Optimization under Uncertainty through Problem Reformulations

Veviurko, G.

DOI
10.4233/uuid:d7608c29-13e3-4149-99c4-dd732137d562

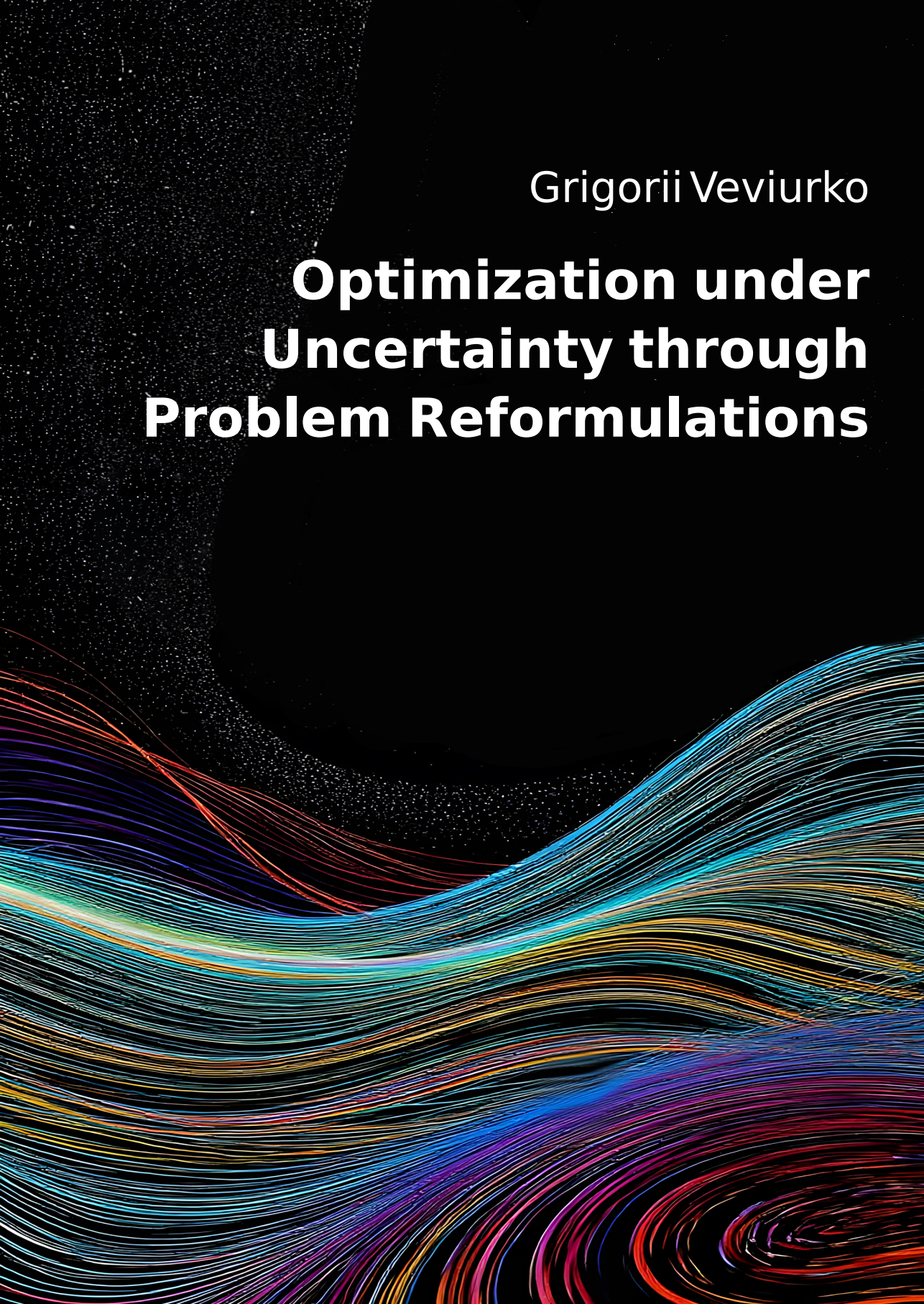
Publication date
2025

Document Version
Final published version

Citation (APA)
Veviurko, G. (2025). *Optimization under Uncertainty through Problem Reformulations*. [Dissertation (TU Delft), Delft University of Technology]. https://doi.org/10.4233/uuid:d7608c29-13e3-4149-99c4-dd732137d562

Grigorii Veviurko

Optimization under Uncertainty through Problem Reformulations



Optimization under Uncertainty through Problem Reformulations

Optimization under Uncertainty through Problem Reformulations

Dissertation

for the purpose of obtaining the degree of doctor at Delft University of Technology by the authority of the Rector Magnificus, Prof. dr. ir. T.H.J.J. van den Hagen chair of the Board for Doctorates to be defended publicly on Friday 12 September 2025 at 10:00 o'clock

by

Grigorii VEVIURKO
Master of Science in Computational Neuroscience
Technische Universität Berlin

Delft University of Technology, the Netherlands
born in Ufa, Russia

This dissertation has been approved by the promotors.

Composition of the doctoral committee:

Rector Magnificus,	Chairperson
Prof. Dr. M.M. de Weerdt,	Delft University of Technology, *promotor*
Dr. J. W. Böhmer,	Delft University of Technology, *copromotor*
Independent members:	
Prof. Dr. M.T.J. Spaan,	Delft University of Technology
Prof. Dr. Ir. J.A. la Poutré,	Delft University of Technology
Prof. Dr. J.L. Hurink,	University of Twente
Dr. H.C. van Hoof,	University of Amsterdam
Dr. M. Lombardi,	University of Bologna

Prof. Dr. M.M. de Weerdt and Dr. J. W. Böhmer of Delft University of Technology have contributed greatly to the preparation of this dissertation.

Keywords: optimization under uncertainty, convex optimization, decision-focused learning, reinforcement learning
Cover by: Tatiana Artemeva

To my mom and dad, who made me who I am

CONTENTS

SUMMARY

The research in this thesis falls within the realm of optimization under uncertainty, a crucial area in computer science and mathematics with broad applications in power systems, finance, machine learning, healthcare, and more. This thesis presents three main contributions across electric vehicle charging scheduling, decision-focused learning, and reinforcement learning. Beyond advancing the state of the art in each of these domains, our contributions emphasize the importance of effective problem formulations. Chapters 2-4 demonstrate how efficient solutions can emerge from formulations that address uncertainty, balance complexity with solution quality, promote computational tractability, and align with human intuition.

The first contribution addresses partial observability in the context of Electric Vehicle (EV) charging in DC microgrids, where EV locations are only known at the cable level. This setup, which may arise due to privacy concerns or communication constraints, complicates the optimization of power flows within the grid. Traditional approaches would require explicitly modeling every possible assignment of EVs to charging locations, quickly leading to computational infeasibility. In Chapter 2, we propose surrogate reformulations that redefine the grid configuration by treating cables as single nodes or parallel node structures, resulting in an uncertainty-agnostic formulation. Tested across simulated microgrids, this approach demonstrates how surrogate models can effectively manage aleatoric uncertainty while balancing computational efficiency and solution quality. In future work, similar techniques could be applied to other problems in power systems and beyond where partial observability arises due to communication limitations.

The second core contribution lies in the field of decision-focused learning (DFL), where predictive models are trained to estimate unknown parameters in constrained optimization problems. The essence of the decision-focused approach is to optimize decision quality rather than minimize prediction errors as the training objective. In Chapter 3, we demonstrate, both theoretically and experimentally, that DFL applied to convex optimization problems suffers from the zero-gradient issue – a phenomenon previously identified only in the linear case. By reformulating the problem using quadratic approximation and local smoothing, we mitigate this issue, enabling stable gradient propagation and, consequently, improved learning performance. This advancement broadens the applicability of decision-focused learning within convex optimization.

The final contribution introduces the max-reward paradigm in reinforcement learning, particularly suited for goal-reaching tasks where optimizing cumulative rewards can lead to convergence on suboptimal behaviors. By shifting the objective from cumulative to maximum reward, the proposed approach better aligns with the human interpretation of the task and offers notable algorithmic benefits. Experimental results demonstrate the effectiveness of this approach in goal-reaching problems, motivating further exploration of non-standard reward formulations in reinforcement learning.

Across these contributions, this thesis illustrates how optimization under uncertainty in various domains can benefit from tailored problem formulations that address uncertainty without explicit modeling, balance scalability with solution quality, and enhance the performance of learning approaches. Overall, this work contributes to various applications of optimization under uncertainty and highlights the potential for further research and the broader application of effective problem formulations across diverse fields.

SAMENVATTING

Het onderzoek in dit proefschrift valt binnen het domein van optimalisatie onder onzekerheid, een cruciaal gebied in de informatica en wiskunde met brede toepassingen in onder andere energiesystemen, financiën, en machine learning. De drie hoofdbijdragen van dit proefschrift betreffen het plannen van het laden van elektrische voertuigen, beslissingsgericht learning, en reinforcement learning. Naast het bevorderen van de state-of-the-art in elk van deze domeinen, benadrukken onze bijdragen het belang van effectieve probleemformuleringen. Hoofdstukken 2, 3 en 4 demonstreren hoe efficiënte oplossingen kunnen voortvloeien uit formuleringen die rekening houden met onzekerheid, complexiteit met oplossingskwaliteit in balans brengen, schaalbaarheid bevorderen, en aansluiten bij menselijke intuïtie. Deze resultaten benadrukken de impact van goed doordachte probleemformulering bij optimalisatie onder onzekerheid en kunnen verder onderzoek stimuleren naar optimalisatietechnieken die gebruik maken van op maat gemaakte formuleringen voor complexe, onzekere omgevingen in diverse toepassingen.

De eerste bijdrage behandelt onvolledige observeerbaarheid in de context van het laden van elektrische voertuigen (EV's) in DC-microgrids, waarbij EV-locaties alleen op kabelniveau bekend zijn. Dit scenario, dat kan ontstaan door privacybezwaren of communicatiebeperkingen, compliceert de optimalisatie van energiestromen binnen het netwerk. Traditionele benaderingen zouden vereisen dat elke mogelijke toewijzing van EV's aan laadlocaties expliciet wordt gemodelleerd, wat snel leidt totonschaalbaarheid. In Hoofdstuk 2 stellen we surrogaat-herformuleringen voor die de netwerkconfiguratie herdefiniëren door kabels als enkelvoudige knopenof parallelle knoopstructuren te behandelen, resulterend in een onzekerheid-agnostische formulering. Op gesimuleerde microgrids demonstreert deze aanpak hoe surrogaatmodellen effectief met aleatorische onzekerheid kunnen omgaan terwijl computationele efficiëntie en oplossingskwaliteit in balans worden gehouden. In toekomstig werk kunnen vergelijkbare technieken worden toegepast op andere problemen in energiesystemen en andere systemen waar onvolledige observeerbaarheid ontstaat door communicatiebeperkingen.

De tweede hoofdijdrage ligt op het gebied van beslissingsgericht learning (DFL), waarbij modellen worden getraind om onbekende parameters in optimalisatieproblemen met beperkingen te voorspellen. De essentie van de beslissingsgerichte aanpak is het optimaliseren van de beslissingskwaliteit in plaats van het minimaliseren van voorspellingsfouten

als trainingsdoelstelling. In Hoofdstuk 3 demonstreren we, zowel theoretisch als experimenteel, dat DFL toegepast op convexe optimalisatieproblemen het nul-gradiënt probleem ondervindt – een fenomeen dat voorheen alleen in het lineaire geval was geïdentificeerd. Door twee herformuleringen te introduceren – een kwadratische benadering en een lokale smoothing-techniek – verzachten we dit probleem, waardoor stabiele gradiëntpropagatie en, als gevolg daarvan, verbeterde leerprestaties mogelijk worden. Deze vooruitgang verbreedt de toepasbaarheid van beslissingsgericht learning voor convexe optimalisatie.

