Enforcing Constraints in Learning-Augmented Online Optimization: Theory and Applications to Energy Systems

Thesis by
James Y. Chen

In Partial Fulfillment of the Requirements for the
Degree of
Bachelor of Science

CALIFORNIA INSTITUTE OF TECHNOLOGY
Pasadena, California

2024
Submitted June 7th, 2024

James Y. Chen
ORCID: 0009-0005-5003-2996

ACKNOWLEDGEMENTS

I want to begin by expressing my gratitude towards Professor Adam Wierman and Nico Christianson for their invaluable mentorship and guidance, both for my thesis and more generally throughout the last year. I continue to be inspired by both the caliber of their research and the way they conduct themselves in the academic community. I would also like to thank the rest of Professor Wierman's group for providing a welcoming and engaging community.

I owe a great deal to Professors Ralph Adolphs, Azita Emami, Mory Gharib, Victoria Kostina and Steven Low for welcoming me into their groups during my time at Caltech. My experiences within their groups were tremendously helpful for clarifying my ever-evolving interests, as well as for training me to be an effective researcher. I would additionally like to express my appreciation to Professors Changhuei Yang and Glen George for their guidance as I navigated the Electrical Engineering option.

I would like to thank Beyond Limits, and particularly Azarang Golmohammadi, for providing us with real-world data and models, as well as CAST for connecting us to these collaborators.

Finally, I would like to thank my family for their continued love and support. I could not have made it this far without their dedication and sacrifices.

Increasing renewable penetration into the power grid is critical for combating climate change. To implement this successfully, it is crucial to design real-time dispatch algorithms that are robust to the uncertainty that renewable sources present. It has proven difficult to produce effective large-scale dispatches on the fly using traditional methods; as such, this has motivated research into incorporating modern machine learning (ML) methods into economic dispatch. In order for ML-based dispatch algorithms to be effectively deployed, they must have the level of performance guarantees necessary for a safety-critical setting like the grid, and also be able to enforce strict operational constraints. In the first part of this work, we consider the problem of designing learning-augmented algorithms for online optimization in the presence of ramp and feasibility constraints, and provide some of the first results in this space to our knowledge. We use these insights to develop learning-augmented algorithms that adhere to these constraints, and demonstrate how they can effectively balance between algorithm performance and the potential for constraint violations. In the second part of this work, we consider the complementary problem of training an ML model to perform economic dispatch in the face of complex operational constraints. In particular, we utilize a plant model and historical data from a real-world co-generation plant, and develop methods to enforce constraints in our ML model. Our results demonstrate that ML models can simultaneously achieve good performance and minimize constraint violations in a real-world dispatch setting.

TABLE OF CONTENTS

Number *Page*

INTRODUCTION

1.1 Motivation

There are numerous technological and societal challenges we will need to overcome to effectively fight climate change, with one of them being the design and implementation of better energy systems. In particular, future energy systems must emit significantly fewer greenhouse gases than their present day versions, but must also be resilient to increasingly frequent disruptions from extreme weather and the like. Renewable penetration into the power grid has gone up significantly in recent decades, and is predicted to continue increasing as renewable energy technologies become more widely adopted. While this is a necessary step towards building a cleaner energy system, it also introduces significantly more uncertainty into the power grid compared to before. After all, a traditional coal-fired plant can maintain a steady output of energy as long as it has a steady supply of coal, but the energy output of renewable technologies like wind and solar can vary drastically and unpredictably throughout the day. Due to this uncertainty, grid operators increasingly have to rapidly make decisions in response to the conditions at any given time. This has motivated widespread research into improving algorithms for economic dispatch - the process by which grid operators decide how to best utilize generation resources to produce energy while minimizing cost. If we develop better dispatch algorithms for our energy resources, we can not only make more effective decisions in the face of uncertainty, but also to make energy systems more efficient as a whole.

One way researchers have developed new dispatch algorithms is through the development of online algorithms that make sequential decisions "on the fly". Online algorithms are better at handling uncertainty since they are designed to make effective decisions given incomplete or no information about the future. These algorithms often have theoretical guarantees on their performance, particularly in the "worst case". As such, they are able to meet the robustness requirements necessary for safety-critical applications like energy systems, but they also tend to be overly conservative in practice.

There is also significant research being done on developing new dispatch algorithms by using machine learning (ML). Machine learning and artificial intelligence (AI) algorithms have become increasingly popular in the last decade, as they are often able to outperform traditional approaches in

various tasks, particular when there is significant data and complexity involved. As such, machine learning appears to have significant potential when applied to energy systems. However, while machine learning algorithms commonly demonstrate strong empirical results, there are several barriers towards their deployment in energy systems. For one, energy systems often have strict operational constraints that that many "off-the-shelf" machine learning methods are not able to satisfy. This has motivated further research into neural networks that are informed by the physics and constraints of electric power grids. In addition, the safety-critical nature of energy systems requires robustness guarantees that are currently incompatible with the black-box nature of many machine learning methods. This observation has motivated research into providing robustness guarantees for ML algorithms. One method for doing this is to use learning-augmented algorithms, which are a class of meta-algorithms that combine traditional, robust algorithms with untrusted ML advice. Notably, learning-augmented algorithms can be designed to have performance guarantees with respect to both the advice and robust algorithms, and ideally get the "best of both worlds".

1.2 Related Work

We now provide a more thorough review of the literature broadly surrounding the use of ML for real-time dispatch in highly-constrained energy systems.

Metrical Task Systems

In many real-world applications, an agent must make decisions "on the fly" given incomplete or no information about the future. One well-studied method of approaching these "online" problems is to design online algorithms, particularly in the metrical task systems setting (MTS) introduced in [8]. MTS captures the essence of many real-world "online" problems, where an agent seeks to minimize some recurring cost at each timestep while also not changing between states too significantly between timesteps. Specifically, a loss function f_t is revealed to the decision maker at every timestep t; the decision maker must then choose an action x_t from an action space $\mathcal{X}$ that incurs a "hitting cost" $f_t(x_t)$ as well as a "switching cost" $d(x_t, x_{t-1})$, where d is some metric. Then, the overall loss function over some horizon $t \in \{1, \dots, T\}$ is given by $\sum_{t=1}^{T} f_t(x_t) + d(x_t, x_{t-1})$. The performance of an online algorithm is often measured by its "competitive ratio", which is the supremum over all sequences $f_1, \dots, f_T$ of the ratio between the loss incurred by the online algorithm given that sequence and the loss incurred by the optimal choices in hindsight. Any determinstic and randomized MTS algorithms are $\Omega(|\mathcal{X}|)$-competitive [8] and $\Omega(\log |\mathcal{X}|)$-competitive respectively [6, 9]. We can also define the competitive difference of an online algorithm, which considers the difference in cost using the optimal choices in hindsight in lieu of the ratio when calculating the competitive ratio.

Smoothed Online Convex Optimization

In the SOCO problem, at every timestep t, an agent makes a decision x_t in a convex action space $\mathcal{X}$, and a convex "hitting cost" f_t is revealed. The SOCO problem with α-penalized cost and lookahead i over a horizon $\{1, \ldots, T\}$ is defined to be $\mathbb{E}\left[\sum_{t=1}^{T} f_t(x_{t+i}) + \alpha\|x_{t+i} - x_{t+i-1}\|\right]$, where $\|\cdot\|$ is some seminorm that defines a "switching cost". Then, we have an MTS instance if $i = 1, \alpha = 1$ [1].

The SOCO problem was introduced by Lin et al. in [40], in which they studied geographical load balancing. They looked into how jobs could be distributed across data centers based on the proportion of renewable energy they were using at a given time, in order to better "follow the renewables". This problem lended itself well to a SOCO formulation since aside from the primary objective of using renewable energy when it is available, they also had to account for switching costs associated with turning servers on and off. The SOCO problem has proven to be useful for many real-world applications where decision-makers must make sequential decisions while also minimizing changes between timesteps. As such, SOCO has seen a diverse range of applications ranging from data centers [39, 40], electricity pricing [31], control [24] and economic dispatch [37].

Much of the initial work in SOCO relied on having access to limited predictions of the future in order to make good decisions. For instance, [40] used a length w window of perfect predictions into the future. Their algorithm, Averaging Fixed Horizon Control (AFHC), is able to achieve a competitive ratio of $1 + O(1/w)$. There has also been significant work that considers SOCO with noisy predictions of the future [13, 14, 36]. [13] introduces a colored noise model for prediction error and demonstrate that under this model, AFHC can achieve sublinear regret and a constant competitive ratio while using a length $O(1)$ prediction window. [36] introduces a gradient-biased algorithm Receding Horizon Inexact Gradient (RHIG) and analyze its dynamic regret in the presence of general prediction errors without assumptions, as well as in the prediction error model introduced in [13].

There has also been considerable work to develop effective SOCO algorithms that do not rely on predictions. Bansal et al. gave a 2-competitive SOCO algorithm in the scalar case [5], which was later shown to match the lower bound [3]. Beyond the scalar case, there is considerable difficulty in finding effective SOCO algorithms, especially since there is a $\Omega(\sqrt{d})$ lower bound for general SOCO problems in d dimensions [23]. Chen et al. introduced the Online Balanced Descent (OBD) algorithm which is able to achieve a constant competitive ratio with locally polyhedral hitting costs and ℓ_2 switching costs [12]. Moreover, if the hitting costs are m-strongly convex and the switching costs are the squared ℓ_2 norm, OBD achieves a constant competitive ratio of $3 + O(1/m)$ [24]. Goel

et al. prove a $\Omega(m^{-1/2})$ lower bound on the competitive ratio for SOCO with m-strongly convex hitting costs and squared ℓ_2 switching costs [25]. They build upon the OBD algorithm and introduce Regularized Online Balanced Descent (R-OBD), whose competitive ratio matches the lower bound up to the constant factors. Notably, R-OBD maintains both a finite, dimension-free competitive ratio and sublinear regret, which was previously shown in [1] to be impossible for general SOCO.

Online Learning

Online learning is another well-studied class of online problems [27, 47]. In this setting, the decision maker must commit actions x_t while only having access to loss functions $f_1, \ldots, f_{t-i}, i \geq 1$, critically not including f_t. Online learning algorithms are often evaluated by their "regret", which include static and dynamic regret. The static regret of an algorithm is the worst-case difference between the loss incurred by the algorithm and the loss incurred if the algorithm had played a single point x^* at every timestep. Meanwhile, the dynamic regret is the worst-case difference between the loss incurred by the algorithm and the loss incurred by the optimal choices in hindsight. There are many well-known examples of online learning problems. As an example, Zinkevich introduced the online convex optimization (OCO) problem, in which at every timestep t, an agent has access to convex loss functions $f_1, \ldots, f_{t-1}$ and must commit an action x_t; the objective is then to minimize $\sum_{t=1}^{T} f_t(x_t)$. Another example is the experts problem, in which an agent presented with n "experts". At each timestep t, the agent must commit a probability distribution $p_t \in \mathbb{R}^n, \|p_t\|_1 = 1$ over these agents, after which a loss vector $\ell_t \in \mathbb{R}^n$ over the agents is revealed. The agent's loss over a horizon $t \in \{1, \ldots, T\}$ is then $\sum_{t=1}^{T} p_t \cdot \ell_t$. The goal in the experts problem is often to minimize the regret with respect to the single best expert in hindsight; it is well-established that the multiplicative weights [41] and Hedge algorithms [22] can achieve static regret sub-linear in T for the experts problem.

Intersection of MTS and Online Learning

Over the years, there has also been significant work trying to connect MTS and online learning. For instance, Blum and Burch study the relationship between algorithms for the MTS setting and the experts setting in [7]. In particular, they introduce MTS with the α-unfair competitive ratio, in which the optimal sequence in hindsight $x_1^*, x_2^*, \ldots, x_T^*$ minimizes $\sum_{t=1}^{T} f_t(x_t) + \alpha \cdot d(x_t, x_{t-1})$. They use this metric to demonstrate that algorithms designed for MTS can be effectively applied to the experts setting and vice versa. In particular, Buchbinder et al. [10] build upon the work of Blum and Burch to give a unified algorithm for the experts and MTS problems that can achieve optimal regret bounds and competitive ratio respectively by adjusting α. They also provide bounds on regert beyond the static expert cases; for instance, they are able to analyze the "drifting experts"

case, which considers the regret compared to a sequence of experts for which the sum of changes between timesteps is bounded.

Andrew et al. consider the relationship between the OCO and MTS settings [1]. In particular, they consider the smoothed online convex optimization (SOCO) problem with lookahead which is a generalization of OCO and MTS with convex costs. With this setup, Andrew et al. show that an algorithm cannot simultaneously achieve sublinear static regret and a finite competitive ratio within this setting. While this result implies that algorithms for MTS and online learning cannot be generally compatible, it does not discount the value of combining insights from the two fields. In particular, the SOCO problem emerged by considering the intersection of MTS and online convex optimization and has proven to be both theoretically rich and practical useful.

Online Optimization with Switching, Ramp and Feasibility Constraints

In online optimization, switching costs are often used to model the costs associated with changing states. The inclusion of switching costs into the objective function acts as a "soft penalty" for changing between states too significantly. However, there is also significant interest in developing online optimization algorithms with hard limits on the extent to which they can change between states.

Badiei et al. consider online convex optimization with ramp constraints, which establish hard limits on the amount actions can change between timesteps [4]. Ramp constraints are common in many real-world systems, such as in power grids. Due to the inherent operational inertia of generation plants, there are strict limits on the amount they can ramp production up or down within a given time. As such, it is crucial to consider ramp constraints when considering economic dispatch [51, 21]. Badiei et al. consider element-wise ramp constraints given by $\underline{X}^{(i)} \leq x_t^{(i)} - x_{t-1}^{(i)} \leq \overline{X}^{(i)}$, where $x^{(i)}$ designates the i-th entry of a vector. They derive asymptotically tight bounds on the competitive difference of AFHC, and also show that AFHC achieves an asymptotically optimal competitive difference among a class of "forward looking" algorithms.

