions. One of the primary advantages is the absence of an accept-reject step, which is a characteristic feature of the Metropolis-Hastings algorithm. This absence significantly accelerates the sampling process. The Metropolis-Hastings method, while powerful, can become computationally intensive due to the repeated evaluations required in its accept-reject mechanism, especially in high-dimensional spaces. By bypassing this step, Langevin dynamics streamline the sampling process, leading to faster convergence and reduced computational overhead.

As we highlighted in previous chapters, one of the main advanteges of Langevin dynamics is that it remains largely unaffected by the problem's dimensionality. Thanks to its continuous updates and the absence of a accept-reject step. In many sampling techniques, the accept-reject step becomes increasingly complex as dimensionality rises, often leading to inefficiencies. However, Langevin dynamics avoids this issue, allowing for more efficient exploration of the state space, even in high-dimensional scenarios. This dimension-independent behavior is a cornerstone of our thesis and plays a fundamental role in the results we observe. This implies that the execution time of the algorithm remains relatively stable, irrespective of the dimensionality of the distribution from which we are sampling.

Given these attributes, we expect that the execution time of our approach, which leverages Langevin dynamics, will exhibit a level of independence from the dimensionality of the problem at hand.

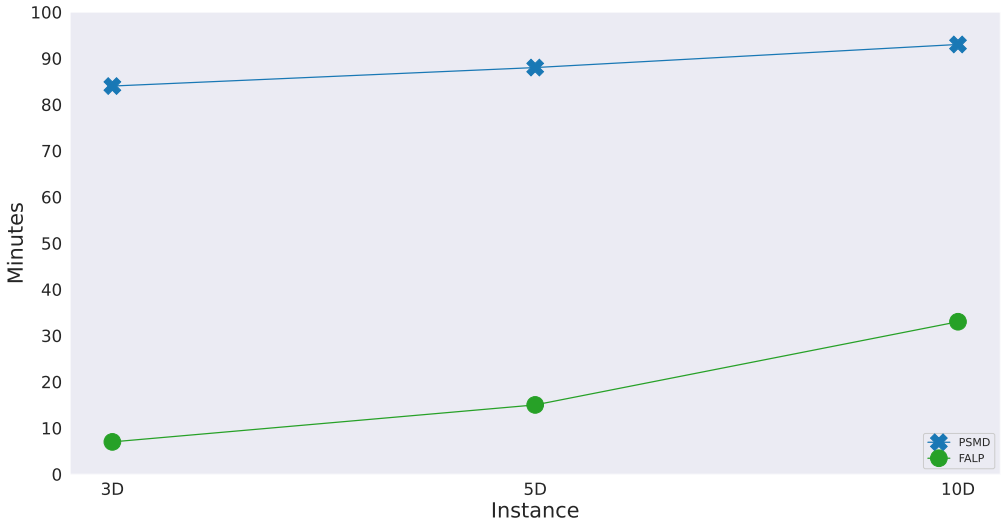


Figure 11: Comparison of Average Running Time using PSMD and FALP for 3D, 5D, and 10D Instances Over 1000 Iterations .

As depicted in Figure 11, which presents the average running time of instances segmented by the dimensionality of the state-action space, the instances using PSMD required 84, 88, and 93 minutes for three, five, and ten dimensions respectively. Transitioning from a three-dimensional instance to a five-dimensional one led to a 4.6% increase in total runtime. In a similar vein, progressing from five to ten dimensions resulted in a 5.4% increment. Such a modest increase in runtime is expected as dimensionality grows, given the heightened computational complexity of associated operations. PyTorch's tensor operations are optimized such that even as the dimensionality of the tensors increases, the operations do not take much longer. This optimization ensures that the computational complexity does not scale with dimension in a substantial

way, allowing for relatively consistent performance across different dimensionalities. We also attempted to execute the code using Metropolis-Hastings sampling with a set runtime limit of three hours. However, for all three instances, the code had not completed its execution by the end of the allotted time.