De laatste bijdrage introduceert het max-reward paradigma in reinforcement learning wat met name geschikt is voor doelgerichte taken waar het optimaliseren van cumulatieve beloningen kan leiden tot convergeren naar suboptimaal gedrag. Door de doelfunctie te veranderen van een cumulatieve naar een maximale beloning, sluit de voorgestelde aanpak beter aan bij de menselijke interpretatie van de taak en biedt het opmerkelijke algoritmische voordelen. Experimentele resultaten demonstreren de effectiviteit van deze aanpak in doelgerichte problemen, wat verder onderzoek naar niet-standaard beloningsformuleringen in reinforcement learning motiveert.

Door al deze bijdragen illustreert dit proefschrift hoe optimalisatie onder onzekerheid in verschillende domeinen kan profiteren van op maat gemaakte probleemformuleringen die onzekerheid aanpakken zonder expliciete modellering, schaalbaarheid met oplossingskwaliteit in balans brengen, en de prestaties van leerbenaderingen verbeteren. In het algemeen draagt dit werk bij aan verschillende toepassingen van optimalisatie onder onzekerheid en benadrukt het de potentie voor verder onderzoek en de bredere toepassing van effectieve probleemformuleringen in diverse vakgebieden.

1

INTRODUCTION

The formulation of a problem is often far more essential than its solution

Albert Einstein

1

This thesis explores several topics within computer science, including constrained optimization, decision-focused learning, and reinforcement learning. Although these areas may appear distinct, they all fall under the broader umbrella of *optimization under uncertainty* – a field in computer science and mathematics with wide-ranging practical applications. The unifying theme across the contributions in this thesis is the emphasis on formulating the right problem. Specifically, the focus lies on deriving problem formulations that are both tractable and yield solutions aligned with human interpretation of the task.

In this introductory chapter, we establish the foundation for subsequent research by introducing the concept of optimization under uncertainty and presenting a framework that contextualizes the contributions of the thesis. We begin with an informal overview of the field, examining how real-world tasks are translated into optimization problems and subsequently solved. This is followed by an analysis of various types of uncertainty, their influence on optimization, and the role of learning in addressing these complexities.

To clarify these concepts, we present several practical examples drawn from diverse fields such as power systems, decision-focused learning, and natural language processing. These examples illustrate the wide-ranging challenges posed by optimization under uncertainty and reinforce the central message of this thesis: effective problem formulation is crucial to success in optimization under uncertainty. Finally, we distill the key insights from these examples into a set of research questions that guide the subsequent chapters.

1.1. OPTIMIZATION UNDER UNCERTAINTY

Optimization is fundamentally the process of determining *decisions* that maximize or minimize an *objective* function while adhering to *constraints* [1]. In practical applications, this process begins with mapping a real-world task onto these components. However, this translation is often complex, requiring a nuanced interpretation of empirical realities into precise and tractable mathematical formulations. Success in this step demands both a deep understanding of the task itself – ensuring the mathematical model aligns with human expectations – and expertise in mathematics to guarantee the problem's solvability. The challenge lies in bridging the gap between the inherently ambiguous nature of real-world scenarios and the rigorously precise language of mathematics.

Once the optimization problem is formulated, the next step involves selecting a suitable solution method. This decision is guided by the problem's structure, scale, and specific requirements. Common approaches include constrained optimization solvers, which are effective for well-defined mathematical problems, data-driven methods [2], which

can adapt to complex or dynamic environments, evolutionary algorithms [3], which excel when other methods struggle, e.g., with highly complex or non-convex problems, and many others. Each approach offers distinct advantages and limitations, and the selection often involves trade-offs between solution quality, computational efficiency, and problem-specific needs.

The introduction of uncertainty into optimization problems significantly complicates both their formulation and solution. Uncertainty challenges the precise definition of objectives and constraints, as these may depend on unknown factors. It also complicates the solution process, as traditional deterministic methods may become inapplicable. Furthermore, uncertainty introduces substantial computational complexity, requiring consideration of a vast (potentially infinite) set of possible outcomes. Addressing uncertainty thus demands efficient methods for both modeling and solving such problems. Given that the nature and degree of uncertainty can vary significantly across different problems, a deeper understanding of its properties is essential. In the following sections, we explore various types of uncertainty and their implications for optimization.

1.1.1. ALEATORIC AND EPISTEMIC UNCERTAINTY

Uncertainty significantly increases the complexity of optimization problems. To effectively address it, we must first better understand its nature. In optimization, the term "uncertainty" refers broadly to unknown elements, though the reasons for this lack of knowledge can vary. For example, an unknown element may be stochastic, like a random variable with a known or unknown probability distribution. Alternatively, it might be deterministic but simply unknown to us. Based on this distinction, uncertainty is typically categorized into two types: *aleatoric* and *epistemic* [4, 5].

Aleatoric uncertainty stems from the inherent randomness of a phenomenon – such as the result of a coin toss – making it irreducible. In contrast, epistemic uncertainty arises from incomplete knowledge about the system being modeled, such as uncertainties in scientific measurements or model parameters. The key distinction between these two types is that epistemic uncertainty can, in principle, be reduced by gathering more data, while aleatoric uncertainty is a fundamental property of the system and cannot be reduced.

In practice, the boundary between aleatoric and epistemic uncertainty is often unclear [5–7]. When data is limited, distinguishing whether observed variability results from inherent randomness or insufficient information becomes challenging. Moreover, this distinction may depend on the context and the modeler’s perspective, introducing an element of subjectivity. Consider weather forecasting as an illustrative

example. From one perspective, weather is inherently stochastic, with statistical models used to address aleatoric uncertainty. However, one could argue that our perception of weather as stochastic largely stems from limitations in our meteorological models and the availability of high-resolution data, thus indicating epistemic uncertainty [8].

Another example appears in partially observable systems, where decision-makers can only observe a portion of the system's state. In some cases, the unobserved part may be treated as epistemic uncertainty, suggesting it could be learned or inferred from available data. Alternatively, the entire state may be treated as aleatoric uncertainty, functioning as a random variable conditioned on the observable parts [7].

While the distinction between aleatoric and epistemic uncertainty proves useful in certain cases, it is not always straightforward to make. Therefore, it can be more beneficial to adopt an application-oriented view and classify uncertainty based on its impact on the optimization problem.

1.1.2. MANIFESTATIONS OF UNCERTAINTY

Uncertainty – whether aleatoric or epistemic – can manifest in optimization problems in various ways. To better understand its influence, we categorize uncertainty based on how it appears in the problem. Below is a brief overview of common forms of uncertainty [7]:

- *Unknown model or parts of the model:* We may lack a complete understanding of the system's model, which often reflects epistemic uncertainty. However, it can also be viewed as aleatoric if the system is considered fundamentally unknowable.
- *Parametric uncertainty:* Even with knowledge of the system's general model, we may not know its exact parameters, which often reflects epistemic uncertainty.
- *Partial observability:* We may only have access to a subset of the system's state. Depending on the context, this can be treated as either epistemic or aleatoric uncertainty.
- *Innate stochasticity:* The system itself may exhibit inherent stochastic behavior, leading to uncertainty in its outcomes. This is a classic case of aleatoric uncertainty.

In complex systems, these different manifestations of uncertainty often overlap. For instance, a model might display innate stochasticity while also having unknown parameters. Consequently, uncertainties related to the model, its parameters, the degree of observability, and the system's inherent randomness are frequently intertwined. Addressing

these interconnected uncertainties collectively during the modeling process is essential.

As highlighted, one of the key challenges in optimization under uncertainty is finding an effective and tractable way to model the uncertainty. A powerful tool that often simplifies this process is *learning* from past experiences and data. Below, we provide a high-level overview of how learning can be leveraged to improve optimization under uncertainty.

1.1.3. UNCERTAINTY AND LEARNING

Building on our discussion of uncertainty in optimization, it is important to highlight the role of learning as an important tool for addressing these challenges [9]. In its broadest sense, learning involves using past experiences and data to update our understanding of a problem. In the context of optimization under uncertainty, learning is often pivotal, as we discuss below.

One common scenario for applying learning arises in the presence of epistemic uncertainty. Since epistemic uncertainty can be reduced through the acquisition of new knowledge, learning naturally offers a solution. A straightforward example is collecting noisy measurements of a physical parameter, such as weighing an object multiple times on imprecise scales. While the true value of the object's weight is deterministic, gathering more data allows us to generate increasingly accurate estimates.

Even in situations where aleatoric uncertainty dominates – where randomness is inherent – learning remains valuable. Instead of modeling the random variable directly, learning methods can be employed to approximate its effects on quantities of interest, such as the objective function. This approach simplifies the handling of pure randomness, enabling us to concentrate on the most relevant aspects of the problem and reducing computational complexity. In some cases, learning can transform what might otherwise be an intractable problem into one that is solvable. A typical example of this is found in reinforcement learning, where the transition function is often bypassed in favor of learning value functions that guide policy updates.

In sum, learning is an important tool for managing uncertainty in optimization. By leveraging available data, it can mitigate the effects of uncertainty, leading to more effective solutions. However, learning itself is not without challenges. Issues such as non-stationary distributions, generalization, low-quality data, and scaling to high-dimensional datasets must be carefully addressed to ensure success.

Having now covered the core concepts of optimization under uncertainty, we turn to real-world examples that illustrate these ideas and explore the practical challenges involved in solving such problems.

1

1.2. EXAMPLES

We have discussed the main components and challenges of optimization under uncertainty. In this section, we provide a series of examples that illustrate these challenges. Through these examples, we also demonstrate the importance of "formulating the right problem" – a theme that unifies the contributions of this thesis.

1.2.1. AC-OPF

We begin with a classical example from the field of power systems control: the *AC Optimal Power Flow (AC-OPF) problem* [10]. This problem focuses on optimizing power flows in AC grids. The primary objective of AC-OPF is to determine the optimal configuration of power, voltage, and current within the grid to maximize a specified financial goal. This must be achieved while adhering to safety standards, respecting grid capacity constraints, and complying with the physical laws that govern electricity transmission and distribution.

The main challenge in solving AC-OPF stems from its computational complexity. The non-convex nature of the physical laws governing power flows makes the problem particularly demanding. Additionally, in real-world applications, uncertain factors such as fluctuating demand, variability in renewable energy generation, and changes in generation costs further complicate the problem, often rendering it intractable. A common approach to address this challenge is to simplify the power flow equations using the *DC-OPF* model [11]. This reformulation is based on several key assumptions, including flat voltage magnitudes, negligible reactive power, and minimal power losses, which result in a linear approximation of the power flow equations. These simplifications reduce the problem's complexity substantially, making it manageable and allowing for explicit modeling of uncertainty. Although the DC-OPF model introduces some inaccuracies, it is particularly effective for high-voltage power grids where the assumptions generally hold. In such cases, DC-OPF provides a computationally efficient approximation that balances solution quality with feasibility.

The DC-OPF approach illustrates how problem reformulation can play an important role in optimization under uncertainty. In this case, the reformulation leverages expert knowledge of power systems to simplify the problem while maintaining practical relevance. This highlights the importance of domain-specific expertise in creating effective problem formulations.

The idea of problem reformulation is not limited to power systems. It extends to other domains where the underlying model may be less well understood. In such cases, even without deep domain expertise, innovative approaches to problem formulation can lead to significantly improved solutions. In our next discussion, we explore decision-focused

learning (DFL) — a framework where the model is only partially known, and learning is used to fill in the missing parts.