Shi et al. consider SOCO with linear hitting costs and ramp constraints [49]. They search for an optimal affine policy by posing a robust optimization problem over an uncertainty set for costs in the future in order to determine the policy parameters, in order to produce a Robust Affine Policy (RAP). Unlike previous work, their framework is also able to enforce hard feasibility constraints such as supply-demand balance constraints, which makes it more amenable for applications like power grids with many operational constraints. They also do not provide a specific competitive ratio for RAP, as its performance depends on the specifics of the uncertainty set it is optimized over.

Chen et al. consider the "switching-constrained OCO problem", in which an adversary and player take turns selecting loss functions f_t and actions x_t respectively. In particular, the player has access to f_t before committing x_t, and the total cost is $\sum_{t=1}^{T} f_t(x_t)$. In lieu of including a switching cost, they consider the case in which the player can change her action x_t less than K times. They demonstrate that the minimax regret of the switching-constrained OCO problem is $\Theta(\frac{T}{\sqrt{K}})$. Sherman and Koren consider the "lazy OCO problem", which is similar to switching-constrained OCO except the number of switches only has to be less than K in expectation [48]. They give an efficient algorithm whose regret is upper bounded by $O(\sqrt{T} + \sqrt{d}T/K)$ and $\tilde{O}(dT/K^2)$ for general convex costs and strongly convex costs respectively, where d is the dimension of the action space. They provide a matching $\Omega(T/K)$ lower bound for the general convex costs, and also obtain results for adaptive, oblivious and stochastic i.i.d. adversaries.

Online Algorithms for Energy Applications

We have established that the SOCO formulation has seen many applications to energy systems [4, 31, 37]. There has also been significant work in online algorithms beyond SOCO as a whole for energy systems applications, for which we provide a limited overview.

Xie and Ilic apply model predictive control (MPC) for the economic dispatch problem, particularly when grid operators only have access to accurate predictions within just a few minutes [53]. Wang et al. consider energy distribution in a smart grid [52]. They start by introducing a convex formulation that includes many aspects of the grid. They give both offline and online algorithms for this formulation and show that the dispatch from their online algorithm converges asymptotically to the optimal offline dispatch. Shi et al. consider online algorithms for microgrids, and claim that many previously existing algorithms failed to consider the feasibility constraints associated with real-world power-grids [50]. They develop an online energy management strategy based on Lyapunov optimization that follows realistic microgrid constraints and experimentally demonstrate that it performs well relative to the optimal offline algorithm. Zhong et al. present an online algorithm based on Lyapunov optimization for managing distributed energy storage networks [58]. Their online control approach is experimentally shown to find a near-optimal solution.

Christianson et al. consider economic dispatch with predictions with a focus on guaranteeing feasibility [16]. They consider a setting where a system operator must first set "planning variables" related to the amount of power to be generated, before committing specific dispatches in real time. They identify that Receding Horizon Control (RHC), which is commonly used in practice for online dispatch, does not guarantee feasibility even if the initial planning variable was properly set to allow for a feasible dispatch. They give an algorithm, Feasible Fixed Horizon Control (FFHC), that takes

advantage of predictions while still guaranteeing a feasible dispatch in the face of uncertainty.

Learning-Augmented Algorithms

Machine learning and artificial intelligence are increasingly replacing traditional approaches in various tasks, including in energy applications. In recent years, there has also been significant work in developing machine learning based approaches to various problems in energy, with applications in DC and AC optimal power flow [15, 57, 29, 28, 43, 20, 56], grid control [11, 18] and economic dispatch [38, 54, 26]. Many of these machine learning based approaches have demonstrated strong empirical performance, as well as significantly faster runtime compared to traditional online algorithms, especially ones that have to perform expensive optimizations. However, while machine learning often presents superior empirical performance on average, one well-known shortfall is that they often lack bounds on their worst-case performance. Traditional algorithms encounter the opposite scenario, where there is much more existing theory on their worst-case performance rather than their average-case performance. Specifically, an algorithm's worst-case performance can be characterized by metrics like its worst-case runtime complexity or its worst-case competitive ratio. This has motivated research into learning-augmented algorithms, or more generally algorithms with predictions. These algorithms combine a traditional, robust algorithm with a separate black-box predictor or advice algorithm to generate a meta-algorithm whose performance captures "the best of both worlds".

Learning-augmented algorithms are often best suited for online problems like in the MTS setting, as the algorithm can continually adjust its parameters based on how the component algorithms are performing. Online algorithms are commonly characterized by their competitive ratio, which gives the ratio of the total loss it incurs to the loss incurred by the optimal choices in hindsight. Furthermore, a learning-augmented algorithm with an advice algorithm and a robust algorithm is commonly characterized by its *consistency* and its *robustness* [44]. Consistency is defined as the competitive ratio of the learning-augmented algorithm with the advice algorithm when the untrusted advice algorithm has zero error; ideally, if the advice algorithm is performing well, then the learning-augmented algorithm should recognize this and match its actions to that of the advice algorithm. Robustness is the competitive ratio with regards to the robust algorithm when the advice algorithm's error goes to infinity. These definitions capture the general design goals of learning-augmented algorithms: performing roughly as well as the black-box advice algorithm when the advice algorithm is performing well, but having the traditional, robust algorithm as a fallback when the advice algorithm is performing poorly. In recent years, significant progress has been made in the design of learning-augmented algorithms for many problem classes, such as ski-rental, caching

and scheduling [44, 42, 19, 30, 45, 32, 46, 2]. In addition, there is also a growing body of work in applying learning-augmented algorithms specifically in energy-systems applications [17, 34, 33, 35].

A learning-augmented algorithm has consistency at least 1, since it is the competitive ratio when the advice algorithm has zero error. So, a learning-augmented algorithm is near optimal in some sense if its consistency is some $(1+\epsilon)$, $\epsilon > 0$, since it can get arbitrarily close to the performance of its advice algorithm. However, a lower consistency ratio has the tradeoff that the robustness ratio will increase as a result.

1.3 Our Contributions

In the first part of this work, we investigate learning-augmented algorithms that obey both ramp and feasibility constraints. We examine fundamental limits on algorithm performance and constraint violations for a general class of ramp-constrained learning-augmented algorithms. We also propose an algorithm and present theoretical and experimental results to demonstrate that we can take advantage of ML advice while maintaining strong worst-case guarantees relative to standard algorithms. In particular, our work contributes to the limited literature on online optimization with hard ramp and feasibility constraints, and is also one of the first, to our knowledge, to consider how learning-augmented algorithms can be designed to enforce difficult real-world constraints.

In the second part of this work, we consider the complementary question of training a machine learning algorithm to enforce constraints in a real-world co-generation setting. We demonstrate that with some small modifications, even a simple ML model can be trained to produce high-quality dispatch decisions in this setting. In particular, we demonstrate that a ML model can learn to not only effectively minimize violations of complex operational constraints, but also meet a simpler class of operational constraints exactly.

Taken as a whole, our work can be extended to produce a full stack for reliable ML-based algorithms for optimization in energy systems that have both strong performance guarantees for cost as well as strong guarantees on the magnitude of constraint violations they are susceptible to incurring.

1.4 Notation

Let $[n] := \{1, \dots, n\}$ for $n \in \mathbb{N}$. $\overline{\mathbb{R}}_+$ denotes the nonnegative extended reals. $\|\cdot\|_p$ denotes the p-norm. $x^{(i)}$ denotes the i-th entry of some vector x. For two vectors $x_1, x_2 \in \mathbb{R}^n$, we take $x_1 \leq x_2$ to mean that $x_1^{(i)} \leq x_2^{(i)}$ for all $i \in [n]$; the same holds for $x_1 \geq x_2$. For an algorithm ALG, we take ALG_t to be the decision it makes at time t.

LEARNING-AUGMENTED ALGORITHMS WITH RAMP AND FEASIBILITY CONSTRAINTS

The first half of this chapter considers how we can exploit good performance of ML advice while limiting the proportion of constraint violations we incur if the ML advice misbehaves and violates operational constraints. We demonstrate that while this is difficult in the general case, we can use a small amount of resource augmentation to drastically reduce the proportion of total constraint violation incurred in the case where ML advice would incur especially large constraint violations. In the second half, we consider how we can maintain good performance with respect to both of our component algorithms. We give an example of a worst-case in which we are hampered by the presence of ramp constraints, and give an algorithm specifically designed to handle this case. We give theoretical and experimental results that demonstrate the ability of our algorithm to take advantage of ML advice while maintaining strong worst-case guarantees.

2.1 Model and Preliminaries

We consider online convex optimization over a horizon of length T. At every timestep $t \in [T]$, a cost function $f_t \colon \mathbb{R}^n \to \overline{\mathbb{R}}_+$, a constraint function $g_t \colon \mathbb{R}^n \to \overline{\mathbb{R}}_+$ and a constraint $s_t \in \overline{\mathbb{R}}_+$ are revealed to the player, who must then commit a decision $x_t \in \mathcal{X} \subset \mathbb{R}^n$ that enforces $g_t(x_t) = s_t$. Let $\underline{d}, \overline{d} \in \mathbb{R}^n$ be the lower and upper ramp limits. Let the switching cost be given by $\|x_t - x_{t-1}\|$. Then, the objective is the following:

$$\begin{aligned} \min_{\{x_t\}_{t=1}^T} \quad & \sum_{t\in[T]} f_t(x_t) + \|x_t - x_{t-1}\| & \\ \text{s.t.} \quad & \underline{d} \le x_t - x_{t-1} \le \overline{d} & \forall t \\ & g(x_t) = s_t & \forall t \end{aligned} \tag{2.1}$$

Application to Energy Dispatch

In this work, we are particularly interested in the energy dispatch setting. Then, x_t represents some dispatch decision, and $f_t(x_t)$ is the cost associated with that decision. We limit $\mathcal{X}$ to have finite diameter D, where $D = \sup_{x_1,x_2\in\mathcal{X}} \|x_2 - x_1\|$. $g_t(x_t)$ represents the amount of power produced by a dispatch decision x_t, while s_t represents the power demand that the system has to meet. Moreover, if we take the entries of x_t to simply be the power produced by a set of generators,

then $g_t(x_t)$ is a constant function $g(x_t) = \mathbf{1}^T x_t$. We assume that f_t is a constant, affine function $f(x) = a^T x + b$, $a \in \overline{R}_+^n, b \in \overline{R}_+$. We also assume that a does not have all entries equal, or else the cost will be directly proportional to the power produced and the problem becomes trivial. We assume the ramp constraints are $\underline{d} = -r\mathbf{1}^n$ and $\overline{d} = r\mathbf{1}^n$, where $r \in \overline{\mathbb{R}}_+$. Since the ramp constraints are already significantly limiting the amount we move between timesteps, we ignore the switching costs $\|x_t - x_{t-1}\|$. Then, our objective is the following:

$$\begin{aligned} \min_{\{x_t\}_{t=1}^T} \quad & \sum_{t\in[T]} f(x_t) \\ \text{s.t.} \quad & -r\mathbf{1}^n \le x_t - x_{t-1} \le r\mathbf{1}^n \quad \forall t \\ & \mathbf{1}^T x_t = s_t \qquad\qquad\qquad\quad \forall t \end{aligned} \tag{2.2}$$

In addition, since x_t represents the power produced by a set of generators, there are also limits on the feasible values of x_t, which we refer to as the system feasibility constraints. We wish to enforce that $\underline{X} \le x \le \overline{X}$, where $\underline{X}, \overline{X} \in \overline{\mathbf{R}}_+^n$. Let $\mathcal{X} \triangleq \{x \mid \underline{X} \le x \le \overline{X}\}$ denote the feasible space that represents the true capabilities of our generators. Then, our true system dispatch is $proj_{\mathcal{X}}(x_t)$, and the system must meet the rest of the power demand s_t using reserve resources. We take $\|x_t - proj_{\mathcal{X}}(x_t)\|_{\mathbf{1}}$ to represent how much we violate the system operation constraints. This is directly associated with a cost, as if $x_t \notin \mathcal{X}$, our system is producing either too much or too little power, and must use external resources to account for $\|x_t - proj_{\mathcal{X}}(x_t)\|_{\mathbf{1}}$ units of power. Our system then has the additional objective to minimize the "total constraint violation" given by $\sum_t \|x_t - proj_{\mathcal{X}}(x_t)\|_{\mathbf{1}}$.

Learning-Augmented Algorithms Setup and Objectives

In order to form our learning-augmented algorithm ALG, we have algorithms ADV and ROB, which are the untrusted advice algorithm and trusted robust algorithm respectively. We assume that ADV, ALG, and ROB start from the same dispatch, so the space of possible dispatches that respect the ramp constraints are the same for ADV_1, ALG_1, and ROB_1. Let the learning-augmented algorithm be given by ALG. We assume that ALG is always a convex combination of ADV and ROB. Denote λ_t as the convex parameter that ALG plays, so $\mathsf{ALG}_t = \lambda_t \mathsf{ADV}_t + (1-\lambda_t)\mathsf{ROB}_t$. Both of these algorithms always follow the ramp constraint $x_t - x_{t-1} \in [-r, r]^n$ and supply-demand balance constraint $g(x_t) = s_t$. We also assume that the system allows resource augmentation such that the ramp constraint for ALG is $[-r(1+\epsilon'), r(1+\epsilon')]$, where $\epsilon \ge 0$. This guarantees that ALG can always "move between" ADV and ROB by at least some small amount. ROB always follows the system feasibility constraints, but it is possible for ADV to violate the constraints and incur constraint violation costs. Note that one consequence of this assumption is that $\|s_t - s_{t-1}\| \le n \cdot r$, since it

must be possible for our algorithms to meet the next power demand within the ramp constraints.