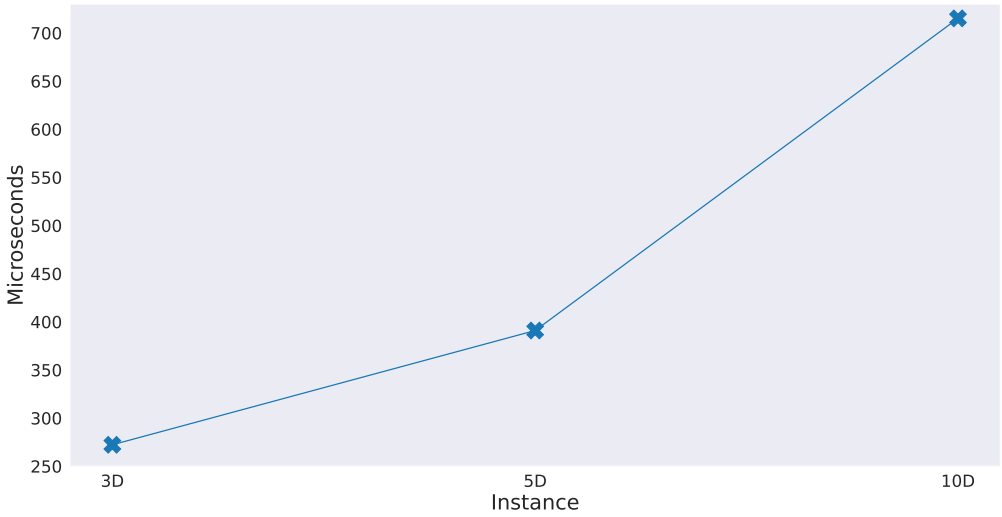


Figure 12: Comparison of average uniform sampling time for 3D, 5D, and 10D spaces.

A potential factor that could significantly impact running times is the reliance on external sampling functions, such as those from *torch.distributions*. The efficiency of these functions tends to degrade as the dimensionality of the state space increases. For instance, in Figure 12, the average running times over 1,000 trials required to sample 1,000,000 samples from 3D, 5D,

and 10D State-Action spaces are presented. It is evident that as the state dimensionality rises, obtaining uniform samples from the state-action space becomes increasingly computationally demanding.

Another consideration is the potential instability of certain functions. In our implementation, we employ the $torch-truncnorm$ package [54] to represent our demand using a truncated normal distribution. This package is grounded in PyTorch's normal distribution framework, which is documented to have stability concerns [55] during the sampling process.

It is important to note that our current implementation runs on a single GPU. However, the inherent structure of our code lends itself to parallelization, which could substantially reduce running times. Utilizing the parallel processing capabilities inherent to GPUs, particularly when paired with frameworks like PyTorch, can result in notable computational efficiencies. Distributing the computational tasks across multiple threads can expedite convergence, a benefit that becomes increasingly pronounced in high-dimensional spaces.

In reflecting upon our computational journey, it is evident that our algorithm exhibits resilience to increasing dimensionality. The observed rise in running time, while present, remains notably restrained even as we venture into higher-dimensional spaces. This modest escalation in computational demand, juxtaposed against the complexity of the problems we address, underscores the potential of our approach to be extended to even larger instances.

However, it is imperative to recognize that no computational methodology is without its intricacies. Our approach, while robust, is not exempt from the challenges posed by external dependencies and potential instabilities.

Looking ahead, the ubiquity of GPUs in modern computational setups presents an enticing opportunity. As these powerful tools become increasingly accessible, harnessing the collective power of multiple GPUs could offer significant advancements in computational efficiency. The prospect of parallelizing our algorithm and distributing its workload across several GPUs holds promise for even more rapid convergence and enhanced efficiency.

In conclusion, while our current methodology has showcased significant potential and promise, the landscape of computational optimization is dynamic and expansive. As we stand at this juncture, it is clear that the path forward will demand continuous exploration, adaptation, and learning. Embracing these tenets will be instrumental in navigating the challenges that lie ahead, propelling us towards new frontiers of efficiency and scalability.

CHAPTER 6

CONCLUSION

As this research journey culminates, we reflect upon our endeavors, achievements, and challenges. Our primary objective was to investigate the capabilities of Langevin Dynamics in the context of optimization, specifically when sampling from an Energy-Based Model.

The central questions that guided our exploration were:

- Does Langevin dynamics sampling function effectively in the context of optimization?
- Can Langevin dynamics sampling, when applied to constraint violation learning, scale efficiently to higher dimensions?

To address these pivotal questions, we turned our attention to Perishable Inventory Control instances, progressively increasing the state-action space dimensionality. This approach provided a tangible and practical framework to test the efficacy and scalability of Langevin dynamics in real-world optimization scenarios.