1.2.2. SMART "PREDICT, THEN OPTIMIZE"

Consider a linear programming (LP) optimization problem where some parameters of the objective function are unknown. Instead of having direct access to these parameters, we are provided with a feature vector that contains information about them. A common example of such a problem can be found in navigation apps, where the app needs to compute the shortest path to the location specified by the user. In this case, the unknown parameters correspond to travel times on various route segments, traffic conditions, or public transport delays. To manage this uncertainty, a two-stage approach is commonly used. First, a machine learning model predicts the missing parameters based on the feature vector. Second, these predicted parameters are fed into a constrained optimization solver to make decisions. The optimization under uncertainty problem corresponding to this setup is the one of training a predictive model.

The standard approach is to train the machine learning model to predict the uncertain parameters by minimizing prediction error – a typical supervised learning task. However, this approach has a significant drawback: the true objective should be decision quality, not prediction accuracy. In the shortest path example, quality of the traffic condition prediction is irrelevant for the user, unlike the quality of the provided route. Therefore, optimizing prediction accuracy only approximates the real task, sacrificing solution quality for computational simplicity.

Decision-focused learning, also referred to as "*predict, then optimize*", addresses this issue by directly optimizing for decision quality rather than prediction accuracy [12]. The main challenge is that using decision quality as a training objective usually requires differentiating the LP solution with respect to the predicted parameters, which is computationally challenging. The *smart "predict, then optimize" (SPO)* approach [12] overcomes this by introducing a differentiable approximation of decision quality, making it feasible to use as a learning objective. Empirical results demonstrate that this method significantly outperforms standard approaches that focus purely on prediction accuracy.

The SPO approach underscores the importance of aligning problem formulation with the true objective. By focusing on decision quality rather than prediction accuracy, it ensures that the solution directly addresses what truly matters, leading to better outcomes. Furthermore, the use of differentiable approximations enables SPO to transform a complex problem into a tractable one, highlighting how thoughtful

reformulation can unlock new potential in machine learning applications.

In the final example, we explore natural language processing (NLP) through the lens of optimization under uncertainty, demonstrating how this complex field has benefited from simple yet powerful formulations like next-token prediction and reinforcement learning from human feedback.

1.2.3. NATURAL LANGUAGE PROCESSING

Consider the task of creating a "general conversational AI" -— an AI system capable of engaging in open-ended dialogue across a broad range of topics. This is an extraordinarily complex optimization problem under uncertainty, characterized by numerous, multifaceted uncertainties:

- Epistemic uncertainty:
  - Incomplete understanding of human conversation patterns and language nuances.
  - Partial observability of the user's intent and expectations for a response.
- Aleatoric uncertainty:
  - Inherent variability in human conversations, with multiple valid responses possible for any given input.
  - Stochastic nature of user satisfaction, which serves as the objective, due to the somewhat chaotic aspects of human behavior.

Given these complexities, formulating the right optimization problem is crucial. One might initially think that creating such an AI would require directly modeling the intricacies of human conversation —- a challenging task given the uncertainties involved [13]. However, a surprisingly effective formulation has emerged: framing the problem as *next-token prediction* [14]. In this approach, the AI's task is simply to predict the next token (word or subword) in a sequence, given all previous tokens. This formulation creates a clear optimization objective: minimizing the cross-entropy loss between the model's predictions and the actual next tokens in the training data. This objective is both well-defined and computationally tractable, despite the underlying uncertainties in language and conversation. Moreover, it allows utilizing vast amounts of text data, much of which represents human-written conversations or knowledge.

This seemingly simplistic formulation has led to AI models with astonishing levels of performance in conversation. LLMs trained on next-token prediction, such as GPT-3 [15] and its successors,

have demonstrated an impressive ability to engage in human-like dialogue, answer questions, and even perform complex reasoning tasks. Moreover, the development of these models introduced another innovative problem formulation: *Reinforcement Learning from Human Feedback (RLHF)* [16]. RLHF addresses some of the limitations of pure next-token prediction by adding an additional optimization problem focused on alignment with human feedback. It can be seen as a method to reduce uncertainty about human preferences and values, by directly incorporating human feedback into the optimization process. This approach allows for fine-tuning the model to better align with human preferences, addressing issues like safety, coherence, and helpfulness that are not fully captured by the next-token prediction objective alone.

The combination of two key problem formulations – next-token prediction and RLHF – has led to AI models with broad capabilities that align more closely with human values and expectations. Even though this success was largely enabled by vast amounts of available data, increased computational power, and the development of the transformer architecture [17], we find that the essential component in the remarkable success of LLMs is the problem formulation itself.

Unlike the AC-OPF problem, which relies on expert knowledge for reformulation, and SPO, which combines expert knowledge of linear programming with machine learning, LLMs do not utilize any specific knowledge of the underlying problem structure. Instead, LLMs introduce an objective function that aligns with desired outcomes, resulting in a conceptually simple optimization problem that alleviates the need for explicit modeling of uncertainty.

The three examples presented in this section derive problem reformulations differently: AC-OPF employs a domain-specific approach, SPO integrates domain expertise with well-defined learning objectives, and LLMs rely purely on aligning learning methodologies with the task at hand. These examples suggest that effective problem formulations can emerge through different approaches and have a broad impact across diverse fields. In the last section of this introduction, we generalize the challenges inherent to the "formulating the right problem" paradigm and present the central research questions that this thesis aims to address.

1.3. FORMULATING THE RIGHT PROBLEM

The examples above illustrate an important principle in optimization under uncertainty: the formulation of the problem itself can be as crucial as the methods used to solve it. This principle, which we term "formulating the right problem" is a central theme of this thesis and has broad implications for tackling complex real-world challenges. Hence, we are aiming to contribute towards answering the following research question:

In optimization under uncertainty, what constitutes a "good" problem formulation, how to obtain it, and what challenges can it resolve?

As this question is too broad to be tackled within a thesis, we derive more specific research questions that we aim to address with the individual chapters:

1. *Can we address aleatoric partial observability by developing a problem formulation that is independent of it?*
   - We address this question in Chapter 2, where we study the electric vehicle charging problem (EVCP) with partially observable locations of the EVs. We propose a surrogate problem formulation that is indifferent to partial observability, thereby resulting in a better solution, while avoiding modeling uncertainty explicitly.
2. *How can we leverage both data and knowledge of the problem structure to create better formulations?*
   - We address this question in Chapter 3 about decision-focused learning (DFL). There, a predictive model is trained to predict unknown parameters of an optimization problem. Crucially, the models in the DFL paradigm are trained to optimize decision quality, rather than prediction accuracy, as this is the actual task goal. However, using this objective naively has certain computational challenges, which we address in this chapter.
3. *How can we align the desired properties of the solution with the mathematical characteristics of the problem formulation?*
   - We address this question in Chapter 4. There, we introduce max-reward reinforcement learning – a novel paradigm where an agent optimizes for the maximum rather than the cumulative reward. Both experimentally and theoretically, we demonstrate that this is a better objective in goal-reaching problems.

These research questions can be connected by viewing them as a spectrum. First, in Chapter 2, we explore an optimization problem with a known model and partially unknown parameters, where no data is available to reduce uncertainty. In this setup, neither modeling uncertainty nor learning or sampling methods are feasible, so we must address uncertainty exclusively by manually derived reformulations of the optimization problem. A natural progression from this is decision-focused learning, which extends the problem by assuming the availability of data. There, we aim at obtaining a “good” formulation for the problem of training an ML model, such that it is aware of the underlying problem structure, while being computationally tractable. In Chapter 3, we investigate and address the drawbacks of the

naive formulation of the training problem that cause convergence to suboptimal solutions. In the final contribution in Chapter 4, we move further by considering a typical reinforcement learning setup, where we have no prior knowledge of the problem beyond the task description, but can gather data through interaction. We demonstrate how the task objective – "reach the goal" – can be translated into a meaningful optimization objective —- "maximize the reward".

In summary, this thesis focuses on solving various optimization under uncertainty problems through effective problem formulations, utilizing different approaches depending on the context. From well-defined models to machine learning and reinforcement learning frameworks, the work demonstrates how the nature of the available information – whether no data, partial data, or data gathered through interaction – impacts the formulation of the problem. By presenting concrete examples from various domains, the thesis shows how carefully designed problem formulations are key to navigating uncertainty and achieving computationally feasible, high-quality solutions.

1

REFERENCES

[1] J. Nocedal and S. J. Wright. *Numerical optimization*. Springer, 1999.

[2] J. Kotary, F. Fioretto, P. Van Hentenryck, and B. Wilder. "End-to-end constrained optimization learning: A survey". In: *arXiv preprint arXiv:2103.16378* (2021).

[3] R. Datta and K. Deb. *Evolutionary constrained optimization*. Springer, 2014.

[4] S. C. Hora. "Aleatory and epistemic uncertainty in probability elicitation with an example from hazardous waste management". In: *Reliability Engineering & System Safety* 54.2-3 (1996), pp. 217–223.

[5] C. R. Fox and G. Ülkümen. "Distinguishing two dimensions of uncertainty". In: *Fox, Craig R. and Gülden Ülkümen (2011),"Distinguishing Two Dimensions of Uncertainty," in Essays in Judgment and Decision Making, Brun, W., Kirkebøen, G. and Montgomery, H., eds. Oslo: Universitetsforlaget* (2011).

[6] M. Valdenegro-Toro and D. Saromo. *A Deeper Look into Aleatoric and Epistemic Uncertainty Disentanglement*. 2022. arXiv: 2204.09308 [cs.LG]. url: https://arxiv.org/abs/2204.09308.

[7] A. Der Kiureghian and O. Ditlevsen. "Aleatory or epistemic? Does it matter?" In: *Structural safety* 31.2 (2009), pp. 105–112.

[8] W. Gong, H. V. Gupta, D. Yang, K. Sricharan, and A. O. Hero III. "Estimating epistemic and aleatory uncertainties during hydrologic modeling: An information theoretic approach". In: *Water resources research* 49.4 (2013), pp. 2253–2273.

[9] C. Ning and F. You. "Optimization under uncertainty in the era of big data and deep learning: When machine learning meets mathematical programming". In: *Computers & Chemical Engineering* 125 (2019), pp. 434–448.

[10] M. B. Cain, R. P. O'neill, A. Castillo, *et al.* "History of optimal power flow and formulations". In: *Federal Energy Regulatory Commission* 1 (2012), pp. 1–36.

[11] B. Stott, J. Jardim, and O. Alsaç. "DC power flow revisited". In: *IEEE Transactions on Power Systems* 24.3 (2009), pp. 1290–1300.

[12] A. N. Elmachtoub and P. Grigas. *Smart "Predict, then Optimize"*. 2020. arXiv: 1710.08005 [math.OC]. url: https://arxiv.org/abs/1710.08005.

[13] N. Chomsky. "The logical structure of linguistic theory". In: (1955).

[14] Y. Bengio, R. Ducharme, and P. Vincent. “A Neural Probabilistic Language Model”. In: *Advances in Neural Information Processing Systems*. Ed. by T. Leen, T. Dietterich, and V. Tresp. Vol. 13. MIT Press, 2000.

[15] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. M. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei. *Language Models are Few-Shot Learners*. 2020. arXiv: 2005.14165 [cs.CL]. url: https://arxiv.org/abs/2005.14165.

[16] N. Stiennon, L. Ouyang, J. Wu, D. M. Ziegler, R. Lowe, C. Voss, A. Radford, D. Amodei, and P. Christiano. *Learning to summarize from human feedback*. 2022. arXiv: 2009.01325 [cs.CL]. url: https://arxiv.org/abs/2009.01325.

[17] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. “Attention is All you Need”. In: *Advances in Neural Information Processing Systems*. Ed. by I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett. Vol. 30. Curran Associates, Inc., 2017.