We are interested in the following general-horizon tradeoffs:

- The tradeoff between the competitive ratio with respect to ADV assuming that ADV never violates the system feasibility constraints, and the proportion of total constraint violation that ALG incurs compared to ADV if ADV does violate the system feasibility constraints.
- The tradeoff between the competitive difference with respect to ADV and ROB, assuming that ADV never violates the system feasibility constraints costs

We assume that our algorithms are always residing in a "demand hyperplane" given by $\{x|\mathbf{1}^T x = s_t\}$. The reason that we let algorithms play "infeasible" dispatches $x_t \notin \mathcal{X}$ is so that they can stay on the demand hyperplane. If ADV and ROB are both on the demand hyperplane, then we can focus our attention on finding the best convex combination parameter λ_t at any given time, without being concerned that the convex combination is potentially not meeting the power demand s_t.

In particular, we are motivated by the setup shown in Figure 2.1. At times t and $t+1$, all the algorithms are playing dispatches that lie on the corresponding power demand hyperplane. Suppose ADV is in the top left corner since it is naively minimizing its cost, while ROB is playing a more conservative dispatch. Then, it is unable to meet the next power demand at time $t+1$ without calling on external resources, as ADV_{t+1} causes a system constraint violation. However, if we play a sufficiently small λ_t, then ALG can avoid violating the system constraints while still meeting the power and ramp constraints.

In addition, one of the questions we are interested in is how to best adjust λ in an online manner. Without resource augmentation, if the power demand changes as much as possible, then ALG is also forced to ramp up or down as much as possible, and λ does not change. By allowing for a bit of resource augmentation, it gives us the freedom to always change λ by some minimum amount between timesteps, which proves to be quite useful. For instance, without resource augmentation, we could consider some adversarial setting in which the power demand changes up or down as much as possible between each timestep, and we cannot change λ at all. We relate ϵ with the amount we can always change λ in the following lemma.

Lemma 1. *Let D be the largest distance in a single dimension between two points in the feasible set (for instance, if the feasible set is hypercube with sides of length D). Let λ by the convex parameter that* ALG *plays between* ADV *and* ROB*. We are guaranteed to be able to change λ by $\pm r\epsilon/D$ at each timestep.*

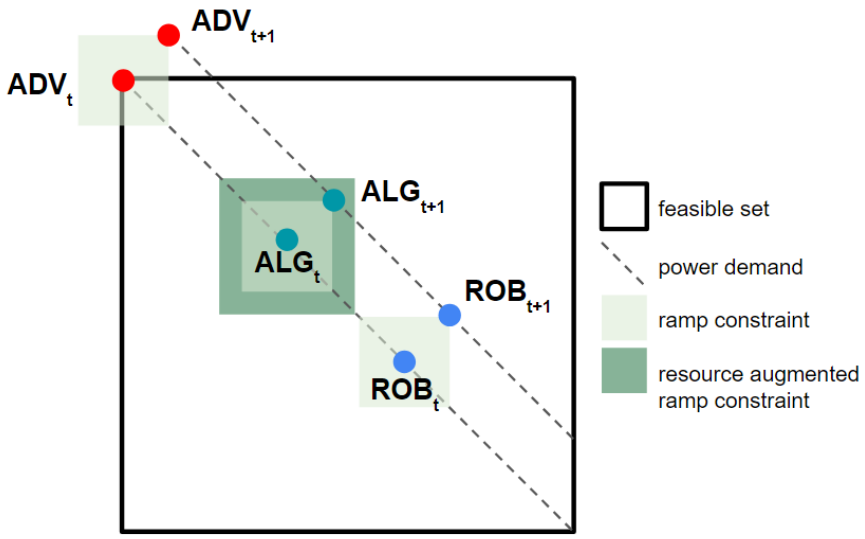

Figure 2.1: Insightful example for algorithm dispatch decisions and power demand

Proof of Lemma 1. Let $\mathcal{R} = \{x \mid -r\mathbf{1} \leq x \leq r\mathbf{1}\}^n$ be a "ramp box", which represents how much ADV_t and ROB_t can change between timesteps. Let $\mathcal{R}_\epsilon = \{x \mid -r(1+\epsilon)\mathbf{1} \leq x \leq r(1+\epsilon)\mathbf{1}\}$ represent the resource-augmented ramp box. Consider ADV_t, ROB_t and $\mathsf{ALG}_t = \lambda_t \mathsf{ADV}_t + (1-\lambda_t)\mathsf{ROB}_t$. Let $\mathsf{ADV}_{t+1} = x_t^{\mathsf{A}} + \mathsf{ADV}_t$ and $\mathsf{ROB}_{t+1} = x_t^{\mathsf{R}} + \mathsf{ROB}_t$, with $x_t^{\mathsf{A}}, x_t^{\mathsf{R}} \in \mathcal{R}$. We can still play $\lambda_{t+1} = \lambda_t$ for ALG_t, as

$$\begin{aligned}\lambda_t \mathsf{ADV}_{t+1} + (1-\lambda_t)\mathsf{ROB}_{t+1} &= \lambda_t x_t^{\mathsf{A}} + (1-\lambda_t)x_t^{\mathsf{R}} + \lambda_t \mathsf{ADV}_t + (1-\lambda_t)\mathsf{ROB}_t \\ &= \lambda_t x_t^{\mathsf{A}} + (1-\lambda_t)x_t^{\mathsf{R}} + \mathsf{ALG}_t\end{aligned}$$

By convexity, $\lambda_t x_t^{\mathsf{A}} + (1-\lambda_t)x_t^{\mathsf{R}} \in \mathcal{R}$. Since $\mathcal{R}_\epsilon$ is larger than $\mathcal{R}$ by $\pm r\epsilon$ in each vector entry, we know that we are guaranteed to be able to further change ALG_{t+1} by $\pm r\epsilon$ in each vector entry. The largest distance in a single dimension between two points in the feasible set is at most D, so we are guaranteed to be able to change λ by $\pm r\epsilon/D$ at each timestep. □

2.2 Tradeoff Between Competitive Ratio With Respect to Advice and Proportion of Constraint Violation Incurred

In this section, we analyze the tradeoff between how well a learning-augmented algorithm can follow ML advice when it performs well versus the risk it is exposed to when the advice misbehaves. In particular, we show that while it is difficult to meaningfully shield the algorithm when the advice incurs small constraint violations, we can get much stronger guarantees against large constraint violations.

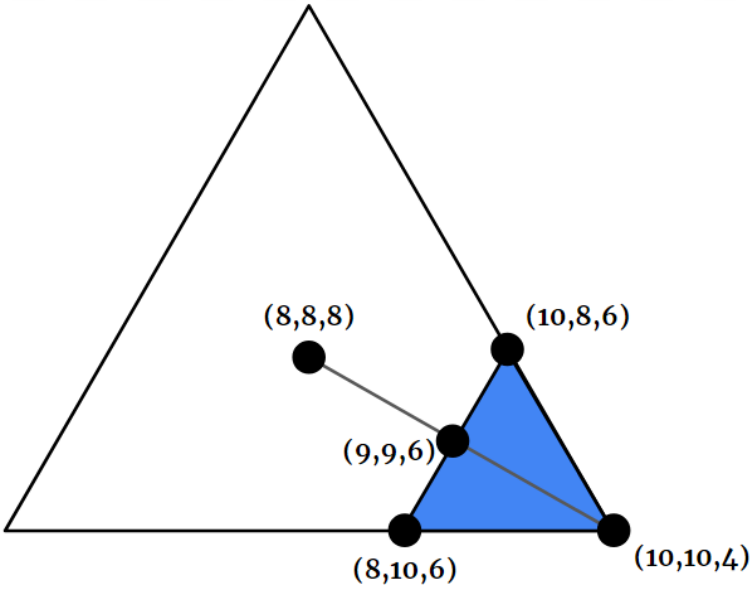

Figure 2.2: Space of dispatches that meet demand of 24 while staying feasible

General Lower Bound on Proportion of Total Constraint Violation Incurred

We denote $C^{\mathsf{A}} = \sum_{t\in[T]} f(\mathsf{ADV}_t)$, $C^{\mathsf{R}} = \sum_{t\in[T]} f(\mathsf{ROB}_t)$ and $C = \sum_{t\in[T]} f(\mathsf{ALG}_t)$

Proposition 1. *Suppose* ALG *has the property that* $C \leq (1-\alpha)C^{\mathsf{R}} + \alpha C^{\mathsf{A}}$ *whenever* ADV *is always feasible. Then, there exists a sequence of power demands and dispatches in which* ALG *must incur at least* α *of the total constraint violation of* ADV *without resource augmentation.*

Proof of Proposition 1. We give an example sequence of power demands and dispatches in which ADV remains feasible, and show that ALG must be playing a certain dispatch in order to guarantee that $C \leq (1-\alpha)C^{\mathsf{R}} + \alpha C^{\mathsf{A}}$. We then show that we can then extend this sequence to force ALG to incur at least α of the (nonzero) infeasibility cost of ADV.

Let $c = [1,1,5], \underline{X} = [0,0,0], \overline{X} = [10,10,10], r = 2, \alpha = 0.5, \epsilon = 0$. Suppose that the demand goes to 24, ADV plays $[10,10,4]$ and ROB plays $[8,8,8]$ for arbitrarily long. For this set of dispatches and demands, ADV is always feasible. We can consider the 2D space of dispatches that meet the demand of 24 while staying feasible, which is depicted in Figure 2.2.

At $[9,9,6]$, the cost per timestep is 48. The intersection of the hyperplane for which cost is 48 with the hyperplane for which supply is 24 is the line from $[8,10,6]$ to $[10,8,6]$. Any algorithm that achieves $C \leq (1-\alpha)C^{\mathsf{R}} + \alpha C^{\mathsf{A}}$ must eventually play a dispatch with cost 24 or less, which is represented with the shaded region. Now, once ALG plays a dispatch in this region, we consider

what happens when demand increases by 6. In order to meet this new demand, we must increase the dispatch on each generator by 2. ROB becomes $[10, 10, 10]$ and remains feasible, while ADV becomes $[12, 12, 6]$, incurring a constraint violation of 4. We now consider the constraint violation that ALG must have incurred.

We can parameterize this area as $[8 + x + y, 10 - x, 6 - y]$ for $x \in [0, 2], y \geq 0$. Once demand increases by 6, it becomes $[10 + x + y, 12 - x, 8 - y]$. Then, the constraint violation is $\max\{10 + x + y, 0\} + \max\{12 - x - 10, 0\} = \max\{x + y, 0\} + \max\{2 - x, 0\} = 2 + y \geq 2$. So, we must incur a constraint violation of at least $2 = \alpha \cdot 4$. □

This implies that if we wish to incur α of the "cost benefit" of ADV with ALG, we can be forced to also incur α of the constraint violation of ADV. In particular, our example illustrates that it is insightful to analyze what happens when a constraint violation occurs while ADV is playing $\lambda = \alpha$.

Algorithm To Control Proportion of Total Constraint Violation Incurred

We now introduce Algorithm 1, which has a tunable parameter α that makes it possible to trade off between the competitive ratio with respect to ADV and the proportion of constraint violation potentially incurred.

Algorithm 1: Ramp-Constrained Learning-Augmented Algorithm

```
for t ∈ [T] do
    if f(ROB_t) ≤ f(ADV_t) then
        ALG_t steps towards ROB_t as much as possible within ramp constraints
    end
    else // f(ROB_t) > f(ADV_t)
        if ADV_n ∈ X ∀n ∈ [t] then
            ALG_t steps towards ROB_t as much as possible within ramp constraints
        end
        else
            ALG_t steps towards ADV_t as much as possible within ramp constraints, while
              enforcing λ ≤ α
        end
    end
end
```

Competitive Ratio of Algorithm 1 with Respect to Component Algorithms

We now provide bounds on the cost incurred Algorithm 1 compared to ADV and ROB. In particular, we show that we get much stronger guarantees when $n = 2$ compared to when $n > 2$.

Proposition 2. *Algorithm 1 achieves* $C \leq \min\{\alpha C^{\mathsf{R}} + (1-\alpha)C^{\mathsf{A}}, C^{\mathsf{R}}\}$ *when* $n = 2$.

Proof of Proposition 2. Consider the algorithm ADV_α, which always just plays a convex combination of ROB and ADV with $\lambda = \alpha$. We know that we can always keep the same λ while following the ramp constraints. Let the total cost be C_α. Trivially, $C_\alpha = \alpha C^{\mathsf{R}} + (1-\alpha)C^{\mathsf{A}}$. Now, we must show that $C \leq C_\alpha$. To do, we use proof by induction to show that at every timestep, ALG incurs cost no more than ADV_α.

For the first timestep, there are two cases: ADV is lower cost than ROB, or not. If ADV is lower cost, we know that since ROB_1 and ADV_1 are within the ramp constraints from the starting point, ALG must be able to set $\lambda = \alpha$ within the ramp constraints. If ROB has the same or lower cost, ALG will have $\lambda = 0$ and incur no more cost than ADV_α.

Next, we perform the inductive step. There are three cases we have to consider based on whether ADV was better at the previous and current timesteps

- Case 1 - ADV is worse at the current timestep. Our algorithm must play some $\lambda \in [0, \alpha]$, so we do no worse than ADV_α.
- Case 2 - ADV is better at the current timestep, and was better at the previous timestep. By our inductive assumption, in the previous step, we must have had $\lambda = \alpha$. Then, we can stay there.
- Case 3 - ADV is better at the current timestep, but was worse at the previous timestep.

Suppose the losses look like $f(x) = c^T x$ (ignore constant term). Let the a_t and r_t be the decisions of the advice and robust algorithms at time t. We know the following:

- $\mathbf{1}^T a_t = \mathbf{1}^T r_t$
- $\mathbf{1}^T a_{t-1} = \mathbf{1}^T r_{t-1}$
- $a_t - a_{t-1} \in [-r\mathbf{1}, r\mathbf{1}]$
- $r_t - r_{t-1} \in [-r\mathbf{1}, r\mathbf{1}]$

We wish to show that if $c^T r_t - c^T a_t > 0$ but $c^T r_{t-1} - c^T a_{t-1} < 0$, then $r_{t-1} - a_t \in [-r\mathbf{1}, r\mathbf{1}]$. The conditions on the cost differences means that the robust algorithm performs better at time $t-1$, but

the advice algorithm performs better at time t. We show that we must be able to jump from robust at $t-1$ to advice at t - this also implies that we jump from any convex combination of advice and robust at $t-1$ to any other convex combination at t.