Regarding our first question, while the efficacy of Langevin dynamics in sampling is well-established in certain contexts, its application to non-log concave functions is less explored. In this work, we delved deeper into this area, adapting and applying Langevin dynamics to a context where its performance was not immediately guaranteed. This adaptation required innovative approaches and considerations, emphasizing the distinctiveness of our work in harnessing Langevin dynamics for this specific setting. Given that the functions we aimed to sample from

in the PSMD algorithm are Energy-Based Models, the choice of Langevin dynamics for sampling was natural. Our findings, as elaborated in the preceding chapters, were promising. The algorithm adeptly handled varying dimensions ranging from simpler two-dimensional instances to the more intricate ten-dimensional ones. When compared to methodologies like PSMD using Metropolis-Hastings and FALP, our approach stood out.

Addressing the second question on the scalability of Langevin dynamics, when applied to constraint violation learning, we delved deeper into the computational intricacies. High-dimensional optimization problems are notorious for their computational demands. The "curse of dimensionality" is a well-known challenge, where the solution space grows exponentially with the problem's dimensionality, making exhaustive searches infeasible. However, Langevin dynamics, characterized by its iterative and adaptive sampling techniques, presents a promising counter to this predicament.

Given this backdrop, we anticipated that Langevin dynamics would demonstrate resilience and consistency, even as we scaled the dimensionality of our problems.

Our experiments, provided a clear trajectory of the algorithm's performance. The minimal escalation in computational time, even with the increasing problem dimensions, affirmed the algorithm's robustness and efficiency.

A novel aspect of our research was the exploration of the parameter lambda's role in controlling the log concavity of the distribution used for sampling. To our knowledge, this is the pioneering work that delves into Langevin dynamics sampling applied to a distribution where log concavity can be modulated through a parameter, lambda. Our experiments provided valu-

able insights into optimal parameter settings, paving the way for more efficient and effective implementations in future applications.

Yet, no research journey is devoid of challenges, and ours was no exception. A significant limitation we grappled with was the dependency on external sampling functions, notably those from *torch.distributions*. As we ventured into higher state dimensionality, these functions began to show signs of reduced efficiency, complicating the uniform sampling process from the state-action space. Additionally, we encountered stability issues, particularly when modeling demand using a truncated normal distribution, necessitating meticulous navigation and problem-solving.

A significant limitation arises from the bounded nature of our state-action space. Unlike an unrestricted domain, our state-action space is confined, necessitating the inclusion of a projection step following the Langevin dynamics update. This step ensures that the sampled points remain within the designated boundaries of the space. This projection was feasible primarily because of the simple structure of our state-action set. However, in more complex applications, such a direct projection might not be viable due to the intricate structure of the state-action space. It is also worth highlighting that the convergence rate of the Projected Langevin dynamics, which we utilized, is inferior to that of the standard Langevin dynamics. This discrepancy in convergence rates underscores the challenges and trade-offs one might encounter when adapting the algorithm to more complex scenarios.

The introduction of Langevin dynamics into the PSMD framework has undeniably enhanced its capabilities. However, this integration has also introduced an added layer of complex-

ity in terms of hyperparameter tuning. PSMD, in its original form, already necessitated the fine-tuning of several parameters to ensure optimal performance. With the incorporation of Langevin dynamics at two distinct points within the code, additional parameters have emerged that require meticulous calibration. This proliferation of parameters has significantly extended the tuning process. Given the expansive parameter space and the computational demands associated with exhaustive tuning, we had to strategically limit the range and combinations of parameters we explored. This constraint, while necessary for the feasibility of our experiments, underscores the challenge of hyperparameter sensitivity and the trade-offs researchers often face between comprehensive exploration and practical execution.

Future work could focus on several promising avenues to further enhance the capabilities and efficiency of our approach. As we have seen, the computational demands of our approach can be intensive, and a deeper exploration into distributed computing techniques stands as a promising avenue for future research. By leveraging multiple GPUs, we can significantly expedite computations, making it feasible to tackle larger and more complex problem instances. The inherent parallel nature of many optimization and sampling algorithms, especially when integrated with frameworks like PyTorch, makes them prime candidates for such distributed approaches.

Building on the merits of Langevin dynamics in sampling, there is an expansive landscape of sampling techniques suggesting room for improvement or variation within the Langevin dynamics framework itself. Future endeavors could delve into modifications or alternative techniques that still fall under the Langevin dynamics umbrella but might be better suited

for specific challenges. This exploration becomes especially pertinent when dealing with state-action spaces of intricate geometries or when traditional Langevin dynamics face challenges.