2

EV CHARGING IN DC MICROGRIDS WITH PARTIAL OBSERVATIONS

As electric vehicles (EVs) become increasingly common, it is essential to develop infrastructure that ensures reliable, accessible, and timely charging. On the electrical engineering side, one promising direction is the use of direct current (DC) microgrids, which are inherently better suited for EVs. On the algorithmic side, new methods are needed to coordinate EV charging in a way that accounts for uncertainty while maintaining grid safety and stability.

This chapter addresses the challenge of EV charging scheduling in DC microgrids under incomplete information. In particular, we study the case where an EV's charging location within the microgrid is only partially known – a situation that can arise in practice due to limited communication, system delays, or privacy constraints. This partial observability is critical because physical grid limits depend on the precise locations of power loads and lacking this knowledge can result in grid overloads and unmet charging demand from EVs.

To tackle this, we propose two reformulations of the scheduling problem that enable robust planning despite incomplete information. Simulations show that these models successfully prevent grid violations, especially when resources are tight. Overall, the results reveal a clear trade-off between speed and performance – and point to a practical way forward for making EV charging more resilient in the real world.

This chapter is based on the article: *Veviurko, G., Böhmer, W., Mackay, L., de Weerdt, M. (2022). Surrogate DC Microgrid Models for Optimization of Charging Electric Vehicles under Partial Observability. Energies, 15(4), 1389*. Compared to the journal article, Section 2.3 was significantly updated to improve clarity.

ABSTRACT

Many electric vehicles (EVs) are using today's distribution grids, and their flexibility can be highly beneficial for the grid operators. This flexibility can be best exploited by DC power networks, as they allow charging and discharging without extra power electronics and transformation losses. From the grid control perspective, algorithms for planning EV charging are necessary. This paper studies the problem of EV charging planning under limited grid capacity and extends it to the partially observable case. We demonstrate how limited information about the EV locations in a grid may disrupt the operation planning in DC grids with tight constraints. We introduce two methods to change the grid topology such that partial observability of the EV locations is resolved. The suggested models are evaluated on the IEEE 16 bus system and multiple randomly generated grids with varying capacities. The experiments show that these methods efficiently solve the partially observable EV charging planning problem and offer a trade-off between computational time and performance. The code is made public [1].

2.1. INTRODUCTION

The increasing penetration of electric vehicles (EVs) and renewable energy sources (RES) into distribution grids provides new opportunities, but also poses new challenges for the system operators. As EVs and some of the RES inherently use DC, they can be integrated into DC grids without additional converters and with smaller power losses [2]. For this reason, DC in distribution grids is considered to be a suitable alternative to the currently used AC [3]. Unlike conventional loads, EVs are flexible in terms of when their demand should be served. This flexibility is provided by a special entity, the EV aggregator [4], which coordinates the charging of EVs. The goal of the aggregator is to solve the EV charging planning (EVCP) problem, i.e., to minimize the sum of unmet demand and operation costs, while not violating the physical constraints of the grid.

In practice, there are two main challenges related to solving the EVCP problem. First, information about the EV parameters (e.g., arrival and departure times, demand and locations in the grid) and/or the grid state (topology, physical properties of the nodes or lines) may not be fully available to the EV aggregator. Hence, the aggregator might have to operate using incomplete information. Second, as the relation between current and voltage is hyperbolic, the power flow equation makes the EVCP problem non-convex [5]. That means not only that there is no theoretical guarantee of obtaining the globally optimal solution, but also that the problem may become computationally intractable for large grids and long planning horizons.

A popular perspective on the EVCP problem is to simplify or even

omit the power flow equations and grid constraints, in order to make the problem tractable. Various studies have pursued this approach and focused on dealing with the uncertainty of the future EV arrivals. Sadeghianpourhamami, Deleu, and Develder [6] applied a reinforcement learning algorithm to coordinate the EV charging in a constraint-free grid. Yang *et al.* [7] minimized the problem's Lagrangian in a decentralized fashion to optimize the charging of the EVs located in a single building with stochastic wind power supply. Wu and Sioshansi [8] designed a two-stage optimization algorithm with a sample-average approximation technique to solve the EVCP problem using a linearization of the AC power flow. On the other hand, some studies have combined the EVCP with the non-convex optimal power flow (OPF) problem and included the exact power flow equations and grid constraints in it. Kayacık, Kocuk, and Yüksel [9] formulated the EVCP problem as a multi-timestep OPF problem extended with additional constraints on the CO_2 emissions and derived a convex second-order cone programming relaxation that can be solved efficiently. In the work by Azizipanah-Abarghooee *et al.* [10], a fuzzy-logic controller was developed to minimize power costs and emissions simultaneously. Chen, Quek, and Tan [11] solved the EVCP+OPF problem by decoupling the OPF part from the planning and using the convex dual problem for it. However, no studies have included both incomplete information or EV uncertainties and power flow equations in the EVCP problem.

While the uncertainty of the EV parameters and non-convexity of the OPF are usually considered separately in the existing literature, in reality the EV aggregator should solve both problems simultaneously. The direct combination of the existing solutions for these problems does not seem possible: two-stage optimization of a non-convex problem easily becomes intractable. On the other hand, model-free methods, such as reinforcement learning, are generally hard to apply to constraint optimization problems. Moreover, if the present, rather than the future, is not fully known (e.g., due to privacy concerns), the standard EVCP formulation becomes irrelevant, and further research is required to formulate and solve the partially observable version of the problem.

An important input parameters for the EVCP problem are the locations of the EVs in the grid, as they are necessary to compute the power flow required for charging. This study investigates the EVCP with partial observability of the EV locations. The problem is reformulated to account for partial observability, and we investigate how it affects the EV aggregator by answering the following questions:

- How crucial is it for the EV aggregator to know exact locations of the EVs in the grid?
- How can one obtain a well-performing, scalable solution when the EV locations are known only up to certain degree?

In order to answer the first question, we conducted experiments on DC grids with various topologies and capacities to evaluate how different degrees of awareness about EV locations affect the planning. The experimental results demonstrate that a lack of information about the locations of the EVs currently present in the grid considerably disrupts the performance, whereas locations of the EVs that are yet to arrive are much less vital. To deal with this issue, this paper introduces two alternatives for modeling the grid topology without knowing the exact locations of the EVs. Experiments on both real and randomly generated grids show that the suggested models perform better than a naive baseline and offer a trade-off between computational cost and performance.

2.2. BACKGROUND

2.2.1. EVCP PROBLEM FORMULATION

Existing studies consider various formulations of the EVCP problem depending on the optimization objective, EV charging approach, and optimization approach [4]. This study adopts the perspective of an EV aggregator that represents the combined interest of all the EVs in a grid. Its goal is to maximize *the social welfare*—the sum of the demand provided to the EVs—with minimal operation costs. In line with earlier work [11, 12], EV charging optimization is combined with power flow equations and grid constraints. Unlike most existing studies, this work does not assume that all vehicles can be charged fully. To account for that, all EVs have a linear utility function quantifying how much value each car assigns to being charged for one Watt Hour.

Advances in power electronics allow for new solutions in the microgrid design. Various aspects of DC microgrids' control, protection, and energy management are being studied in the literature (e.g., see the work by Kaur *et al.* [13] for an overview). As this paper aims at solving the charging planning problem with partial observability, we employ the relatively simple DC microgrid model suggested by Li *et al.* [14]. For the same reason, this work does not consider the properties of the EV batteries, as they do not affect the charging planning algorithm. This study considers DC microgrids that consist of multiple EV charging stations and one or several generators that can represent connections to an external power grid or distributed generators (DGs). Prior to defining the optimization problem for the EVCP in DC grids, we introduce the necessary notation.

The set $\mathcal{N} = \mathcal{L} \sqcup \mathcal{G}$ of nodes in the grid (the squared cup symbol $\sqcup$ is the disjoint union operator) is a disjoint union of the loads $\mathcal{L}$ and generators $\mathcal{G}$. For convenience, we use n and m as subscripts when we mean an arbitrary node and use subscripts l and g to highlight the difference between the loads and generators. The set of lines $\mathcal{E} \subset \mathcal{N} \times \mathcal{N}$

contains the node pairs (n, m) that are connected by a line. The set $\{t_0, t_1, \ldots, t_T\}$ is the set of planning timesteps, each timestep having a constant length, being defined as $\Delta t := t_{s+1} - t_s, \forall s \in \{0, \ldots, T-1\}$. At each timestep t, voltage and power in the loads l and generators g are denoted by v_l^t, p_l^t and v_g^t, p_g^t correspondingly. Line current and conductance between nodes n and m are denoted by i_{nm}^t and y_{nm}, respectively. For each generator $g \in \mathcal{G}$, we define energy supply costs as a linear function of the generator's power with coefficient c_g^t. We use a convention that power at the generators is negative and power at the loads is positive. The bounds for the voltages, power, and currents are denoted by $\bar{V}_n, \underline{V}_n, \bar{P}_n, \underline{P}_n, \bar{I}_{nm}$. The set of the electric vehicles is denoted by $\mathcal{K}$, and each EV $k \in \mathcal{K}$ has several parameters. Let t_k^{arr}, t_k^{dep} be the arrival and departure times, $\bar{E}_k$ be the desired state-of-charge (SOC) at departure, and u_k be the coefficient of the linear utility function specifying the priority of the particular EV. SOC at timestep t is denoted by e_k^t. For simplicity, we use $l(k)$ to denote the load where EV k is being charged and $k(l)$ for the EV at load l.

First, the power flow constraints are defined as follows:

$$\underline{V}_l \le v_l^t \le \bar{V}_l, \quad \underline{P}_l \le p_l^t \le \bar{P}_l, \quad \forall l \in \mathcal{L} \tag{2.1a}$$

$$\underline{V}_g \le v_g^t \le \bar{V}_g, \quad \underline{P}_g \le p_g^t \le \bar{P}_g, \quad \forall\, g \in \mathcal{G} \tag{2.1b}$$

$$-\bar{I}_{nm} \le i_{nm}^t \le \bar{I}_{nm}, \quad \forall (n, m) \in \mathcal{E} \tag{2.1c}$$

$$i_{nm}^t = (v_n^t - v_m^t) y_{nm}, \quad \forall (n, m) \in \mathcal{E} \tag{2.1d}$$

$$p_n^t = -v_n^t \sum_{\{m | (n,m) \in \mathcal{E}\}} y_{nm}(v_n^t - v_m^t), \quad \forall n \in \mathcal{N} \tag{2.1e}$$

Then, the EV state-of-charge is subject to the following constraints:

$$e_k^{t_k^{arr}} = 0, \quad \forall k \in \mathcal{K} \tag{2.2a}$$

$$0 \le e_k^t \le \bar{E}_k, \quad \forall k \in \mathcal{K}, \quad \forall t \in [t_k^{arr}, t_k^{dep}] \tag{2.2b}$$

$$e_k^{t+1} = e_k^t + \Delta t p_{l(k)}^t, \quad \forall k \in \mathcal{K}, \quad \forall t \in [t_k^{arr}, t_k^{dep}) \tag{2.2c}$$

The optimization objective of the EVCP problem is defined as

$$J(p) = \sum_t \Big(\sum_{l \in \mathcal{L}} u_{k(l)} p_l^t \Delta t + \sum_{g \in \mathcal{G}} c_g^t p_g^t \Delta t \Big). \tag{2.3}$$