There must exist some unique λ such that $\lambda(c^T r_t - c^T a_t) + (1-\lambda)(c^T r_{t-1} - c^T a_{t-1}) = 0$. This implies that $c^T(\lambda r_t + (1-\lambda) r_{t-1}) = c^T(\lambda a_t + (1-\lambda) a_{t-1})$. We also know that $\mathbf{1}^T(\lambda a_t + (1-\lambda) a_{t-1}) = \mathbf{1}^T(\lambda r_t + (1-\lambda) r_{t-1})$. If $\mathbf{1}$ and c are linearly dependent, we note the original problem is trivial (cost solely a function of demand). Otherwise, if they are linearly independent, we must have that $\lambda a_t + (1-\lambda) a_{t-1} = \lambda r_t + (1-\lambda) r_{t-1} = x^*$.

$$
\begin{aligned}
x^* - a_t &\in \lambda[-r\mathbf{1}, r\mathbf{1}] \\
x^* - r_{t-1} &\in (1-\lambda)[-r\mathbf{1}, r\mathbf{1}] \\
r_{t-1} - a_t &\in [-r\mathbf{1}, r\mathbf{1}]
\end{aligned}
$$

Our algorithm will be able to jump to $\lambda = \alpha$, so case 3 also holds. Since ALG never incurs more cost than ADV_α across all timesteps, it must hold that $C \leq \alpha C^{\mathsf{R}} + (1-\alpha) C^{\mathsf{A}}$.

We can use the same argument to conclude that at every timestep, ALG never incurs more cost than ROB, so $C \leq C^{\mathsf{R}}$. So, $C \leq \min\{\alpha C^{\mathsf{R}} + (1-\alpha) C^{\mathsf{A}}, C^{\mathsf{R}}\}$ □

Our proof in 2D relied critically on the observation that when the costs of ADV goes from being higher to lower than the cost of ROB (or vice versa), their dispatches must be close to each other. However, this is generally untrue for 3D and beyond. Then, we can consider a worst-case in which ROB starts very slightly better than ADV, causing Algorithm 1 to play $\lambda = 0$. Suppose that ADV starts rapidly getting better, while ROB rapidly gets worse. It may take several steps for ALG to go from ROB to ADV_α, and thereby do much worse than ADV_α.

Proposition 3. *For $n \geq 3$, our algorithm can be forced to incur a competitive difference relative to* ROB *of order $\Omega(\alpha^4 \epsilon^{-2})$.*

Proof of Proposition 3. Suppose that ADV and ROB start at the same point, but their dispatches go to opposite points in the feasible space. The cost of ADV is always greater than ROB by some $\delta << 1$. Then, Algorithm 1 must stay at $\lambda = \alpha$.

Then, suppose that ROB starts going in the direction of maximum descent for the cost, while ADV_α does the opposite. First, we must calculate that amount that the cost difference between ROB and ADV_α can change between each timestep. The cost difference at time t is equal to $a^T(\mathsf{ADV}_{\alpha,t} - \mathsf{ROB}_t) = \alpha a^T(\mathsf{ADV}_\mathsf{t} - \mathsf{ROB}_\mathsf{t})$. Since ROB and ADV are both ramp constrained, $a^T(\mathsf{ADV}_\mathsf{t} - \mathsf{ROB}_\mathsf{t})$ can increase by at most $\overline{c} \triangleq 2rn\|a\|_1$ if each generator ramps fully up or down based on the signs of each entry of a. So, the cost difference between ADV_α and ROB can change by at most $\alpha\overline{c}$ between timesteps. In the most constrained case, ALG can change λ by $\Delta\lambda = \frac{d\epsilon}{D}$ each time. Suppose it takes $N = \alpha\frac{D}{d\epsilon}$ steps to go from ADV_α to ROB. In each of these steps $i = 1, \ldots, N$, the cost difference between ADV_α and ROB is $i\alpha\overline{c} - \delta$, and ALG incurs $\frac{N-i}{N}$ of the cost difference relative to ROB. The total "extra loss" that ALG incurs relative to ROB that we incur is then:

$$\sum_{i=1}^{N} \frac{N-i}{N} \cdot (i\alpha\overline{c} - \delta)$$

We can ignore δ, so this expression grows as $\Omega(N^2\alpha\overline{c}) = \Omega(\alpha^3\epsilon^{-2})$. □

If we use a similar example as in Proposition 3, we can see that we can incur a large competitive difference with respect to both ROB and ADV_α. In particular, if we repeat the example cyclically, Algorithm 1 get unbounded competitive difference with one of the component algorithms. This is suboptimal, as we demonstrate in Section 2.3. However, it is still reasonable to analyze Algorithm 1 for insights into the proportion of total constraint violation incurred by a ramp-constrained learning-augmented algorithm.

Upper Bounds on Proportion of Constraint Violation Incurred By Algorithm 1

Trivially, Algorithm 1 never achieves more than α of the constraint violation of ADV since $\lambda_t \leq \alpha \ \forall t \in [T]$. We can also construct a sequence of demands and dispatches to force Algorithm 1 to incur at least α of the constraint violation of ADV with the example in Figure 2.3. In this example, we have $n = 2$ generators, $\underline{X} = 0 \cdot \mathbf{1}^n$, $\overline{X} = 10 \cdot \mathbf{1}^n$ and $r = 2$. At time t, $\mathsf{ROB}_t = [8, 8]$ and $\mathsf{ADV}_t = [10, 6]$. The demand then increases maximally from 16 to 20, and $\mathsf{ROB}_{t+1} = [10, 10]$ and $\mathsf{ADV}_{t+1} = [12, 8]$. Then, note that λ cannot change between t and $t + 1$, and ALG incurs λ of the constraint violation of ADV. We can imagine that this is the first time that ADV incurred a constraint violation, and t is sufficiently large that $\lambda = \alpha$. Then, ALG must have incurred α of the constraint violation of ADV.

While this seems quite pessimistic, we show that it possible for a learning-augmented algorithm ADV to get up to α of the performance benefit of ADV while incurring less than α of constraint violations if we consider resource augmentation and the magnitude of constraint violations. To do

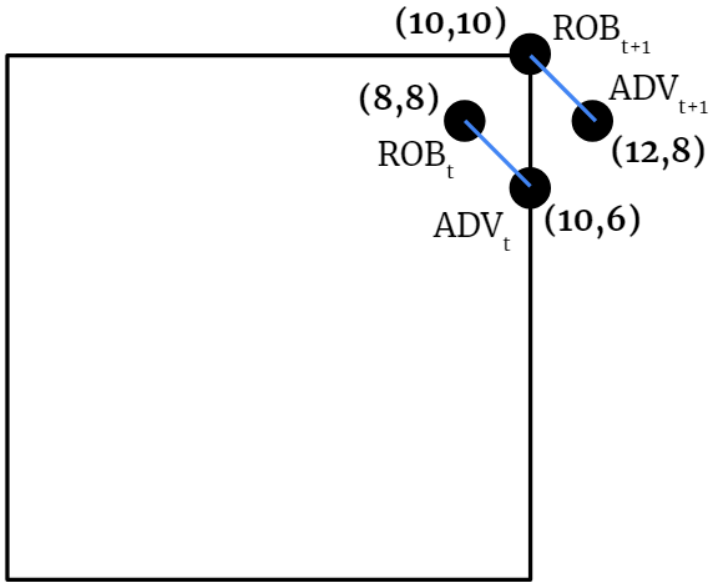

Figure 2.3: Numerical example of "worst case" for Algorithm 1 with regards to constraint violations.

so, we must consider what happens when ADV starts incurring constraint violations while ALG is playing some λ, and ALG responds by lowering λ as much as possible within the ramp and demand constraints for all future timesteps until $\lambda = 0$. Intuitively, ALG can be forced to incur a larger proportion of the total constraint violation is demand increases or decreases maximally; if demand does not change substantially, ALG can decrease λ much more and incur a significantly smaller proportion of the total constraint violation - this is illustrated in Figure 2.4. In the example on the left, demand increases enough to max out the ramp constraints for ADV and ROB. As a result, even though ALG has resource augmentation, it still cannot decrease λ by much. On the other hand, in the example on the right, demand does not increase by as much, as both ALG and ADV take the opportunity to play a dispatch with as low a constraint violation as possible. Thanks to the geometry, ALG is not only able to incur a much smaller constraint violation for time $t + 1$, but also lower λ significantly for all future trajectories, which also lowers the proportion of total constraint violation it incurs. Based on this intuition, we arrive at the following conjecture:

Conjecture 1. *In order to maximize the proportion of total constraint violation that Algorithm 1 incurs relative to* ADV*, the demand needs to increase and decrease as much as possible to force full ramps up and down.*

Assuming Conjecture 1, we can derive upper bounds on the proportion of total constraint violation that Algorithm 1 can incur. In particular, we demonstrate a tradeoff between the maximum

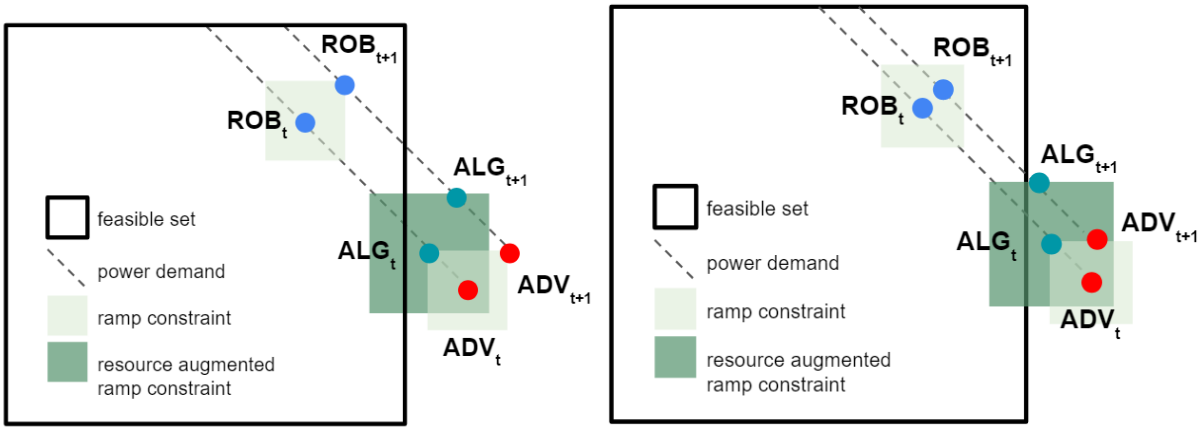

Figure 2.4: Two scenarios for demand and dispatches. Demand does not increase maximally in the example on the right.

proportion of total constant violation incurred trades off and the maximum constraint violation by ADV at any individual instance. We first consider the case without resource augmentation. Let the maximum constraint violation by ADV at any single instance be V. Assume that the ramp constraints are normalized such that when demand increases up or down as much as possible, the constraint violation of ADV changes by 1. (Note that this depends on exactly how many generators are at infeasible setpoints.) The worst case under Conjecture 1 must correspond to demand increasing by 1 until ADV has a constraint violation of V, and then indefinitely decreasing then increasing by 1. ALG incurs the greatest proportion of constraint violation if ROB is on the edge of $\mathcal{X}$ when ADV incurs a constraint violation of V. In the limit, ADV alternates between constraint violations of V and $\max\{0, V-1\}$, while ROB alternates between constraint violations of 0 and -1 (abusing notation). Then, ALG alternates between constraint violations of αV and $\max\{0, \alpha(V-1) - (1-\alpha)\}$. So, the proportion of total constraint violation converges to $\frac{\alpha V + \max\{0, \alpha(V-1)-(1-\alpha)\}}{V + \max\{0, V-1\}}$. These are plotted in Figure 2.5. From these plots, we see that for $\alpha < 1$, the proportion starts at α, dips down when $1 \leq V \leq 1/\alpha$, then goes back up towards α for $V \geq 1/\alpha$. This implies that without resource augmentation, we cannot do much better than incurring α of the total constraint violation, both when the maximum constraint violation is especially small or large.

Fortunately, we get more optimistic results when considering resource augmentation. We consider the same setting as before, but ALG can now decrease λ by $d\epsilon/D$ at each timestep. The results are shown in Figures 2.6 and 2.7. As the magnitude of the maximum constraint violation increases, when we have even small resource augmentation, the ratio converges towards zero instead of upwards to α.

In Figure 2.8 shows the maximum proportion of total constraint violation with resource augmentation for each value of V. We see that with resource augmentation, the maximum proportion eventually decreases to 0 as the maximum violation per timestep increases. ALG will return to $\lambda = 0$ within $\alpha/\frac{d\epsilon}{D}$ steps, so if the maximum violation exceeds $\alpha/\frac{d\epsilon}{D}$, ALG will not incur any constraint violation. We also see that the plots are not smooth, particular around integer values of V; to see why this is reasonable, we can consider the examples $V = 2$ and $V = 2.01$. Then, ADV can attain a constraint violation of 2 within two steps, but it would take at least three steps to reach 2.01. So, $V = 2$ and $V = 2.01$ are effectively separate cases. Since the ratio can vary so much even when the parameters change a little, it explains why the plots are neither smooth nor monotonic. From these plots, we can see that the Algorithm 1 is much more effective at reducing the proportion of total constraint violation that it incurs when ADV has large violations.