Furthermore, as we push the boundaries of what our algorithms can achieve, understanding their behavior in extreme conditions becomes paramount. Rigorous scalability studies, especially targeting very high-dimensional problems, could be the next step. Such studies would not only test the limits of our current approach but also provide insights into potential bottlenecks or areas of improvement. Systematically increasing problem complexity and observing algorithmic performance can offer valuable information that guides subsequent refinements and innovations.

This research stands as one of the pioneering efforts to incorporate Langevin dynamics to solve large scale linear programs. While traditional optimization techniques have their merits, the integration of Langevin dynamics offers a fresh perspective and a set of tools with complementary benefits. However, as with any nascent integration, there remains a vast expanse of uncharted territory. The full breadth and depth of what Langevin dynamics can offer to solve large scale linear programs is yet to be fully realized.

Furthermore, the performance of Langevin dynamics, in Constraint Violation Learning when applied to diverse types of problems, remains an open question.

In conclusion, while this thesis has shed light on the potential of Langevin dynamics in optimization, it also paves the way for future investigations. We are at the early stages of this exploration, and the future holds promise for further advancements in this exciting confluence of Langevin dynamics and optimization.

CITED LITERATURE

1. Lin, Q., Nadarajah, S., and Soheili, N.: Revisiting approximate linear programming: Constraint-violation learning with applications to inventory control and energy storage. Management science, 66(4):1544–1562, 2020.

2. Nguyen, T. V., Jagatap, G., and Hegde, C.: Provable compressed sensing with generative priors via langevin dynamics, 2021.

3. Li, C., Chen, C., Carlson, D., and Carin, L.: Preconditioned stochastic gradient langevin dynamics for deep neural networks. Proceedings of the AAAI Conference on Artificial Intelligence, 30(1), Feb. 2016.

4. Qin, L., Welleck, S., Khashabi, D., and Choi, Y.: Cold decoding: Energy-based constrained text generation with langevin dynamics. In Advances in Neural Information Processing Systems, eds. S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, volume 35, pages 9538–9551. Curran Associates, Inc., 2022.

5. Maken, F. A., Ramos, F., and Ott, L.: Estimating motion uncertainty with bayesian icp. In 2020 IEEE International Conference on Robotics and Automation (ICRA), pages 8602–8608. IEEE, 2020.

6. Kamalaruban, P., Huang, Y.-T., Hsieh, Y.-P., Rolland, P., Shi, C., and Cevher, V.: Robust reinforcement learning via adversarial training with langevin dynamics. In Advances in Neural Information Processing Systems, eds. H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, volume 33, pages 8127–8138. Curran Associates, Inc., 2020.

7. Chau, N. H., Moulines, É., Rásonyi, M., Sabanis, S., and Zhang, Y.: On stochastic gradient langevin dynamics with dependent data streams: The fully nonconvex case. SIAM Journal on Mathematics of Data Science, 3(3):959–986, 2021.

8. Wu, Z. and Li, H.: Stochastic gradient langevin dynamics for massive mimo detection. IEEE Communications Letters, 26(5):1062–1065, 2022.

9. Trick, M. A. and Zin, S. E.: Spline approximations to value functions: linear programming approach. Macroeconomic Dynamics, 1(1):255–277, 1997.

CITED LITERATURE (continued)

10. Adelman, D.: A price-directed approach to stochastic inventory/routing. Operations Research, 52(4):499–514, 2004.

11. Adelman, D.: Dynamic bid prices in revenue management. Operations Research, 55(4):647–661, 2007.

12. De Farias, D. P. and Van Roy, B.: On constraint sampling in the linear programming approach to approximate dynamic programming. Mathematics of operations research, 29(3):462–478, 2004.

13. Farias, V. F. and Van Roy, B.: Tetris: A study of randomized constraint sampling. Probabilistic and randomized methods for design under uncertainty, pages 189–201, 2006.

14. Restrepo, M.: Computational methods for static allocation and real-time redeployment of ambulances. 2008.

15. Sun, P., Wang, K., and Zipkin, P.: Quadratic approximation of cost functions in lost sales and perishable inventory control problems. Fuqua School of Business, Duke University, Durham, NC, 2014.