We call J *the social welfare* as it combines the fulfilled demand weighted by the EV utility coefficients and the negative of the operation costs. The coefficients c_g^t can represent the generation costs or price

for buying power from the external grid. The utility coefficients u_k are defined per EV and represent the importance of each vehicle. It is worth mentioning that it is possible to define the utility coefficients per load rather than per EV. In that case, it can be used to model the pricing tariff in the grid. However, the problem will not change mathematically, and hence we do not deliberately study this case. The constraints (2.1a - 2.1e) and (2.2a - 2.2c) and the objective (2.3) define the EVCP problem:

$$\begin{aligned} \max_{v,i,p,e} \quad & J(p) \\ \text{subject to:} \quad & 2.1a - 2.1e \\ & 2.2a - 2.2c \end{aligned} \qquad \text{(Exact EVCP)}$$

2.2.2. SOCP RELAXATION

Due to Equation 2.1e, the OPF problem and hence the EVCP problem are non-convex. Several methods to convexify the OPF problem are suggested in the literature, including semi-definite programming (SDP) [5] and second-order convex programming (SOCP) [14] formulations. As the latter has been shown to perform optimally in multiple real DC grids [14], we adapt it to the EVCP problem. The SOCP relaxation can be obtained by the following change of variables:

$$\begin{cases} \hat{v}_n^t &= (v_n^t)^2 \\ p_{nm}^t &= ((v_n^t)^2 - v_n^t v_m^t) y_{nm} \\ l_{nm}^t &= y_{nm}^2 (v_n^t - v_m^t)^2 \end{cases}$$

The constraints (2.1a - 2.1e) can be relaxed to the following quadratic cone constraints:

$$\underline{V}_l^2 \le \hat{v}_l^t \le \bar{V}_l^2, \quad \underline{P}_l \le p_l^t \le \bar{P}_l, \quad \forall l \in \mathcal{L} \tag{2.4a}$$

$$\underline{V}_g^2 \le \hat{v}_g^t \le \bar{V}_g^2, \quad \underline{P}_g \le p_g^t \le \bar{P}_g, \quad \forall g \in \mathcal{G} \tag{2.4b}$$

$$-\bar{I}_{mn}^2 \le l_{mn}^t \le \bar{I}_{mn}^2, \quad \forall (m,n) \in \mathcal{E} \tag{2.4c}$$

$$\frac{l_{nm}^t}{y_{nm}} = p_{nm}^t + p_{mn}^t, \quad \forall (n,m) \in \mathcal{E} \tag{2.4d}$$

$$y_{nm}(\hat{v}_n^t - \hat{v}_m^t) = p_{nm}^t - p_{mn}^t, \quad \forall (n,m) \in \mathcal{E} \tag{2.4e}$$

$$p_n^t = \sum_{\{m|(n,m)\in\mathcal{E}\}} p_{nm}^t, \quad \forall n \in \mathcal{N} \tag{2.4f}$$

$$l_{nm}^t \hat{v}_n^t \ge (p_{nm}^t)^2, \quad \forall (n,m) \in \mathcal{E} \tag{2.4g}$$

Then, the SOCP relaxation of the EVCP problem is the following optimization problem:

$$\begin{aligned} \max_{\hat{v},l,p,e} \quad & J(p) \\ \text{subject to:} \quad & 2.4a - 2.4g \\ & 2.2a - 2.2c \end{aligned} \qquad \text{(Relaxed EVCP)}$$

2.2.3. INCLUDING UNCERTAINTIES

If the full information about the EVs and the grid is available to the EV aggregator, then the only challenge for solving the exact EVCP problem is its non-convexity. As discussed in the previous section, convex relaxation techniques such as SDP or SOCP are demonstrated to be effective solutions for that. In practice, however, the information that the aggregator has access to might be limited. For example, due to causality, parameters of an EV usually become known only after it arrives in the grid, hence making the EVCP a stochastic optimization problem. Moreover, if RES [7] or inelastic loads [11] are included in the problem, their generation and demand are usually also modeled as random variables. A common approach to solving the EVCP problem with uncertainties is *online* planning, where the problem is repeatedly solved at each timestep.

An important feature of online planning is that it allows one to obtain a feasible solution of the EVCP problem even in the presence of future uncertainties. However, a conceptually different example emerges when the state of the grid is only partially observable. For example, changes in the grid topology, variations in nodal and line limits, or locations of the EVs in the grid might be reported to the aggregator with a delay. In this case, the EVCP problem cannot be formulated in the standard way, because its constraints are unknown. Furthermore, as the solution is not guaranteed to be feasible, the objective value cannot be used to evaluate the solution's quality. The next section presents a way to define the EVCP problem with partial observability and introduce an evaluation framework that can be used with it.

2.3. PARTIALLY OBSERVABLE EVCP

In a partially observable EVCP (PO-EVCP) problem, the EV aggregator may lack knowledge of certain input parameters of the EVCP. In this study, we consider a scenario where the locations of the EVs within the grid are only partially known. Specifically, the loads in the grid are grouped into several regions, referred to as *cables*. The EV aggregator has access only to cable-level information about where the EVs are parked. In practice, these cables may correspond to charging stations

on different streets or EV parking lots. This setup is particularly relevant in cases where EV privacy is a concern or when communication between the EV aggregator and the grid operator is limited.

First, we introduce additional notation to formalize the notion of a cable mathematically. The grid loads are split into cables, $\mathcal{L} = \bigsqcup_{c \in \mathcal{C}} c$, where $\mathcal{C}$ is the set of cables, and each cable $c \in \mathcal{C}$ is a subset of loads, $c \subset \mathcal{L}$. For convenience, we write $c(l)$ to denote the cable containing load l.

For an EV $k \in \mathcal{K}$ charging at load $l(k)$, the EV aggregator observes only the cable $c(l(k))$ (also denoted as $c(k)$), not the specific load $l(k)$ itself. The latter is treated as a random variable, uniformly distributed over the unoccupied loads on that cable.

In the PO-EVCP problem, the feasibility region defined by (2.1a–2.1e) and (2.2a–2.2c) is uncertain and therefore cannot be directly used in a constrained optimization solver and hence a method to address this uncertainty is needed. The remainder of this section introduces three approaches that handle the partial observability of the problem through problem reformulations.

2.3.1. BLIND GUESSING MODEL

The simplest way to resolve the partial observability of the PO-EVCP problem is to sample the positions of incoming EV uniformly from the unoccupied loads. Note that the sampled and the actual positions will rarely coincide. Let $\vec{l}$ and $\vec{k}$ be vectors of the loads and EVs, respectively, and let $\vec{l}(\vec{k})$ be the true unknown assignment of grid loads to EVs. *Blind guessing* is defined as this random sampling of an assignment $\vec{l}'(\vec{k})$, and we denote the resulting EV-to-load and load-to-EV maps as $l'(k)$ and $k'(l)$, respectively. In the EVCP problem, only constraint 2.2c and objective 2.3 depend on $\vec{l}'(\vec{k})$. Hence, they can be rewritten as follows:

$$e_k^{t+1} = e_k^t + \Delta t p^t_{l'(k)}, \quad k \in \mathcal{K}, \quad t \in [t_k^{arr}, t_k^{dep}) \tag{2.2c'}$$

$$\hat{J}(p) = \sum_t [\sum_{l \in \mathcal{L}} u_{k'(l)} p_l^t + \sum_{g \in \mathcal{G}} c_g^t p_g^t]. \tag{2.3'}$$

Then, the PO-EVCP problem with blind guessing can be defined:

$$\begin{aligned} \max_{v,i,p,e} \quad & \hat{J}(p) \\ \text{subject to:} \quad & 2.1a - 2.1e \\ & 2.2a, 2.2b, 2.2c' \end{aligned} \tag{Guessing PO-EVCP}$$

Essentially, the *blind guessing* model resolves the uncertainty in the PO-EVCP problem by randomly assuming where EVs are parked within the corresponding cable. Then, the problem is reduced to the standard exact EVCP problem, defined in Section 2.2.1.

2.3.2. SURROGATE GRID MODELS

As an alternative to blindly guessing the EV assignments, it is possible to instead modify the grid topology used in the optimization problem. The goal is to derive a surrogate grid with such topology that the optimization problem becomes agnostic to the non-observable information. In other words, such surrogate grid should be invariant over different EV assignments, as it effectively makes the problem deterministic. This section presents two different surrogate grid models with this property. Before defining these models, some additional notation is introduced. Let $(m, n) \in \mathcal{N}$ be a pair of nodes and let $P(m, n)$ be a set of all paths between m and n without cycles. Then, let

$$p = ((m, n_1), (n_1, n_2), \ldots, (n_u, n)) \in P(m, n)$$

be one such path. Its current limit $\bar{I}_p$ and conductance y_p are defined as

$$\bar{I}_p = \min_{(a,b)\in p} \bar{I}_{ab} \quad \text{and} \quad 1/y_p = \sum_{(i,j)\in p} 1/y_{ij}.$$

Then, we can define the path with highest conductance as

$$p^* = \arg\max_{p\in P(m,n)} y_p.$$

We denote its conductance as y^{path}_{mn} and its current limit as $\bar{I}^{path}_{mn}$. Then, the conductance between a cable $c \in \mathcal{C}$ and an external node $m \notin c$ is defined as the average conductance over the paths from m to all loads in c :

$$y^{path}_{mc} := \frac{1}{|c|} \sum_{n\in c} y^{path}_{mn}.$$

The current limit between c and m is defined as

$$\bar{I}^{path}_{mc} := \min_{n\in c} \bar{I}^{path}_{mn}.$$

Essentially, y^{path}_{mc} and $\bar{I}^{path}_{mc}$ quantify how current can flow from a node m into cable c assuming the exact destination node is unknown.

Parallel nodes model. The *parallel nodes* model transforms each cable c into a set of nodes connected in parallel. In this way, the model becomes invariant to permutations of EV assignments to loads within each cable. To derive the parallel nodes surrogate grid, all lines that connect nodes within c are removed. For each connection (m, n) from an external node $m \notin c$ to $n \in c$, a new node $\hat{n}$ is added and connected to m with $\bar{I}_{m\hat{n}} = \bar{I}_{mn}$ and $y_{mn} = \infty$. Then, node $\hat{n}$ is connected to all loads in the cable $n_i \in c$ with new lines, such that $\bar{I}_{\hat{n}n_i} = \bar{I}_{mn}$ and $y_{\hat{n}n_i} = y^{path}_{mn_i}$. The

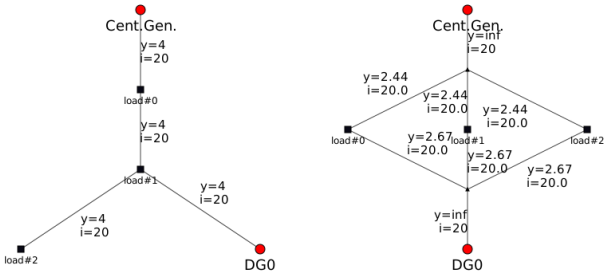


Figure 2.1: An example of the parallel nodes surrogate model. The original grid is on the left, the surrogate is on the right. Rectangles are loads, circles are generators. Line capacity is y and current limit is i.

new node $\hat{n}$ is passive; i.e., $\underline{P}_{\hat{n}} = \bar{P}_{\hat{n}} = 0$. The parallel nodes' surrogate grid is illustrated in Figure 2.1.

Let $\mathcal{N}_{par} = \mathcal{L}_{par} \sqcup \mathcal{G}_{par}$ and $\mathcal{E}_{par}$ be the sets of nodes and lines in the parallel nodes surrogate grid. Then, the optimization problem for the *parallel nodes* models is the same as the exact EVCP problem with $\mathcal{N}_{par}, \mathcal{E}_{par}$:

$$\max_{v,i,p,e} \quad \sum_t \Big(\sum_{l \in \mathcal{L}_{par}} u_{k(l)} p_l^t \Delta t + \sum_{g \in \mathcal{G}_{par}} c_g^t p_g^t \Delta t \Big)$$

subject to: 2.1a – 2.1e for $\mathcal{N}_{par}, \mathcal{E}_{par}$
2.2a – 2.2c

(Parallel PO-EVCP)

Crucially, the parallel nodes model is, by construction, independent of the exact assignment of EVs to loads within each corresponding cable. Therefore, it effectively addresses the partial observability challenge.