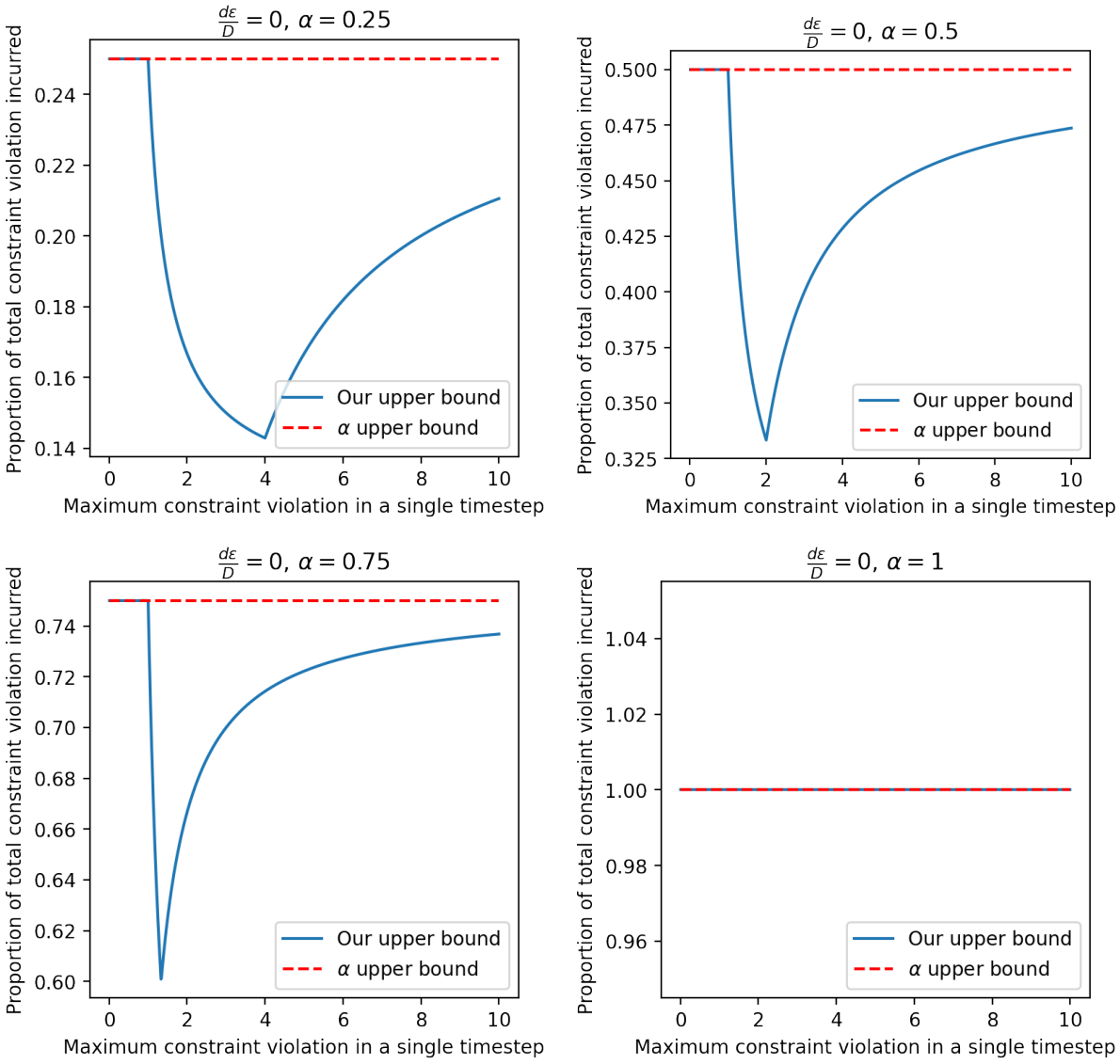

Figure 2.5: Plots of upper bound on proportion of total constraint violation incurred without resource augmentation

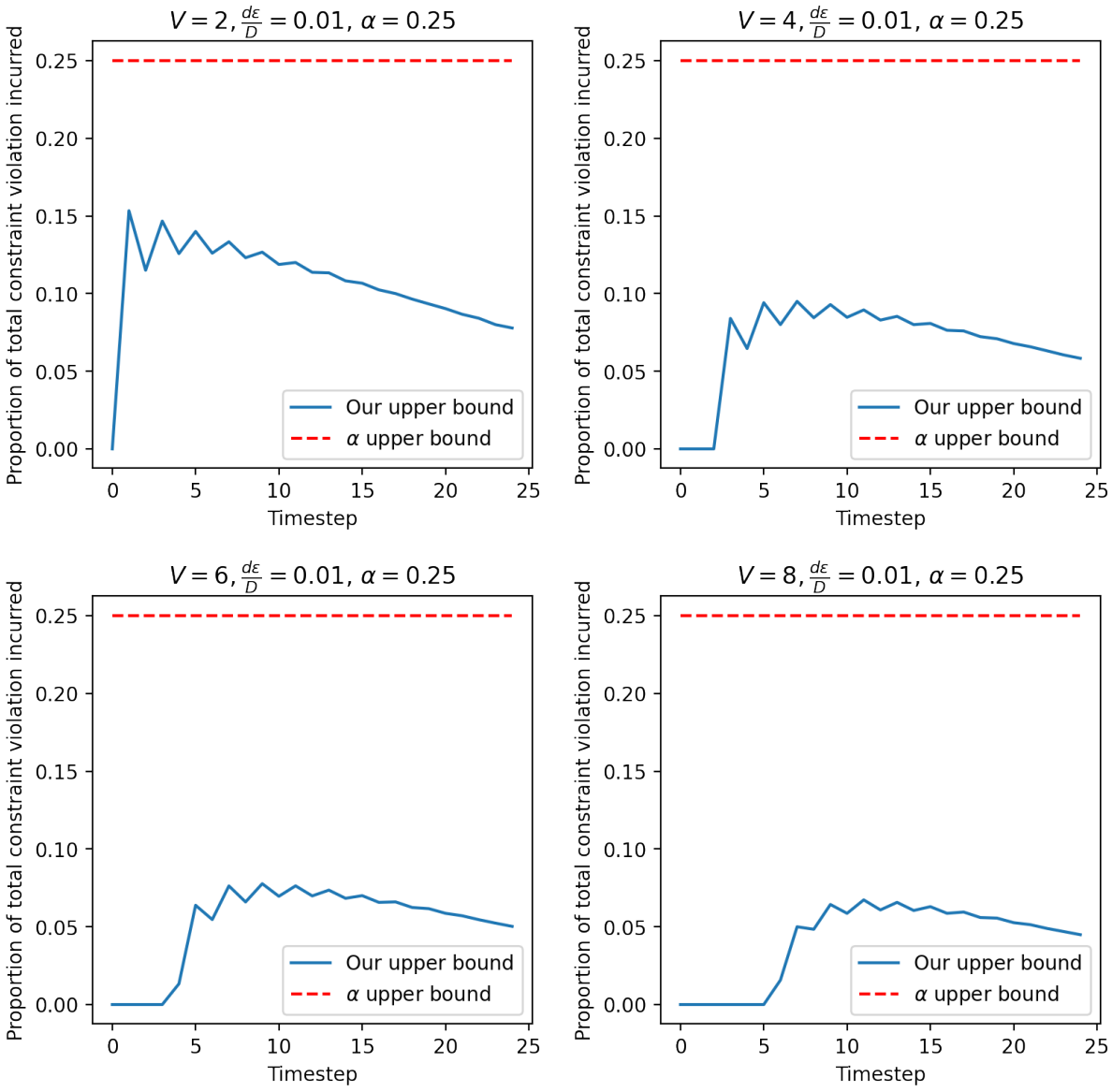

Figure 2.6: Evolution of ratio of total constraint violation, $\frac{d\epsilon}{D} = 0.01, \alpha = 0.25$

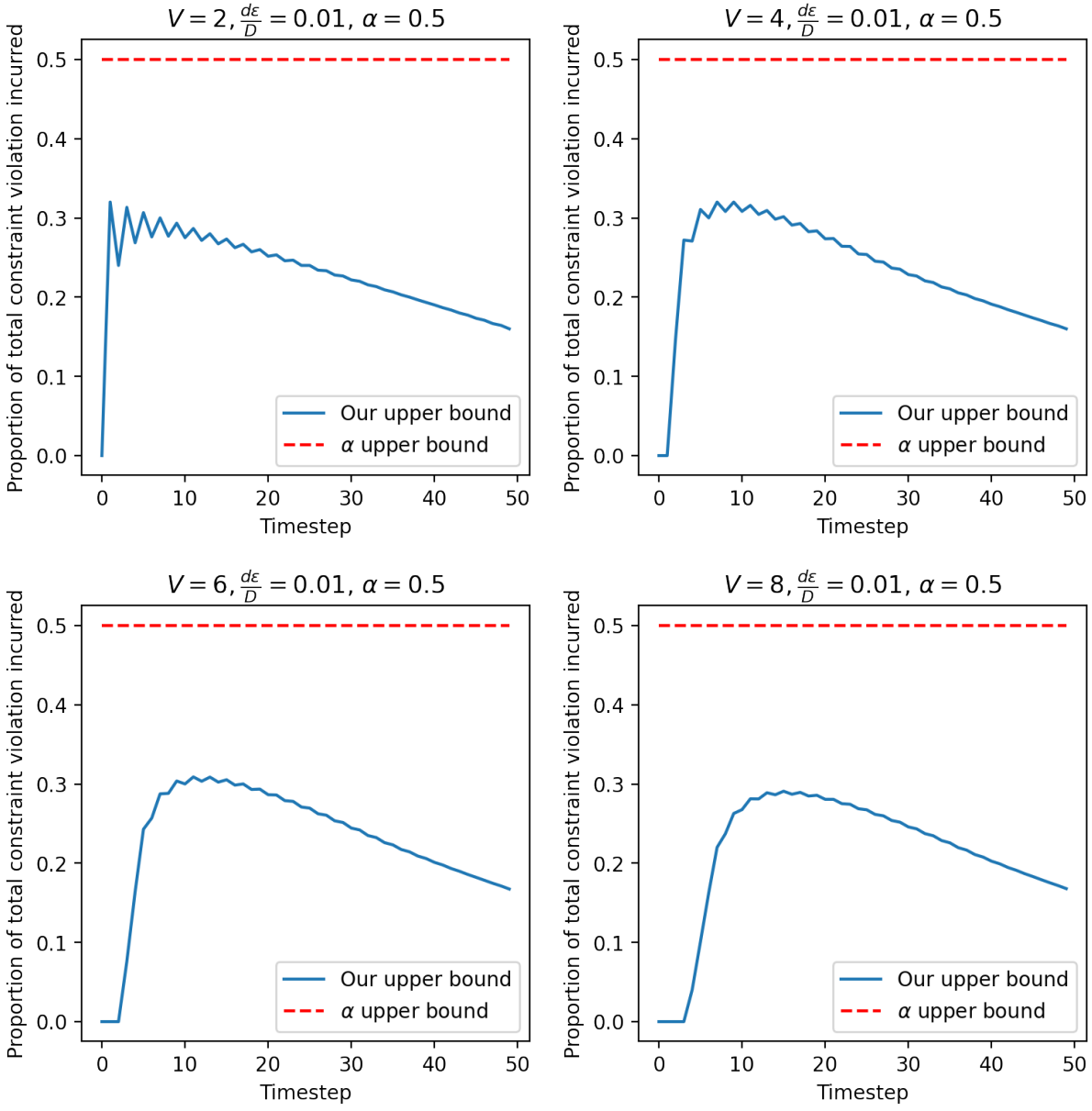

Figure 2.7: Evolution of ratio of total constraint violation, $\frac{d\epsilon}{D} = 0.01, \alpha = 0.5$

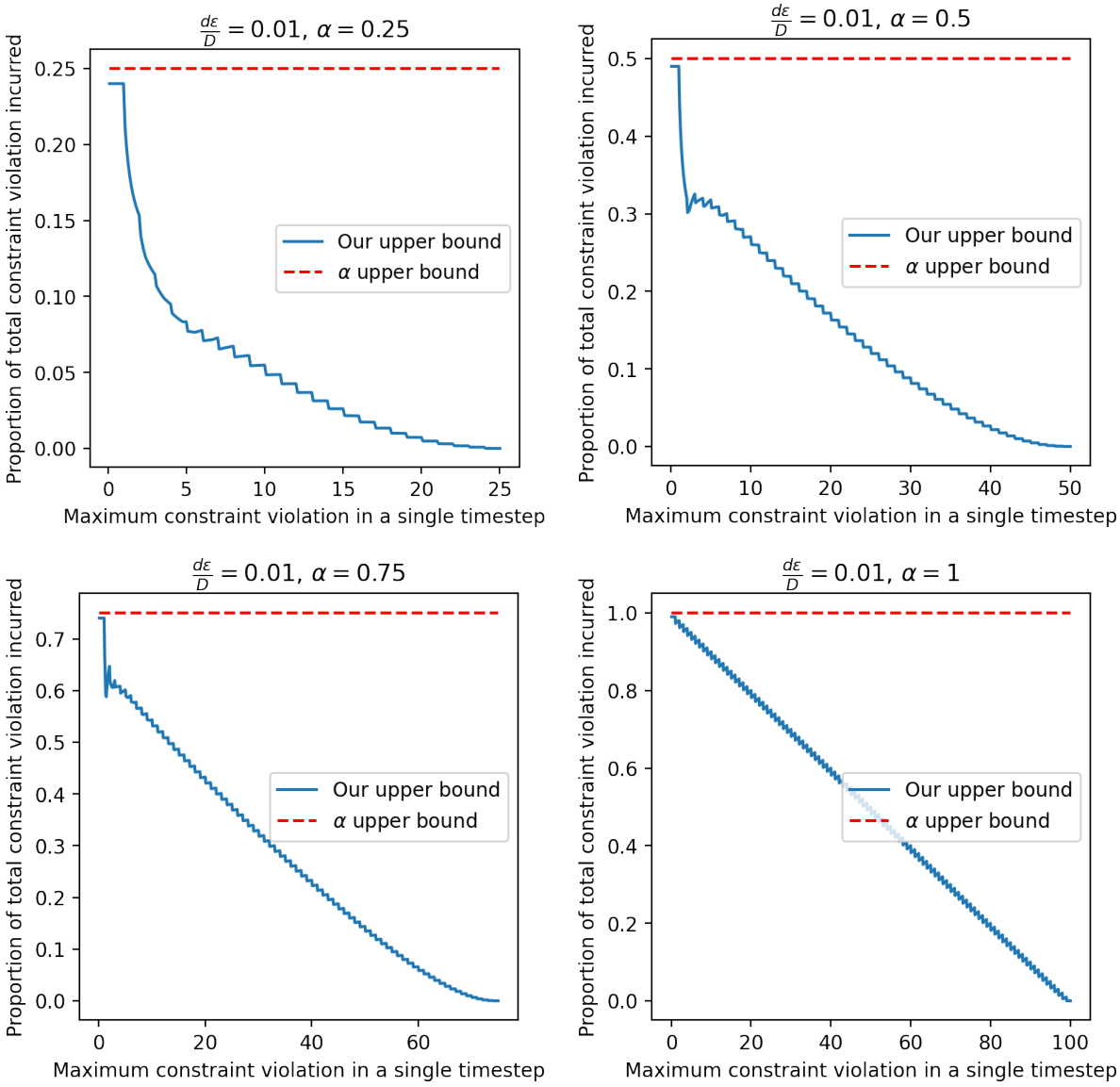

Figure 2.8: Plots of upper bound on proportion of total constraint violation incurred with resource augmentation