16. Hernández-Lerma, O. and Lasserre, J. B.: Discrete-time Markov control processes: basic optimality criteria, volume 30. Springer Science & Business Media, 2012.

17. Rahimi, A. and Recht, B.: Uniform approximation of functions with random bases. In 2008 46th Annual Allerton Conference on Communication, Control, and Computing, pages 555–561, 2008.

18. Koike, T. and Hofert, M.: Markov chain monte carlo methods for estimating systemic risk allocations. Risks, 8(1), 2020.

19. Beskos, A., Roberts, G., and Stuart, A.: Optimal scalings for local metropolis–hastings chains on nonproduct targets in high dimensions. 2009.

20. Langevin, P.: Sur la théorie du mouvement brownien. Compt. Rendus, 146:530–533, 1908.

21. Doob, J. L.: The brownian movement and stochastic equations. Annals of Mathematics, pages 351–369, 1942.

CITED LITERATURE (continued)

22. Doob, J.: Edward nelson, dynamical theories of brownian motion. The Annals of Mathematical Statistics, 39(2):686–686, 1968.

23. Grenander, U. and Miller, M. I.: Representations of knowledge in complex systems. Journal of the Royal Statistical Society: Series B (Methodological), 56(4):549–581, 1994.

24. Raginsky, M., Rakhlin, A., and Telgarsky, M.: Non-convex learning via stochastic gradient langevin dynamics: a nonasymptotic analysis, 2017.

25. Gelfand, S. B. and Mitter, S. K.: Recursive stochastic algorithms for global optimization in $\mathbb{R}^d$. SIAM Journal on Control and Optimization, 29(5):999–1018, 1991.

26. Durmus, A. and Moulines, E.: High-dimensional bayesian inference via the unadjusted langevin algorithm, 2018.

27. He, Y., Balasubramanian, K., and Erdogdu, M. A.: Heavy-tailed sampling via transformed unadjusted langevin algorithm. arXiv preprint arXiv:2201.08349, 2022.

28. Welling, M. and Teh, Y. W.: Bayesian learning via stochastic gradient langevin dynamics. In Proceedings of the 28th international conference on machine learning (ICML-11), pages 681–688, 2011.

29. Ahn, S., Korattikara, A., and Welling, M.: Bayesian posterior sampling via stochastic gradient fisher scoring. arXiv preprint arXiv:1206.6380, 2012.

30. Hasenclever, L., Webb, S., Lienart, T., Vollmer, S., Lakshminarayanan, B., Blundell, C., and Teh, Y. W.: Distributed bayesian learning with stochastic natural gradient expectation propagation and the posterior server. The Journal of Machine Learning Research, 18(1):3744–3780, 2017.

31. Li, C., Chen, C., Carlson, D., and Carin, L.: Preconditioned stochastic gradient langevin dynamics for deep neural networks. In Proceedings of the AAAI conference on artificial intelligence, volume 30, 2016.

32. Aneja, J., Schwing, A., Kautz, J., and Vahdat, A.: Ncp-vae: Variational autoencoders with noise contrastive priors. 2020.

33. Sato, I. and Nakagawa, H.: Approximation analysis of stochastic gradient langevin dynamics by using fokker-planck equation and ito process. In International Conference on Machine Learning, pages 982–990. PMLR, 2014.

CITED LITERATURE (continued)

34. Dalalyan, A. S.: Theoretical guarantees for approximate sampling from smooth and log-concave densities. Journal of the Royal Statistical Society Series B: Statistical Methodology, 79(3):651–676, 2017.

35. Mou, W., Flammarion, N., Wainwright, M. J., and Bartlett, P. L.: Improved bounds for discretization of langevin diffusions: Near-optimal rates without convexity. Bernoulli, 28(3):1577–1601, 2022.

36. Du, Y. and Mordatch, I.: Implicit generation and modeling with energy based models. Advances in Neural Information Processing Systems, 32, 2019.

37. Xie, J., Lu, Y., Zhu, S.-C., and Wu, Y.: A theory of generative convnet. In Proceedings of The 33rd International Conference on Machine Learning, eds. M. F. Balcan and K. Q. Weinberger, volume 48 of Proceedings of Machine Learning Research, pages 2635–2644, New York, New York, USA, 20–22 Jun 2016. PMLR.