Single node model. The *single node* model transforms each cable c into a single node n_c that aggregates information about all nodes within c. For each neighboring node $m \notin c$, such that $\exists i \in c$, $(m, i) \in \mathcal{E}$, a line (m, n_c) is added. Its parameters are defined as $\bar{I}_{mn_c} = \bar{I}_{mc}^{path}$ and $y_{mn_c} = y_{mc}^{path}$. The power bounds for n_c are defined as the sum over original nodes: $\bar{P}_{n_c} = \sum_{n \in c} \bar{P}_n$, $\underline{P}_{n_c} = \sum_{n \in c} \underline{P}_n$. Voltage bounds are defined as the average: $\bar{V}_{n_c} = \frac{1}{|c|} \sum_{n \in c} \bar{V}_n$, $\underline{V}_{n_c} = \frac{1}{|c|} \sum_{n \in c} \underline{V}_n$. The parallel nodes' surrogate grid is illustrated in Figure 2.2.

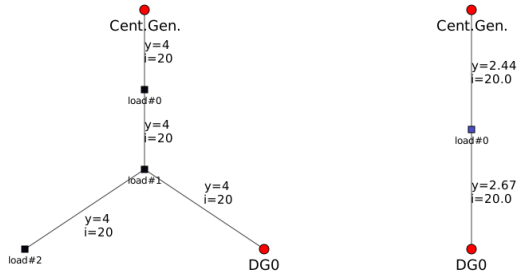


Figure 2.2: An example of a single node surrogate grid. Rectangles are loads, circles are generators. The original grid is on the left, the surrogate is on the right. Line capacity is y and current limit is i.

In the *single node* surrogate model, all nodes from a cable c are mapped to new node n_c. Hence, there are multiple EVs charging at n_c. Prior to defining the optimization problem, new notation is introduced. Let p_k^t be the power provided to EV k at timestep t. Then, there are the following constraints on p_k^t and EVs' SOC.

$$e_k^{t+1} = e_k^t + \Delta t p_k^t, \quad \forall k \in \mathcal{K}, \quad \forall t \in [t_k^{arr}, t_k^{dep}) \tag{2.5a}$$

$$p_k^t = 0, \quad \forall k \in \mathcal{K}, \quad \forall t \notin [t_k^{arr}, t_k^{dep}) \tag{2.5b}$$

$$\sum_{k \in \{k \in \mathcal{K} | c(k) = c\}} p_k^t = p_{n_c}^t, \quad \forall c \in \mathcal{C}, \quad \forall t \in [t_k^{arr}, t_k^{dep}) \tag{2.5c}$$

Let $\mathcal{N}_{sng} = \mathcal{L}_{sng} \sqcup \mathcal{G}_{sng}$ and $\mathcal{E}_{sng}$ be the sets of nodes and lines in the *single node* surrogate grid. Then, the optimization problem for the *single node* model is defined as follows:

$$\max_{v,i,p,e} \quad \sum_t \Big(\sum_{l \in \mathcal{L}_{sng}} u_{k(l)} p_l^t \Delta t + \sum_{g \in \mathcal{G}_{sng}} c_g^t p_g^t \Delta t \Big)$$

subject to: 2.1a – 2.1e for $\mathcal{N}_{sng}$, 2.2a, 2.2b, 2.5a – 2.5c (Single PO-EVCP)

Since all EVs belonging to a cable c in the original problem are charged at the same node n_c in the *single node* model, the value of the true assignment $\vec{l}(\vec{k})$ does not affect the problem. Hence, the partial observability is resolved.

2.3.3. PLANNING AND EXECUTION

Crucially, the charging schedules produced by blind guessing, single-node, and parallel-node models can violate the true grid constraints

2.1a–2.1e and 2.2a–2.2c, rendering them infeasible and thus impossible to execute. Moreover, while Li *et al.* [14] demonstrate that SOCP relaxations are often exact for real-world grids, these guarantees do not extend to scenarios with line current constraints. Therefore, solutions obtained via SOCP relaxation of the PO-EVCP problem are not guaranteed to be feasible in the actual grid.

To address this challenge, we propose a *planning and execution* framework that splits the solution process into two stages:

- The *planner*, implemented by the EV aggregator, computes an optimal charging schedule under partial observability, solving an SOCP relaxation of the PO-EVCP problem.
- The *executor*, a control algorithm embedded in the grid, has full observability and is responsible for enforcing physical constraints. It solves a single-timestep OPF problem to safely implement the planner's proposed schedule.

This division enables robust and adaptive grid operation. At each timestep t, the planner computes a proposed charging schedule based on limited information, possibly using blind guessing or a surrogate grid model. However, due to the relaxation and partial observability, the resulting plan may be infeasible. To account for this, planning and execution proceed in an online fashion: at each timestep, the planner generates a schedule, and the executor attempts to implement it while ensuring grid safety.

Let $\mathcal{K}^t = \{k \in \mathcal{K} \mid t_k^{\text{arr}} \leq t < t_k^{\text{dep}}\}$ denote the set of EVs present at time t, and let $\hat{p}_l^t$ be the planned power at each load l where an EV from $\mathcal{K}^t$ is parked. The executor then solves the following OPF problem:

$$\begin{aligned} \max_{v^t, i^t, p^t} \quad & J^t(p^t) = \sum_{l \in \mathcal{L}} u_{k(l)} p_l^t + \sum_{g \in \mathcal{G}} c_g^t p_g^t \\ \text{subject to:} \quad & \text{grid constraints } 2.1a - 2.1e \\ & p_{l(k)}^t \leq \hat{p}_k^t \quad \text{for all } k \in \mathcal{K}^t \end{aligned} \tag{Executor}$$

This problem seeks to implement the planned schedule $\hat{p}_k^t$ as closely as possible, while ensuring all physical constraints are satisfied. For example, if the planner's proposed $\hat{p}_k^t$ is feasible with respect to 2.1a–2.1e, and if user utility u_k significantly outweighs generator cost c_g, then the executor will match the planner's schedule exactly.

The complete interaction between planner and executor is summarized in Algorithm 1.

Algorithm 1 Planning and execution framework.

Initialize total social welfare $J_{ex} = 0$
Initialize EVs' SOC $e_k = 0$, for $k \in \mathcal{K}$
for $t \in \{t_0, t_1, \ldots, t_T\}$ **do**
  For active EVs $k \in \mathcal{K}^t$, obtain $\hat{p}_k^t$ by solving the planner problem
  Create the executor problem using $\hat{p}_k^t$
  Solve the executor problem, get loads power p_l^t and objective $J^t(p^t)$
  $J_{ex} \leftarrow J_{ex} + J^t(p^t)$
  for $k \in \mathcal{K}^t$ **do**
    $e_k \leftarrow e_k + \Delta t \times p_{l(k)}^t$
  end for
end for

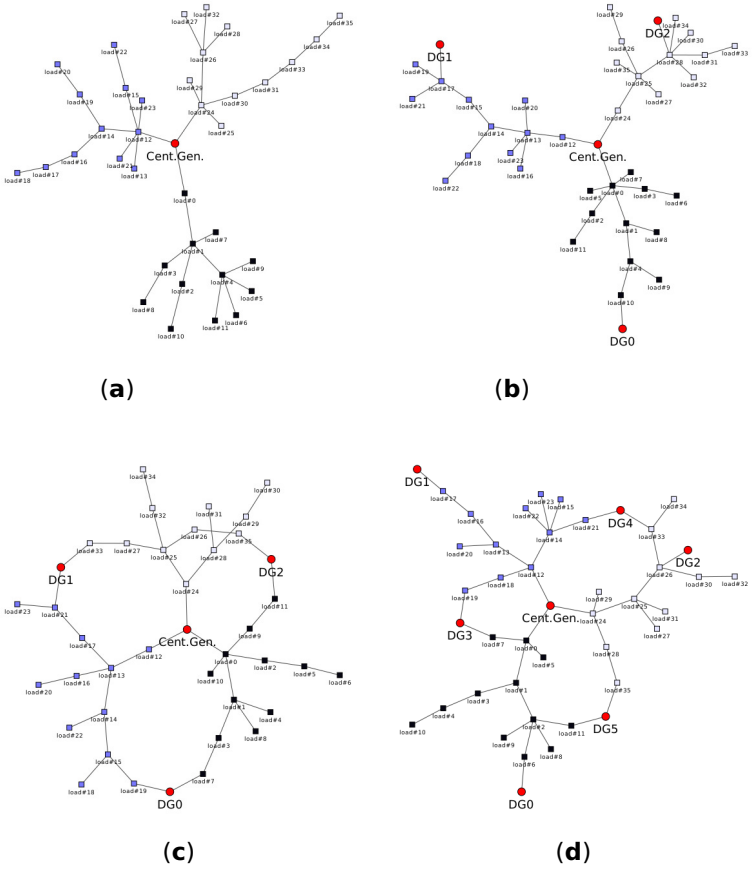
The planning and execution framework guarantees that each grid state is always feasible, and hence allows for evaluation of inexact planners, such as surrogate grid models and blind guessing planners.

2.4. EXPERIMENTS

We simulated the EVCP problem using the planning and execution framework with different planners in order to answer the two following questions:

- How important is the knowledge of the precise EV locations in different DC grids for the EV aggregator to solve the EVCP problem?
- Which approach should the EV aggregator use when EV locations are known only at the cable level?

This section describes the details of the simulations and presents the results. Two main components of each simulation are the grid configuration and the scenario. The grid is defined by the numbers of loads, generators, and lines between them; the voltage and power limits of the nodes; and the current limits and conductance of lines. In the experiments, the grid topology was either sampled from the random topology classes (Figure 2.3) or taken from the real IEEE16 grid [14] (Figure 2.4). For both meshed and radial topologies of the IEEE16 grid, we considered two cases: with and without connection to the external grid. In the former case, capacities of the feeders A, B, and C were infinite $\underline{P}_g = \infty$. In the disconnected case, the feeder capacities were set to zero $\underline{P}_g = 0$. For all nodes, the voltage limits were defined as $\underline{V}_n$ = 300 V, $\bar{V}_n$ = 400 V. The load power bounds were set to $\underline{P}_l = 0, \bar{P}_l$ = 10,000 W. The generators power upper bound was set to zero $\bar{P}_g$ = 0 W (meaning generators cannot consume power), and the lower

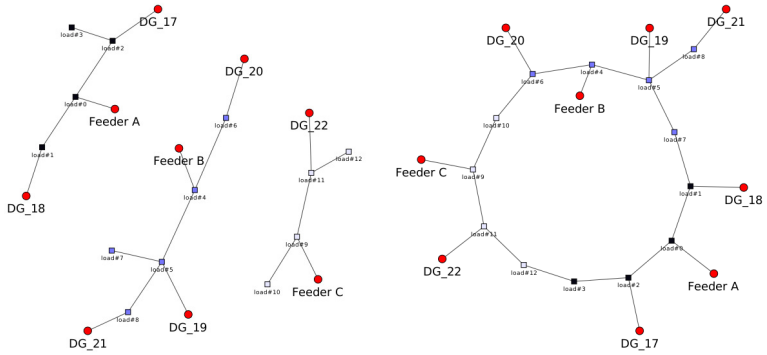


(**a**) (**b**)

(**c**) (**d**)

Figure 2.3: Random grid topologies. Rectangular nodes are loads, red circles are generators. Colors of the loads encode the cables they belong to. (**a**) Radial grid with single generator. (**b**) Radial grid with DGs. (**c**) Meshed grid with DGs between cables. (**d**) Meshed grid with DGs between and at cables.

bound $\underline{P}_g$ varied across the experiments. The line current limits $\bar{I}_{mn}$ also varied. The conductance was equal for all lines in all grids, $y_{nm} = 15$ S. We always set all generators to have the same capacity $\underline{P}_g$ and cost coefficient c_g. The line capacities $\bar{I}_{nm}$ were also equal for all lines. In the experiments with random grids, we sampled one random topology per value of $\underline{P}_l$ and $\bar{I}_{mn}$.