2.3 Competitive Difference With Respect to Component Algorithms

Algorithm 1-competitive with ROB in general horizon

In this section, we focus on the the competitive difference with respect to ADV and ROB. In the previous section, we noted in Lemma 3 that Algorithm 1 can incur a competitive difference of $O(\epsilon^2)$ with ADV or ROB every time one algorithm becomes better than the other. Intuitively, this case occurs when we greedily go towards one algorithm when the cost benefit is not that large, and then the costs rapidly start changing rapidly in an adversarial manner. We note that adversarial costs can always force us to do worse than at least one of the algorithms in the general horizon; however, we now give an algorithm that is 1-competitive with ROB despite ramp constraints. The main idea is that we hedge against the cost we would incur if ROB rapidly improves and we would need to slowly jump back towards ROB.

Suppose ALG is currently at convex parameter λ_t, where $\lambda = 0$ is ROB and $\lambda = 1$ is ADV. Note that

$$f(\mathsf{ALG}_t) = \lambda f(\mathsf{ADV}_t) + (1-\lambda) f(\mathsf{ROB}_t)$$
$$f(\mathsf{ALG}_t) - f(\mathsf{ROB}_t) = \lambda_t (f(\mathsf{ADV}_t) - f(\mathsf{ROB}_t))$$

We have previous established that ALG can change its λ_t by at least $\pm\frac{d\epsilon}{D}$ in each step. So, if ALG adopts a policy of stepping back towards $\lambda = 0$ as much as possible, ALG is guaranteed to be able to step back to ROB in $\lceil \lambda_t / \frac{d\epsilon}{D} \rceil$ steps or less. Define $\mathcal{T}(T, \lambda) := \{T+1, T+2, \ldots, T + \lceil \lambda_T / \frac{d\epsilon}{D} \rceil\}$. Then, we define

$$g(\lambda_T, \mathsf{ADV}_T, \mathsf{ROB}_T) \triangleq \max_{\mathsf{ADV}_t, \mathsf{ROB}_t, t \in \mathcal{T}(T,\lambda)} \sum_{t \in \mathcal{T}(T,\lambda)} \lambda_t (f(\mathsf{ADV}_t) - f(\mathsf{ROB}_t))$$

where $\lambda_t, t \in \mathcal{T}(T, \lambda)$ correspond to ALG stepping towards $\lambda = 0$ as fast as possible given the dispatches of ADV and ROB.

Algorithm 2: Ramp-Constrained Learning-Augmented Algorithm, Resource Augmented

Data: ramp box $[-d, d]^n$, resource augmentation factor ϵ

Result:

1 **for** $t \in \{1, 2, 3, \ldots\}$ **do**

2 | ALG_t plays the largest λ_t possible while following ramp constraint and $g(\lambda_t, \mathsf{ADV}_t, \mathsf{ROB}_t) \leq \Delta C_t$

3 **end**

Note that we can consider Algorithms 1 and 2 as two meta-algorithms that can be combined. The former determines the maximum λ that can be selected in order to not incur too much total

constraint violation, while the latter determines how to update λ within the limit set by the former. So, when we analyze 2, we assume that $\alpha = 1$.

Proposition 4. *Algorithm 2 is 1-competitive with respect to* ROB

Proof of Proposition 4. we first prove that if $g(\lambda_T, \mathsf{ADV}_T, \mathsf{ROB}_T) \leq \Delta C_T$, then $g(\lambda_{T+1}, \mathsf{ADV}_{T+1}, \mathsf{ROB}_{T+1}) \leq \Delta C_{T+1}$ if we minimize λ_{T+1} within the ramp constraints. Let $\lambda_t^*, \mathsf{ADV}_t^*, \mathsf{ROB}_t^*, t \in \mathcal{T}(T, \lambda_t)$ correspond to the optimal values for $g(\lambda_T, \mathsf{ADV}_T, \mathsf{ROB}_T)$.

$$\begin{aligned}
g(\lambda_T, \mathsf{ADV}_T, \mathsf{ROB}_T) &= \lambda_{T+1}^*(f(\mathsf{ADV}_{T+1}^*) - f(\mathsf{ROB}_{T+1}^*)) + g(\lambda_{T+1}^*, \mathsf{ADV}_{T+1}^*, \mathsf{ROB}_{T+1}^*) \\
&\geq \lambda_{T+1}(f(\mathsf{ADV}_{T+1}) - f(\mathsf{ROB}_{T+1})) + g(\lambda_{T+1}, \mathsf{ADV}_{T+1}, \mathsf{ROB}_{T+1}) \\
g(\lambda_{T+1}^*, \mathsf{ADV}_{T+1}^*, \mathsf{ROB}_{T+1}^*) &\leq g(\lambda_T, \mathsf{ADV}_T, \mathsf{ROB}_T) - \lambda_{T+1}^*(f(\mathsf{ADV}_{T+1}^*) - f(\mathsf{ROB}_{T+1}^*)) \\
&\leq \Delta C_T - \lambda_{T+1}^*(f(\mathsf{ADV}_{T+1}^*) - f(\mathsf{ROB}_{T+1}^*)) \\
&= \Delta C_{T+1}
\end{aligned}$$

We can always maintain $g(\lambda_t, \mathsf{ADV}_t, \mathsf{ROB}_t) \leq \Delta C_t$ by minimizing λ_t. So, when ALG_t plays the largest viable λ_t, there must exist at least one viable option.
Next, we will show that $\Delta C_t \geq 0$ for all time via induction. In the base case, $t = 1$. If $f(\mathsf{ROB}_t) \leq f(\mathsf{ADV}_t)$, we maintain $\lambda = 0$, so $\Delta C_1 = 0$. If $f(\mathsf{ROB}_t) > f(\mathsf{ADV}_t)$, any choice of λ_1 ensures $\Delta C_1 \geq 0$.
Next, we perform the inductive step. Suppose, $\Delta C_{t-1} \geq 0$. If $f(\mathsf{ROB}_t) \leq f(\mathsf{ADV}_t)$, we note that $g(\lambda, \mathsf{ADV}_t, \mathsf{ROB}_t) \geq 0$ for all λ, and our algorithm ensures that $\Delta C_t \geq g(\lambda_t, \mathsf{ADV}_t, \mathsf{ROB}_t) \geq 0$. On the other hand, $f(\mathsf{ROB}_t) > f(\mathsf{ADV}_t)$, we must have that $\Delta C_t \geq \Delta C_{t-1} \geq 0$. In either case, $\Delta C_t \geq 0$, so our induction is complete. Since our algorithm maintains $\Delta C_t \geq 0$ for all t, it is 1-competitive with respect to ROB. □

This algorithm handles the case where we go towards ADV_t, but then ROB rapidly becomes better than ADV and we are not able to switch back in time, causing us to incur a lot of additional cost compared to ROB. At every step, our algorithm stays close enough to ROB so that if ROB becomes better, the extra cost ALG incur is less than or equal to the benefit it already got from going towards ADV. In addition, if $f(\mathsf{ADV}_t) - f(\mathsf{ROB}_t)$ is sufficiently negative, $g(\lambda, \mathsf{ADV}_t, \mathsf{ROB}_t) < 0 \leq \Delta C_t$ for all λ, so our algorithm is free to go all the way towards ADV.

Upper Bound on Competitive Difference With Respect to ADV
We next show that our algorithm maintains a finite competitive difference with respect to ADV in the case where ADV always maintains a cost no greater than ROB.

Theorem 1. *If* ADV *never incurs a higher cost than* ROB *at any timestep, then the competitive difference of Algorithm 2 with respect to* ADV *is* $O(\epsilon^{-3})$.

Proof of Theorem 1. Define $c_t = f(\mathsf{ADV}_t) - f(\mathsf{ROB}_t)$. Therefore,

$$\sum_t c_t = \overbrace{\sum_t f(\mathsf{ALG}_t) - f(\mathsf{ADV}_t)}^{\text{regret wrt ADV}} + \overbrace{\sum_t f(\mathsf{ROB}_t) - f(\mathsf{ALG}_t)}^{\text{benefit wrt ROB}}$$

The sketch of our proof is as follows. We have previously shown that $\sum_t f(\mathsf{ROB}_t) - f(\mathsf{ALG}_t) \geq 0$ for all trajectories. We claim that if $\sum_t f(\mathsf{ALG}_t) - f(\mathsf{ADV}_t) \geq \gamma$, where γ is some threshold, then $\sum_t c_t \geq \gamma$. We then show that while $\sum_t c_t \geq \gamma$, then our algorithm is playing $\lambda = 1$, and we cannot be incurring any more regret with respect to ADV. This establishes γ as a finite upper limit on the regret wrt ADV in the case where $c_t \geq 0\ \forall t$.

Note that while $c_t \geq 0$, $g(c_t, \lambda = \epsilon) = 0$. We have previously established that $\sum_{t=1}^{T} f(\mathsf{ROB}_t) - f(\mathsf{ALG}_t) \geq 0$ for all possible trajectories, so $\lambda_t \geq \epsilon\ \forall t$. Furthermore, this implies that $\sum_{t=1}^{T} f(\mathsf{ROB}_t) - f(\mathsf{ALG}_t) \geq \epsilon \sum_t c_t\ \forall t$. Now, we consider $g(c_t, \lambda = 1)$. If $c_t \geq 0$, then the risk is maximized by $c_t = 0$. In order to play $\lambda = 1$, we need $\sum_{t=1}^{T} f(\mathsf{ROB}_t) - f(\mathsf{ALG}_t) \geq g(c_t = 0, \lambda = 1)$. To get a loose upper bound, let C be the maximum possible c_t. c_t cannot be unbounded because our feasible region is bounded. If $\sum_{t=1}^{T^*} f(\mathsf{ROB}_t) - f(\mathsf{ALG}_t) \geq g(c_t = 0, \lambda = 1) + \frac{1}{\epsilon}C$ then $\sum_{t=1}^{T^*-1/\epsilon} f(\mathsf{ROB}_t) - f(\mathsf{ALG}_t) \geq g(c_t = 0, \lambda = 1)$. Starting at time $T^* - 1/\epsilon$, ALG has been trying to set λ to 1 within the ramp constraints, so by time T^*, ALG must have reached $\lambda = 1$. $\sum_{t=1}^{T} f(\mathsf{ROB}_t) - f(\mathsf{ALG}_t) \geq g(c_t = 0, \lambda = 1)$ for all $T \geq T^*/\epsilon$ since $c_t \geq 0\ \forall t$, so ALG will stay at $\lambda = 1$ for the rest of the trajectory and stop incurring regret with respect to ADV.

If $\sum_{t=1}^{T} f(\mathsf{ALG}_t) - f(\mathsf{ADV}_t) \geq \frac{1}{\epsilon}\left(g(c_t = 0, \lambda = 1) + \frac{1}{\epsilon}C\right)$, then $\sum_{t=1}^{T} c_t \geq \frac{1}{\epsilon}\left(g(c_t = 0, \lambda = 1) + \frac{1}{\epsilon}C\right)$, and $\sum_{t=1}^{T} f(\mathsf{ROB}_t) - f(\mathsf{ALG}_t) \geq g(c_t = 0, \lambda = 1) + \frac{1}{\epsilon}C$. We have shown that this implies for $t \geq T$, ALG is playing $\lambda = 1$, so the regret with respect to ADV is capped at $\frac{1}{\epsilon}\left(g(c_t = 0, \lambda = 1) + \frac{1}{\epsilon}C\right)$.

$g(c_t = 0, \lambda = 1)$ is $O(1/\epsilon^2)$, so our regret with respect to ADV is $O(1/\epsilon^3)$. □

Lower Bound on Competitive Difference With Respect to ADV

Suppose we start at $c_0 = 0, \lambda = 0$. Consider the case where c_t grows by $\overline{c}$ at every step, reminiscent of the adversarial example in Lemma 3. We can lower bound the 'regret' with respect to ADV

if we assume that λ grows by $\frac{d\epsilon}{D}$ at every step. Define $N = \frac{D}{d\epsilon}$. In this case, the total regret is $\sum_{t=1}^{N} t\overline{c} \cdot (N-t)\frac{d\epsilon}{D} = \overline{c}\frac{d\epsilon}{D}\sum_{t=1}^{D/d\epsilon} t(N-t) = \Omega(1/\epsilon^2)$.

Our lower and upper bounds are therefore only separated by a factor of $1/\epsilon$.