38. Balasubramanian, K., Chewi, S., Erdogdu, M. A., Salim, A., and Zhang, S.: Towards a theory of non-log-concave sampling: first-order stationarity guarantees for langevin monte carlo. In Conference on Learning Theory, pages 2896–2923. PMLR, 2022.

39. Chewi, S., Erdogdu, M. A., Li, M. B., Shen, R., and Zhang, M.: Analysis of langevin monte carlo from poincar\'e to log-sobolev. arXiv preprint arXiv:2112.12662, 2021.

40. Ma, Y.-A., Chatterji, N. S., Cheng, X., Flammarion, N., Bartlett, P. L., and Jordan, M. I.: Is there an analog of nesterov acceleration for gradient-based mcmc? 2021.

41. Lyu, H., Sha, N., Qin, S., Yan, M., Xie, Y., and Wang, R.: Advances in neural information processing systems. Advances in neural information processing systems, 32, 2019.

42. Durmus, A. and Moulines, E.: Nonasymptotic convergence analysis for the unadjusted langevin algorithm. 2017.

43. Majka, M. B., Mijatović, A., and Szpruch, Ł.: Nonasymptotic bounds for sampling algorithms without log-concavity. 2020.

44. Erdogdu, M. A. and Hosseinzadeh, R.: On the convergence of langevin monte carlo: The interplay between tail growth and smoothness. In Conference on Learning Theory, pages 1776–1822. PMLR, 2021.

45. Gurobi Optimization, LLC: Gurobi Optimizer Reference Manual, 2023.

CITED LITERATURE (continued)

46. Nahmias, S. and Smith, S. A.: Optimizing inventory levels in a two-echelon retailer system with partial lost sales. Management Science, 40(5):582–596, 1994.

47. Rabinowitz, G., Mehrez, A., Chu, C.-W., and Patuwo, B. E.: A partial backorder control for continuous review (r, q) inventory system with poisson demand and constant lead time. Computers & operations research, 22(7):689–700, 1995.

48. Benjaafar, S., ElHafsi, M., and Huang, T.: Optimal control of a production-inventory system with both backorders and lost sales. Naval Research Logistics (NRL), 57(3):252–265, 2010.

49. Wang, K.: Heuristics for Inventory Systems Based on Quadratic Approximation of L-Convex Value Functions. Doctoral dissertation, Duke University, 2014.

50. Karaesmen, I. Z., Scheller-Wolf, A., and Deniz, B.: Managing perishable and aging inventories: review and future research directions. Planning Production and Inventories in the Extended Enterprise: A State of the Art Handbook, Volume 1, pages 393–436, 2011.

51. Chen, X., Pang, Z., and Pan, L.: Coordinating inventory control and pricing strategies for perishable products. Operations Research, 62(2):284–300, 2014.

52. Pakiman, P., Nadarajah, S., Soheili, N., and Lin, Q.: Self-guided approximate linear programs. arXiv preprint arXiv:2001.02798, 2020.

53. Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., and Chintala, S.: Pytorch: An imperative style, high-performance deep learning library, 2019.

54. Anton Obukhov, Jack Morton, P. S.: torch_truncnorm. `https://github.com/toshas/torch_truncnorm`. Accessed: 2023-02-21.

55. Adding a numerically stable inverse cdf of standard gaussian. `https://discuss.pytorch.org/t/adding-a-numerically-stable-inverse-cdf-of-standard-gaussian/106542`. Accessed: 2023-03-17.

VITA

NAME	Virginia Marcante
EDUCATION	
	Master of Science in “Computer Science”, University of Illinois at Chicago, Dicember 2023, USA
	Master of Science in “ Data Science and Engineering ”, October 2023, Polytechnic of Turin, Italy
	Bachelor’s Degree in ”Mathematics for Engineering”, July 2021, Polytechnic of Turin, Italy
LANGUAGE SKILLS	
Italian	Native speaker
English	Full working proficiency
	2021 - IELTS examination (8/9)
	A.Y. 2022/23 One Year of study abroad in Chicago, Illinois
	A.Y. 2021/22. Lessons and exams attended exclusively in English
German	Fluent
SCHOLARSHIPS	
Summer 2023	Graduate Assistantship (GA) position (20 hours/week) with full tuition waiver plus monthly stipend
Spring 2023	Graduate Assistantship (GA) position (10 hours/week) with full tuition waiver plus monthly stipend
Fall 2022	Italian scholarship for TOP-UIC students