(**a**) (**b**)

Figure 2.4: IEEE16 grid [14]. Rectangular nodes are loads, red circles are generators. Colors of the loads encode the cables they belong to. (**a**) Radial version. (**b**) Meshed version.

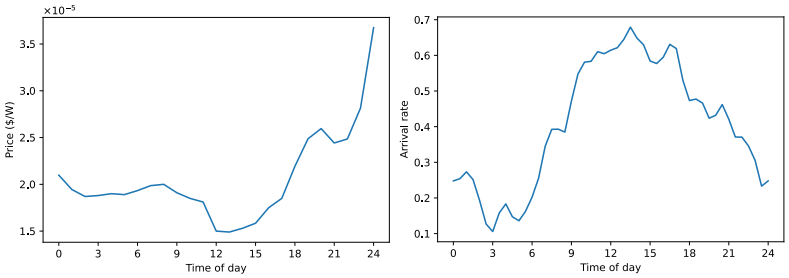
Each scenario contained a power price and EV parameters. The power price for each scenario was sampled from the day-ahead price data for the Netherlands [15]. Figure 2.5a shows an example of the power price curve for a single day. To sample the EV arrival times, we simulated a non-homogeneous Poisson process independently in each load. The arrival rate was similar across the loads and scenarios and derived from the dataset of EV charging sessions in Dundee, Scotland [16]. Figure 2.5b demonstrates the dynamics of the EV arrival rate.

The demand, parking time, and utility coefficient were sampled from the normal distributions described in Table 2.1. It is worth mentioning that we chose values for the utility coefficients c_k such that they were at least an order of magnitude larger than the power price c_k^t. In this case, the objectives in the executor and planner problems were monotonically increasing, i.e., charging an EV always increases the social welfare.

The planning horizon was set to 24 hours for all simulations,

Table 2.1: Distributions of the EV parameters.

	Distribution	**Unit**
Desired SOC $\bar{E}_k^t$	$\mathcal{N}(8500, 500)$	Watt
Charging time $t_k^{dep} - t_k^{arr}$	$\mathcal{N}(3.75, 0.5)$	Hour
Utility coefficient u_k	$\mathcal{N}(5 \times 10^{-4}, 5 \times 10^{-7})$	\$/Watt



(**a**) (**b**)

Figure 2.5: (**a**) An example of the power price curve for a single day. (**b**) Poisson process rate used to model the EV arrivals.

and timestep size to 30 min. For each grid topology and parameters, we used 6 scenarios to evaluate the performance of the planners.

The optimization routine was performed by MOSEK Fusion API for Python 9.3.10 [17] on a single machine with Intel® Core i7-10700K Processor. We used default solver parameters except for setting *basisRelTolS*, *basisTolS*, and *intpntCoTolDfeas* to 10^{-9}. The initial conditions and convergence criteria were also default for MOSEK.

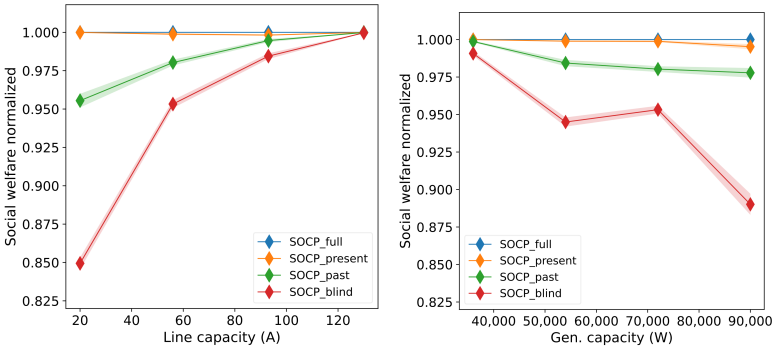
2.4.1. IMPORTANCE OF THE LOCATIONS

To estimate the importance of knowing the precise EV locations, we tested a blind guessing SOCP planner on the PO-EVCP problem using different degrees of observability of the EV locations (Table 2.2). The planners were obtaining the assignments $\vec{l}(\vec{k})$ by first fixing the precisely known locations and then randomly sampling the remaining locations within the corresponding cables.

We evaluated all four planners by sampling random meshed grids (Figure 2.3c) with 12 nodes and 4 generators and varying generation capacity $\underline{P}_g$ and the lines' current limit $\bar{I}_{mn}$. Figure 2.6 demonstrates

Table 2.2: Four degrees of observability of the EV locations.

Observability	**Precisely Known EVs**	**EVs Known to Cable**
Full	All EVs	None
Present	EVs that already arrived	EVs yet to arrive
Past	EVs that stayed for at least one timestep	EVs that have just arrived and future EVs
Blind	None	All EVs



(**a**) (**b**)

Figure 2.6: Results of *random guessing* planners with different degree of knowledge of the EV locations. Values are normalized relative to the *full* planner. Horizontal axis is the lines current limit. (**a**) Normalized social welfare over varying lines current limit, per-generator capacity is fixed at 72 kW. (**b**) Normalized social welfare over varying single generator's capacity, per-line current limit is fixed at 56 A.

the provided social welfare as a function of $\underline{P}_g$ and $\bar{I}_{mn}$. For better comparability between different topologies, the social welfare is normalized relative to the *full* planner.

The simulations imply that knowing EV locations becomes important when the line current constraints are tight. On the contrary, the tighter the generation constraints are, less effect is caused by the partial observability. Moreover, the planner with access to the present, but not the future, EV locations (labelled SOCP_present) performed almost optimally, while *past* and *blind* planners (SOCP_past and SOCP_blind, respectively) provided considerably less social welfare. Practically, that means that knowing locations of the EVs arriving in the future is not crucial and knowing currently present EVs locations is enough for the EV aggregator to compute the charging schedule. However, when the locations are unavailable even for the active EVs in the grid, the EV aggregator might want to use surrogate grid models for planning.

2.4.2. SURROGATE GRID MODELS

As demonstrated in the previous section, the blind planner performs suboptimally for tight line constraints. In this section we compare *single node* and *parallel nodes* models with the *blind* planner and demonstrate their benefits for the EV aggregator.

We used all four random topologies from Figure 2.3 and two versions of the IEEE 16 buses grid (Figure 2.4) for the simulations. We varied $\bar{I}_{mn}$

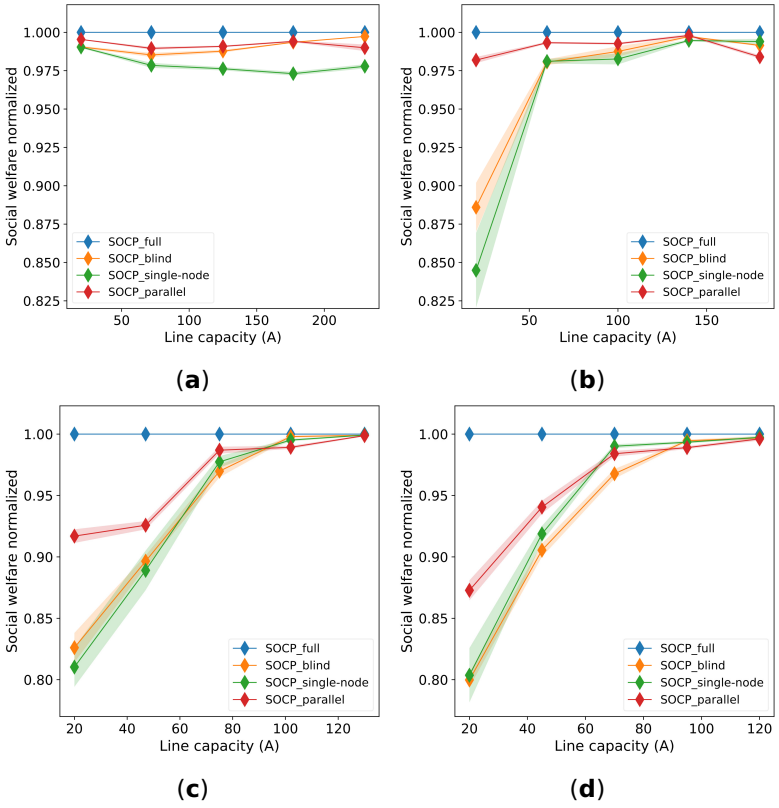


Figure 2.7: Social welfare provided the surrogate models, *full* and *blind* planners. Values are normalized relative to the *full* planner. Horizontal axis is the lines current limit. (**a**) Radial grid with single generator. (**b**) Radial grid with DGs. (**c**) Meshed grid with DGs between cables. (**d**) Meshed grid with DGs between and at cables.

to determine how the tightness of the line constraints affects planing. Importantly, we used different ranges for $\bar{I}_{mn}$ in different grids, such that the highest value in each range represents the case when the line constraints in the *full* planner's solution were not binding. The results for random grids and IEEE16 grids are presented in Figures 2.7 and 2.8, respectively. Appendix A also shows the solutions achieved by different planners at one planning timestep.

The results in Figure 2.7 suggest that all three planners reach nearly optimal performances as the line current constraints become irrelevant. In the tight constraints case, however, the *parallel nodes* planner (labeled SOCP_parallel) is clearly dominant. The *single node* (SOCP_single) performs slightly worse than the *blind* planner in the radial grids, but the gap decreases when the number of generators

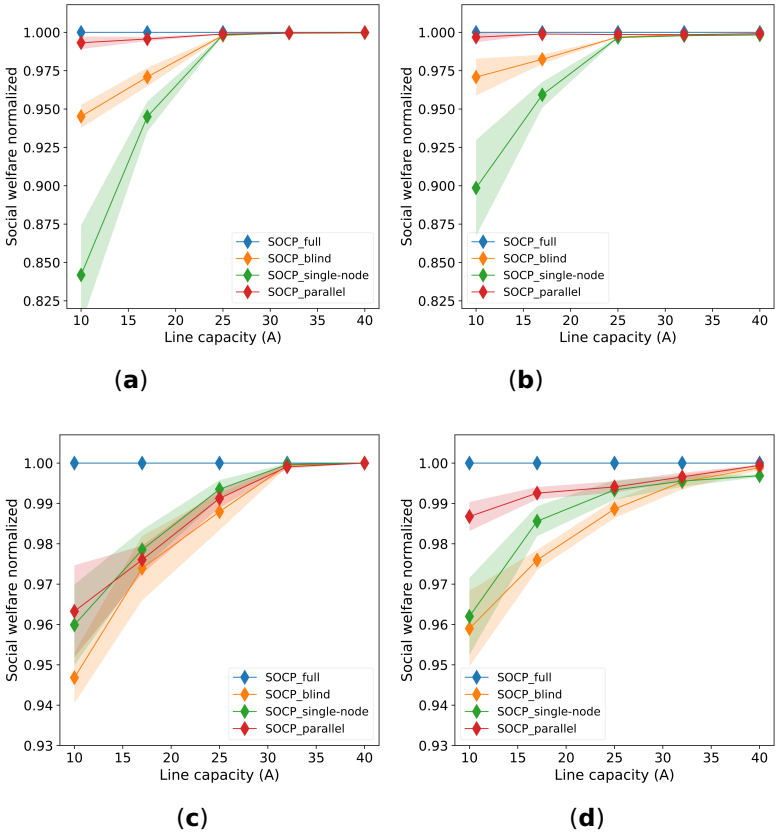


Figure 2.8: Social welfare provided by the surrogate models, *full* and *blind* planners. Values are normalized relative to the *full* planner. Horizontal axis is the lines' current limit. (**a**) Radial IEEE16 grid connected to the external grid. (**b**) Disconnected radial IEEE16 grid. (**c**) Meshed IEEE16 grid connected to the external grid. (**d**) Disconnected meshed IEEE16 grid.

increases. In the meshed grids with additional DGs, the *single node* planner outperforms the blind guessing planner.