Simulations of Algorithm 2

Theorem 1 only applies to the case where ADV never achieves cost lower than ROB. We were unable to obtain an upper bound on the general case, due to the potential of an adversarial case in which ADV first has slightly higher cost than ROB in order to influence our algorithm into playing a lower λ. Then, ADV could rapidly become lower cost, and since ALG started at a lower λ, it would incur a larger competitive difference as it chases ADV. We simulate Algorithm 2 across more general scenarios and show our results in Figure 2.9.

In the first case, the cost of ADV rapidly becomes lower than ROB, and ALG quickly increases λ in response. Due to the ramp constraints, we incur a competitive difference. In the second case, ADV becomes starts slightly better, briefly becomes higher cost, then rapidly becomes lower cost than ROB in the same way as the first case. In this second case, we see that the overall competitive difference is noticeably lower than the first case. Finally, in the third case, ADV and ROB oscillate back and forth between having lower cost than the other. Interestingly, the competitive difference of Algorithm 2 soon becomes negative compared to ADV. Our experimental results suggest that the competitive difference of Algorithm 2 with respect to ADV is finite. In particular, our simulation results suggest that the competitive difference is maximized in the first case in Figure 2.9 in which ADV rapidly becomes better, and does not try to make adversarial adjustments to influence ALG into playing a smaller λ.

2.4 Discussion and Future Work

Our lower bound in Proposition 1 demonstrates the difficulty of designing effective algorithms that respect ramp constraints in the general case. However, our experimental results for Algorithm 1 illustrate that we are able to get more optimistic results by considering resource augmentation and the magnitude of constraint violations we are trying to protect against. Our theoretical and experimental results for Algorithm 2 demonstrate that it is possible to get worst-case guarantees with respect to a robust baseline, even in the presence of ramp constraints.

In future work, it would be illuminating to determine whether the competitive difference of Algorithm 2 with respect to ADV is bounded in the general case. In addition, it would be useful to prove Conjecture 1, as well as establishing lower bounds on the proportion of total constraint violation that can be incurred in the presence of resource augmentation.

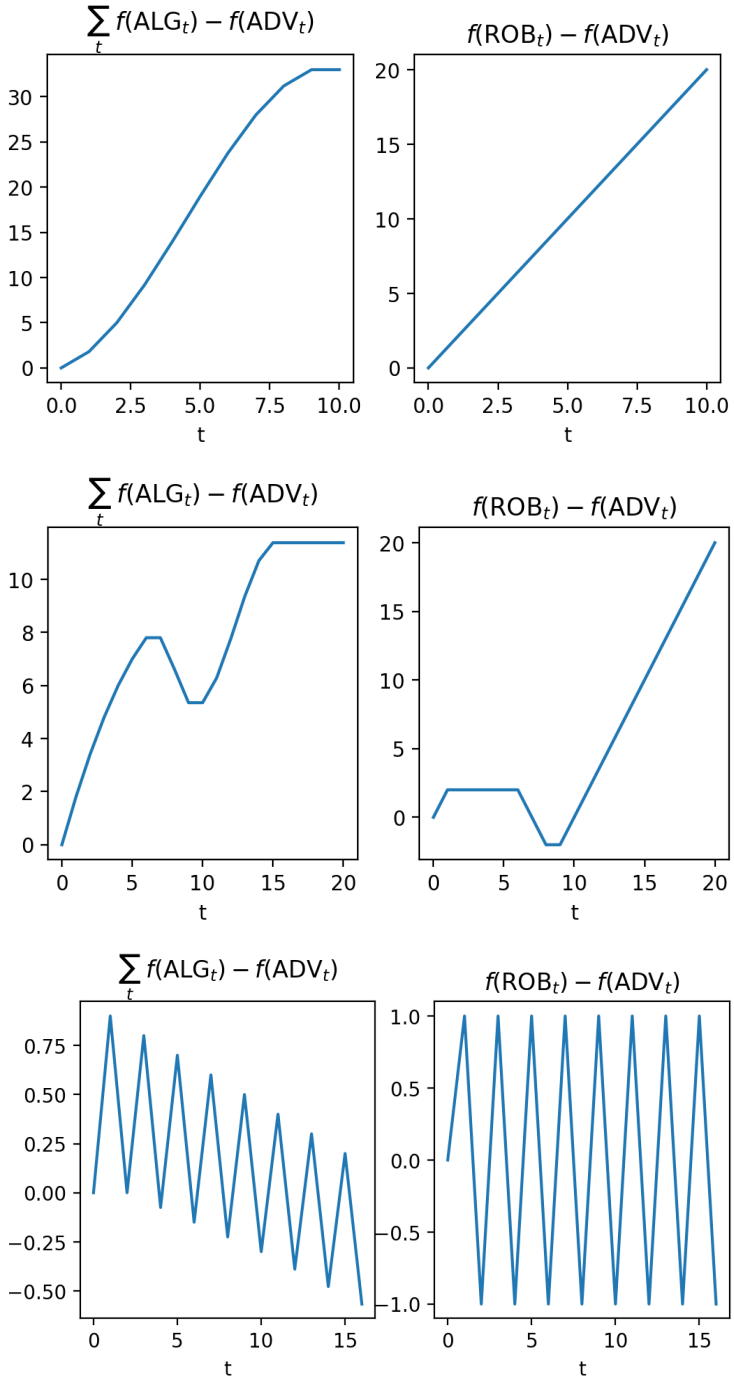

Figure 2.9: Simulations of Algorithm 2 with $\overline{c} = 2$, $d\epsilon/D = 0.1$ in three cases of interest: ADV always better than ROB, ADV better that ROB in all but a few timesteps, ADV and ROB alternating in relative performance.

TRAINING A MACHINE LEARNING MODEL TO ENFORCE CONSTRAINTS IN A REAL-WORLD CO-GENERATION PLANT

In the previous chapter, we considered how an ML advice algorithm could be effectively integrated into a learning-augmented algorithm for dispatch. In this chapter, we consider the complementary problem of how to train an ML algorithm for the dispatch problem to begin with. The main technical challenge is in training an ML model that adheres to complex operational constraints while still maintaining good performance. We develop a training procedure that enables the ML algorithm to adhere to operational constraints while simultaneously still maintaining good profit. We also introduce a constraint enforcement layer into our model architecture that strictly enforces a subset of operational constraints. We demonstrate the efficacy of our ML models on a real-world co-generation plant and its historical data. In particular, the co-generation environment that we train in captures many of the intricacies of real-world economic dispatch that a simpler toy model would not.

It is important to emphasize that the ML model discussed in this chapter does not exactly match the ML advice algorithm considered Chapter 2. In Chapter 2, we assume that the cost function is affine and constant, the ramp constraints form a hypercube, and that the operational constraints can be described by a simple linear function of the input parameters. The co-generation model we use in this chapter does not have any of those simplifying assumptions. On the other hand, the ML model we develop in this chapter produces a dispatch at each timestep independently of the others, so it does not consider ramp constraints.

3.1 Model and Preliminaries

Our industry partner Beyond Limits supplied us with historical data and a neural-network model for a real-world co-generation plant that produces both power and steam. The plant has complex operational constraints that need to be met for any given dispatch, and our goal was to investigate whether a machine learning model could be trained to produce dispatches that met the plant's internal constraints. The plant had both static and dynamic constraints; static constraints on input parameters did not change, whereas the dynamic constraints would change based on the particular input. We were not given an explicit set of constraints, but rather a black-box neural network model of the plant that would take a proposed dispatch as input and produce the corresponding set of

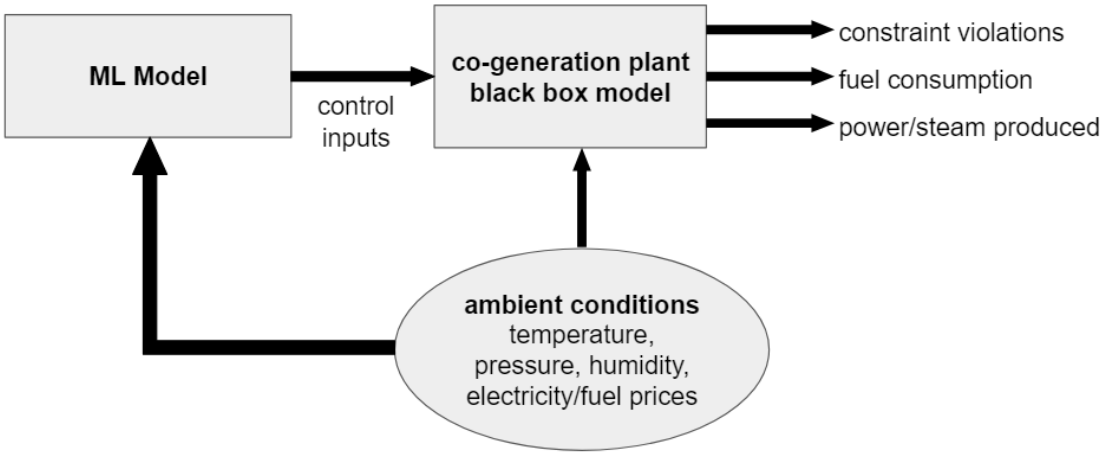

Figure 3.1: Overall system model that demonstrates how our ML model produces dispatches for the co-generation plant

constraints as an output. As such, it was not possible to ascertain whether a proposed dispatch would follow the constraints ahead of time. The model is also a modified version of the co-geneneration environment in SustainGym, which is a collection of environments meant to provide a testbench for reinforcement learning algorithms for sustainability applications [55]. One difference is steam supply-demand is incorporated into the loss function in SustainGym, whereas our model explicitly enforces steam supply-demand balance.

The overall system is shown in Figure 3.1. Our ML model was to take in the ambient conditions at any given time and propose a set of inputs for the co-generation plant. The proposed dispatch and the ambient conditions would be inputted into the neural network model of the plant, which would then output the extent of constraint violations, the amount of fuel consumed, and the amount of power and steam produced. With these values, we can then evaluate how much profit the plant would expect for that timestep compared to the amount of constraint violations. The plant model did not consider violations of ramp constraints, and due to the difficulty of training the ML model to enforce black-box plant constraints, we opted to not consider ramp constraints for our ML model either. Without the presence of ramp constraints, dispatches across timesteps are not coupled with each other, so we only need our model to produce dispatches for a single timestep at a time.

Our goals for the model are to:

- enforce steam supply-demand balance
- enforce the system constraints reflected in the plant model

- maximize the profit at each timestep

Let θ_t be the ambient conditions at timestep t, and let $\{x_t |\ h(x_t, \theta_t) \leq r\}$ define the set of dispatches that satisfy the system operational constraints. h and r are constant throughout time and are captured via the neural-network plant model. The problem we are training our ML model to solve is then:

$$\begin{aligned} \min_{\{x_t\}_{t=1}^T} \quad & \sum_{t\in[T]} f_t(x_t, \theta_t) \\ \text{s.t.} \quad & h(x_t, \theta_t) \leq r \qquad \forall t \\ & g(x_t) = s_t \qquad \forall t \end{aligned} \tag{3.1}$$

To enforce steam supply-demand balance, we use the gauge map method introduced in [56]. This is a differentiable transformation that projects our ML model output onto the set of dispatches whose steam output matches the demand at that time. The gauge map works best with polyhedral constraint sets, which fits with steam supply-demand balance constraints since the steam output is a linear function of our input parameters.

On the other hand, we cannot enforce the system constraint or maximize the profit using the gauge map due to the black-box nature of the plant model. In order to train our ML model to enforce these, we put them into our loss function, as described in the following section.

3.2 Model Architecture and Training

The plant model produces the profit at each stage as well as a vector containing the constraint violations on each input parameter. We take the overall constraint violation to be the ℓ_2 norm of the constraint violations vector. We also have a penalty term λ that weights profit with constraint violation. Our ML model does not incorporate classical optimization techniques, but instead performs task-based training with the training objective $-\text{profit} + \lambda \cdot (\text{constraint violation})$. We found that training was more stable when we minimized the equivalent objective

$$\frac{1}{\lambda}(-\text{profit}) + \text{constraint violation}$$

This ensured that the magnitude of the loss does not change significantly for different values of λ.

In order to train our model, we employed the penalty method and increased λ in stages over $\lambda \in \{10, 100, 1000, 10000, 100000\}$. For each λ we trained for 100 epochs, 10 batches per epoch, and a batch size of 100 datapoints. We used a learning rate of .01 and automatically decreased it using ReduceLROnPlateau.

For the architecture, we settled on a simple two-layer feed-forward ReLU network with a final layer that implements the gauge map transformation. By keeping our network simple, we aim to demonstrate that we can achieve high performance in this setting using simple ML architectures.

3.3 Training Results

Our training results are shown in Figure 3.2 over 20 training runs across different random seeds. We see that there is a clear Pareto frontier shown by the dashed blue line that we are often able to approach. Notably, we see that it is possible to train the model to achieve near-zero constraint violations while still having sizable profits. There is considerable variance in the quality of the models we are able to learn at each value of λ. For instance, the constraint violation only approached zero at $\lambda = 10^5$ for five of the training runs, and sometimes still ended as high as 50. This is reasonable given the inherent non-convexity of the neural-network based plant model, so there should be many local minima that can be difficult to escape. If we compare models with the same λ, we see that their constraint violation and profit both decrease significantly on average as we increase λ from 10 to 10^4. By the time λ increases from 10^4 to 10^5, the constraint violation and profit stops decreasing as much; this is reasonable since when λ is sufficiently high, the model is primarily learning to minimize the constraint violations rather than negative profit.

3.4 Discussion and Future Work

We are able to produce a ML model with good performance that has near zero constraint violation while also exactly matching the steam demand. This demonstrates the efficacy of using ML algorithms for energy applications, as we were able to achieve good performance with a very simple network. For our model, one next step is to change it to produce dispatches for multiple timesteps at once; this initially proved difficult to train compared to our "single timestep" model. We anticipate that we can employ similar techniques to enforce other constraints. For instance, to employ ramp constraints we can add in a projection layer to keep the dispatch at each timestep within the ramp constraints of the previous dispatch. For non-convex constraints like up-time or down-time constraints, we anticipate that we can also explicitly penalize them in the loss function, just as we penalized the original constraint violations from the plant model.

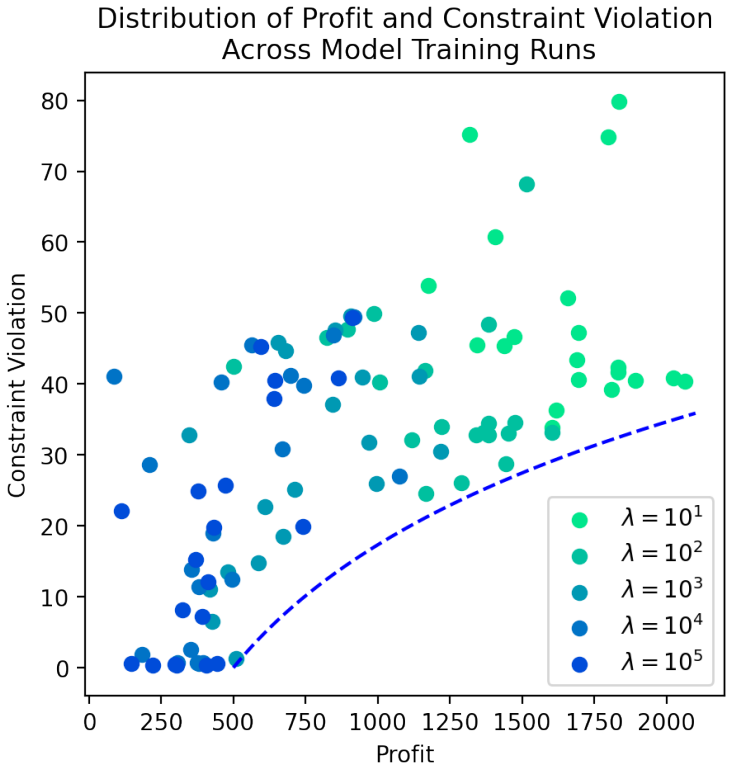

Figure 3.2: Results from 20 training runs initialized on different random seeds

[1] Lachlan Andrew et al. "A tale of two metrics: Simultaneous bounds on competitiveness and regret". In: *Conference on Learning Theory*. PMLR. 2013, pp. 741–763.

[2] Spyros Angelopoulos et al. "Online computation with untrusted advice". In: *arXiv preprint arXiv:1905.05655* (2019).

[3] Antonios Antoniadis and Kevin Schewior. "A tight lower bound for online convex optimization with switching costs". In: *International Workshop on Approximation and Online Algorithms*. Springer. 2017, pp. 164–175.

[4] Masoud Badiei, Na Li, and Adam Wierman. "Online convex optimization with ramp constraints". In: *2015 54th IEEE Conference on Decision and Control (CDC)*. 2015, pp. 6730–6736. DOI: 10.1109/CDC.2015.7403279.

[5] Nikhil Bansal et al. "A 2-competitive algorithm for online convex optimization with switching costs". In: *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2015)*. Schloss-Dagstuhl-Leibniz Zentrum für Informatik. 2015.

[6] A. Blum et al. "A decomposition theorem and bounds for randomized server problems". In: *2013 IEEE 54th Annual Symposium on Foundations of Computer Science*. Los Alamitos, CA, USA: IEEE Computer Society, Oct. 1992, pp. 197–207. DOI: 10.1109/SFCS.1992.267772. URL: https://doi.ieeecomputersociety.org/10.1109/SFCS.1992.267772.

[7] Avrim Blum and Carl Burch. "On-line learning and the metrical task system problem". In: *Proceedings of the Tenth Annual Conference on Computational Learning Theory*. 1997, pp. 45–53.

[8] Allan Borodin, Nathan Linial, and Michael E. Saks. "An optimal on-line algorithm for metrical task system". In: *J. ACM* 39.4 (Oct. 1992), pp. 745–763. ISSN: 0004-5411. DOI: 10.1145/146585.146588. URL: https://doi.org/10.1145/146585.146588.

[9] Sébastien Bubeck, Christian Coester, and Yuval Rabani. "The Randomized k-Server Conjecture Is False!" In: *Proceedings of the 55th Annual ACM Symposium on Theory of Computing*. 2023, pp. 581–594.

[10] Niv Buchbinder et al. "Unified algorithms for online learning and competitive analysis". In: *Conference on Learning Theory*. JMLR Workshop and Conference Proceedings. 2012, pp. 5–1.

[11] Bingqing Chen et al. "Enforcing policy feasibility constraints through differentiable projection for energy optimization". In: *Proceedings of the Twelfth ACM International Conference on Future Energy Systems*. 2021, pp. 199–210.

[12] Niangjun Chen, Gautam Goel, and Adam Wierman. "Smoothed online convex optimization in high dimensions via online balanced descent". In: *Conference On Learning Theory*. PMLR. 2018, pp. 1574–1594.

[13] Niangjun Chen et al. "Online convex optimization using predictions". In: *Proceedings of the 2015 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems*. 2015, pp. 191–204.

[14] Niangjun Chen et al. "Using predictions in online optimization: Looking forward with an eye on the past". In: *ACM SIGMETRICS Performance Evaluation Review* 44.1 (2016), pp. 193–206.

[15] Yize Chen, Ling Zhang, and Baosen Zhang. "Learning to solve DCOPF: A duality approach". In: *Electric Power Systems Research* 213 (2022), p. 108595.

[16] Nicolas Christianson et al. "Dispatch-aware planning for feasible power system operation". In: *Electric Power Systems Research* 212 (2022), p. 108597.

[17] Nicolas Christianson et al. "Robustifying machine-learned algorithms for efficient grid operation". In: *NeurIPS 2022 Workshop on Tackling Climate Change with Machine Learning*. 2022. URL: https://www.climatechange.ai/papers/neurips2022/19.

[18] Wenqi Cui, Jiayi Li, and Baosen Zhang. "Decentralized safe reinforcement learning for inverter-based voltage control". In: *Electric Power Systems Research* 211 (2022), p. 108609.

[19] Ilias Diakonikolas et al. "Learning online algorithms with distributional advice". In: *International Conference on Machine Learning*. PMLR. 2021, pp. 2687–2696.

[20] Priya Donti et al. "Adversarially robust learning for security-constrained optimal power flow". In: *Advances in Neural Information Processing Systems* 34 (2021), pp. 28677–28689.

[21] Antonio Frangioni, Claudio Gentile, and Fabrizio Lacalandra. "Solving unit commitment problems with general ramp constraints". In: *International Journal of Electrical Power & Energy Systems* 30.5 (2008), pp. 316–326.

[22] Yoav Freund and Robert E Schapire. "A decision-theoretic generalization of on-line learning and an application to boosting". In: *Journal of Computer and System Sciences* 55.1 (1997), pp. 119–139.

[23] Joel Friedman and Nathan Linial. "On convex body chasing". In: *Discrete & Computational Geometry* 9.3 (1993), pp. 293–321.

[24] Gautam Goel and Adam Wierman. "An online algorithm for smoothed regression and lqr control". In: *The 22nd International Conference on Artificial Intelligence and Statistics*. PMLR. 2019, pp. 2504–2513.

[25] Gautam Goel et al. "Beyond online balanced descent: An optimal algorithm for smoothed online optimization". In: *Advances in Neural Information Processing Systems* 32 (2019).

[26] Fanghong Guo et al. "An alternative learning-based approach for economic dispatch in smart grid". In: *IEEE Internet of Things Journal* 8.19 (2021), pp. 15024–15036.

[27] Elad Hazan. *Introduction to Online Convex Optimization*. 2023. arXiv: 1909.05207 [cs.LG].

[28] Bin Huang and Jianhui Wang. "Applications of physics-informed neural networks in power systems-a review". In: *IEEE Transactions on Power Systems* 38.1 (2022), pp. 572–588.

[29] Wanjun Huang et al. "DeepOPF-V: Solving AC-OPF problems efficiently". In: *IEEE Transactions on Power Systems* 37.1 (2021), pp. 800–803.

[30] Zhihao Jiang, Debmalya Panigrahi, and Kevin Sun. "Online algorithms for weighted paging with predictions". In: *ACM Transactions on Algorithms (TALG)* 18.4 (2022), pp. 1–27.

[31] Seung-Jun Kim and Geogios B Giannakis. "Real-time electricity pricing for demand response using online convex optimization". In: *ISGT 2014*. IEEE. 2014, pp. 1–5.

[32] Silvio Lattanzi et al. "Online scheduling via learned weights". In: *Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms*. SIAM. 2020, pp. 1859–1877.

[33] Adam Lechowicz et al. "Chasing Convex Functions with Long-term Constraints". In: *arXiv preprint arXiv:2402.14012* (2024).

[34] Adam Lechowicz et al. "Online Conversion with Switching Costs: Robust and Learning-Augmented Algorithms". In: *arXiv preprint arXiv:2310.20598* (2023).

[35] Russell Lee et al. "Online Search with Predictions: Pareto-optimal Algorithm and its Applications in Energy Markets". In: *The 15th ACM International Conference on Future and Sustainable Energy Systems*. 2024, pp. 50–71.

[36] Yingying Li and Na Li. "Leveraging predictions in smoothed online convex optimization via gradient-based algorithms". In: *Advances in Neural Information Processing Systems* 33 (2020), pp. 14520–14531.

[37] Yingying Li, Guannan Qu, and Na Li. "Using predictions in online optimization with switching costs: A fast algorithm and a fundamental limit". In: *2018 Annual American Control Conference (ACC)*. IEEE. 2018, pp. 3008–3013.

[38] Lin Lin et al. "Deep reinforcement learning for economic dispatch of virtual power plant in internet of energy". In: *IEEE Internet of Things Journal* 7.7 (2020), pp. 6288–6301.

[39] Minghong Lin et al. "Dynamic right-sizing for power-proportional data centers". In: *IEEE/ACM Transactions on Networking* 21.5 (2012), pp. 1378–1391.

[40] Minghong Lin et al. "Online algorithms for geographical load balancing". In: *2012 International Green Computing Conference (IGCC)*. IEEE. 2012, pp. 1–10.

[41] Nick Littlestone and Manfred K Warmuth. "The weighted majority algorithm". In: *Information and Computation* 108.2 (1994), pp. 212–261.

[42] Thodoris Lykouris and Sergei Vassilvitskii. "Competitive caching with machine learned advice". In: *Journal of the ACM (JACM)* 68.4 (2021), pp. 1–25.

[43] Xiang Pan et al. "DeepOPF-AL: Augmented Learning for Solving AC-OPF Problems with a Multi-Valued Load-Solution Mapping". In: *Proceedings of the 14th ACM International Conference on Future Energy Systems*. 2023, pp. 42–47.

[44] Manish Purohit, Zoya Svitkina, and Ravi Kumar. "Improving online algorithms via ML predictions". In: *Advances in Neural Information Processing Systems* 31 (2018).

[45] Dhruv Rohatgi. "Near-optimal bounds for online caching with machine learned advice". In: *Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms*. SIAM. 2020, pp. 1834–1845.

[46] Ziv Scully, Isaac Grosof, and Michael Mitzenmacher. "Uniform bounds for scheduling with job size estimates". In: *arXiv preprint arXiv:2110.00633* (2021).

[47] Shai Shalev-Shwartz et al. "Online learning and online convex optimization". In: *Foundations and Trends® in Machine Learning* 4.2 (2012), pp. 107–194.

[48] Uri Sherman and Tomer Koren. "Lazy OCO: Online convex optimization on a switching budget". In: *Conference on Learning Theory*. PMLR. 2021, pp. 3972–3988.

[49] Ming Shi, Xiaojun Lin, and Sonia Fahmy. "Competitive online convex optimization with switching costs and ramp constraints". In: *IEEE/ACM Transactions on Networking* 29.2 (2021), pp. 876–889.

[50] Wenbo Shi et al. "Real-time energy management in microgrids". In: *IEEE Transactions on Smart Grid* 8.1 (2015), pp. 228–238.

[51] Alva J Svoboda et al. "Short-term resource scheduling with ramp constraints [power generation scheduling]". In: *IEEE Transactions on Power Systems* 12.1 (1997), pp. 77–83.

[52] Yu Wang, Shiwen Mao, and R Mark Nelms. "Online algorithm for optimal real-time energy distribution in the smart grid". In: *IEEE Transactions on Emerging Topics in Computing* 1.1 (2013), pp. 10–21.

[53] Le Xie and Marija D Ilic. "Model predictive economic/environmental dispatch of power systems with intermittent resources". In: *2009 IEEE Power & Energy Society General Meeting*. IEEE. 2009, pp. 1–6.

[54] Yan Yang et al. "Fast economic dispatch in smart grids using deep learning: An active constraint screening approach". In: *IEEE Internet of Things Journal* 7.11 (2020), pp. 11030–11040.

[55] Christopher Yeh et al. "SustainGym: Reinforcement Learning Environments for Sustainable Energy Systems". In: *Advances in Neural Information Processing Systems* 36 (2024).

[56] Ling Zhang, Daniel Tabas, and Baosen Zhang. "An Efficient Learning-Based Solver for Two-Stage DC Optimal Power Flow with Feasibility Guarantees". In: *arXiv preprint arXiv:2304.01409* (2023).

[57] Ling Zhang and Baosen Zhang. "Learning to solve the AC optimal power flow via a lagrangian approach". In: *2022 North American Power Symposium (NAPS)*. IEEE. 2022, pp. 1–6.

[58] Weifeng Zhong et al. “Online control and near-optimal algorithm for distributed energy storage sharing in smart grid”. In: *IEEE Transactions on Smart Grid* 11.3 (2019), pp. 2552–2562.