The results in Figure 2.8 demonstrate similar behavior. Interestingly, the *parallel nodes* planner is much more dominant in the radial version of the grid. In the meshed case, when more generators are connected to each cable, the *single node* model again slightly outperforms the *blind* planner.

Based on Figures 2.7 and 2.8, we may conclude that the *parallel nodes* planner seems to be the best choice for tightly constrained grids in terms of performance. In practice, however, the computational time may also be an important factor for the EV aggregator. Larger grids and

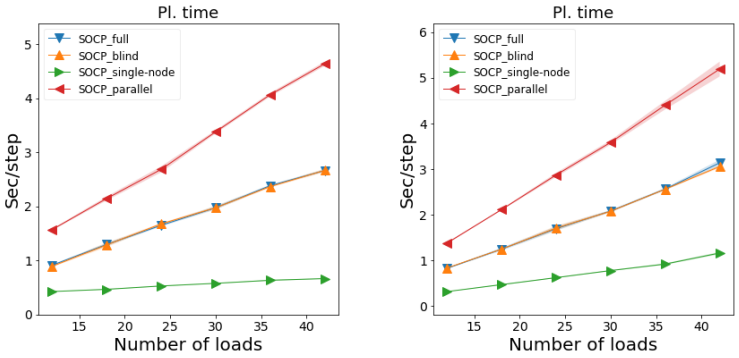


Figure 2.9: Planning time per timestep for different grid sizes. (**a**) Random radial grid with DGs with 3 cables each of equal length varying from 4 to 14 (**b**) Random radial grid with DGs with cables of length 6. The amount of cables varied from 2 to 7.

longer planning horizons may make the EVCP problem hardly tractable even using the SOCP relaxation. Since the *parallel nodes*' surrogate grid uses more lines and passive nodes than the original grid, it is expected to be the slowest method. Similarly, the *single node* model should scale best. Figure 2.9 compares the planning time of different planners as a function of the grid size. For that experiment, we used random radial grids with DGs from Figure 2.3b. We investigated the planning time in the grids with three cables of varying cable lengths (Figure 2.9a) and in the grids with varying numbers of cables with six loads (Figure 2.9b).

The results in Figure 2.9 confirm the poor scalability of the *parallel nodes* planner and show the superiority of the *single node* solution.

2.5. CONCLUSIONS

In this paper, we have extended the EVCP problem for DC microgrids by including partial observability of the EV locations (PO-EVCP). We have studied the effects of partial observability on the performance of planners and suggested two solutions tailored to deal with the unknown EV locations. The experiments in this study lead to the following conclusions about the PO-EVCP:

1. In DC grids with tight line constraints, knowing the locations of the active EVs in the grid is important for computing charging schedules which maximize the social welfare. Practically, this should be considered when designing a communication protocol between EV owners and the EV aggregator.

2. If, due to the limitations of the communication scheme or privacy concerns, the EV aggregator can only partially observe the EV locations, it may prefer to use *parallel nodes* or *single node* models. The former model is clearly dominant performance-wise, but requires more resources, and the latter offers great scalability at the cost of a tiny performance drop.

To ensure realistic conditions, the experiments in this work included a wide range of line capacities and demands, and also randomly generated topologies. Therefore, we are confident that the conclusions drawn from these simulation results are quite general and will also hold in practice.

AUTHORS' CONTRIBUTIONS:

Conceptualization, G.V., W.B., L.M. and M.d.W.; methodology, G.V.; software, G.V.; supervision, W.B., L.M. and M.d.W.; visualization, G.V.; writing – original draft, G.V.; writing —- review and editing, G.V., W.B. and M.d.W. All authors have read and agreed to the published version of the work.

REFERENCES

[1] G. Veviurko. *veviurko/EVCP_partially_observable_locations: Surrogate DC Microgrid Models for Optimization of Charging Electric Vehicles under Partial Observability*. Version publication. Jan. 2025. doi: 10.5281/zenodo.14645510. url: https://doi.org/10.5281/zenodo.14645510.

[2] L. Mackay, N. H. van der Blij, L. Ramirez-Elizondo, and P. Bauer. "Toward the Universal DC Distribution System". In: *Electric Power Components and Systems* 45.10 (2017), pp. 1032–1042. doi: 10.1080/15325008.2017.1318977.

[3] F. S. Al-Ismail. "DC Microgrid Planning, Operation, and Control: A Comprehensive Review". In: *IEEE Access* 9 (2021), pp. 36154–36172. issn: 21693536. doi: 10.1109/ACCESS.2021.3062840.

[4] M. Amjad, A. Ahmad, M. H. Rehmani, and T. Umer. "A review of EVs charging: From the perspective of energy optimization, optimization approaches, and charging techniques". In: *Transportation Research Part D: Transport and Environment* 62 (2018), pp. 386–417. issn: 1361-9209. doi: https://doi.org/10.1016/j.trd.2018.03.006. url: https://www.sciencedirect.com/science/article/pii/S1361920917306120.

[5] J. Lavaei and S. H. Low. "Zero Duality Gap in Optimal Power Flow Problem". In: *IEEE Transactions on Power Systems* 27.1 (2012), pp. 92–107. doi: 10.1109/TPWRS.2011.2160974.

[6] N. Sadeghianpourhamami, J. Deleu, and C. Develder. "Definition and evaluation of model-free coordination of electrical vehicle charging with reinforcement learning". In: *IEEE Transactions on Smart Grid* 11.1 (Sept. 2018), pp. 203–214. arXiv: 1809.10679. url: http://arxiv.org/abs/1809.10679.

[7] Y. Yang, Q. S. Jia, X. Guan, X. Zhang, Z. Qiu, and G. Deconinck. "Decentralized EV-Based Charging Optimization With Building Integrated Wind Energy". In: *IEEE Transactions on Automation Science and Engineering* 16.3 (2019), pp. 1002–1017. issn: 15455955. doi: 10.1109/TASE.2018.2856908.

[8] F. Wu and R. Sioshansi. "A two-stage stochastic optimization model for scheduling electric vehicle charging loads to relieve distribution-system constraints". In: *Transportation Research Part B: Methodological* 102 (2017), pp. 55–82. issn: 0191-2615. doi: https://doi.org/10.1016/j.trb.2017.05.002. url: https://www.sciencedirect.com/science/article/pii/S0191261516305343.

[9] S. E. Kayacık, B. Kocuk, and T. Yüksel. “The Promise of EV-Aware Multi-Period OPF Problem: Cost and Emission Benefits”. In: (2021), pp. 1–10. arXiv: 2107.03868. url: http://arxiv.org/abs/2107.03868.

[10] R. Azizipanah-Abarghooee, V. Terzija, F. Golestaneh, and A. Roosta. “Multiobjective Dynamic Optimal Power Flow Considering Fuzzy-Based Smart Utilization of Mobile Electric Vehicles”. In: *IEEE Transactions on Industrial Informatics* 12.2 (2016), pp. 503–514. issn: 15513203. doi: 10.1109/TII.2016.2518484.

[11] N. Chen, T. Q. Quek, and C. W. Tan. “Optimal charging of electric vehicles in smart grid: Characterization and valley-filling algorithms”. In: *2012 IEEE Third International Conference on Smart Grid Communications (SmartGridComm)*. 2012, pp. 13–18. doi: 10.1109/SmartGridComm.2012.6485952.

[12] Y. Shi, H. D. Tuan, A. V. Savkin, T. Q. Duong, and H. V. Poor. “Model Predictive Control for Smart Grids With Multiple Electric-Vehicle Charging Stations”. In: *IEEE Transactions on Smart Grid* 10.2 (2019), pp. 2127–2136. doi: 10.1109/TSG.2017.2789333.

[13] S. Kaur, T. Kaur, R. Khanna, and P. Singh. “A state of the art of DC microgrids for electric vehicle charging”. In: *2017 4th International Conference on Signal Processing, Computing and Control (ISPCC)*. 2017, pp. 381–386. doi: 10.1109/ISPCC.2017.8269708.

[14] J. Li, F. Liu, Z. Wang, S. H. Low, and S. Mei. “Optimal Power Flow in Stand-Alone DC Microgrids”. In: *IEEE Transactions on Power Systems* (2018). doi: 10.1109/tpwrs.2018.2801280.

[15] *ENTSO-E Transparency Platform*. https://transparency.entsoe.eu/dashboard/show. Accessed: 2021-09-01.

[16] *Electric Vehicle Charging Sessions Dundee*. https://data.dundeecity.gov.uk/dataset/ev-charging-data. Accessed: 2021-09-01.

[17] M. ApS. *MOSEK Fusion API for Python 9.3.11.* 2021. url: https://docs.mosek.com/latest/pythonfusion/index.html.

APPENDIX

This appendix demonstrates an example solution for the PO-EVCP problem obtained by different planners. This example is taken from the IEEE16 meshed grid connected to the external grid (Figure 2.4b). The generation capacitiy $\underline{P_g}$ in the feeders was unlimited, and set to -144444W for the distributed generators. The line current limit, $\bar{I}_{nm}$, was equal for all lines and set to 17A. Tables 2.3 and 2.4 demonstrate the solutions obtained by different planners at a single timestep during the simulation.

Table 2.3: One timestep of the simulation of the IEEE16 meshed grid with exernal grid connection. Comparison of the exact (fully-observable) and blind planners. Columns voltage(V) and p(W) correspond to the nodal voltage and power obtained by the executor; planned p(W) is the solution derived by the corresponding planner.

		Exact			Blind	
	voltage (V)	p (W)	planned p (W)	voltage (V)	p (W)	planned p (W)
Feeder A	400.0	-7000	-7000	399.1	-6984	-7000
Feeder B	399.9	-6999	-6999	400.0	-7000	-6998
Feeder C	400.0	-7000	-7000	399.7	-6994	-7000
load_4	398.8	6626	6626	397.9	6965	6965
load_5	398.8	6962	6962	398.4	19	19
load_6	398.8	3756	3756	397.4	5374	5374
load_7	398.2	526	526	396.7	4995	4995
load_8	398.8	3753	3753	398.8	30	30
load_9	398.9	6610	6610	398.2	6215	6215
load_10	398.2	7495	7495	398.4	6980	6980
load_11	398.8	35	35	397.8	6325	6325
load_12	398.8	6980	6980	397.7	9450	9450
load_13	398.8	0	0	398.5	0	0.0
load_14	397.7	10000	10000	397.9	6403	6403
load_15	398.8	0	0	397.9	0	0.0
load_16	397.7	10000	10000.0	396.7	6483	10000
DG_17	399.9	-6999	-6999	398.6	-6975	-6995
DG_18	400.0	-7000	-7000	399.6	-6993	-7000
DG_19	400.0	-7000	-7000	399.3	-6988	-7000
DG_20	399.3	-6989	-6989	399.5	-6992	-6994
DG_21	400.0	-7000	-7000	398.9	-6981	-7000

Table 2.4: One timestep of the simulation of the IEEE16 meshed grid with exernal grid connection. Comparison of the single-node and parallel nodes planners. Columns voltage(V) and p(W) correspond to the nodal voltage and power obtained by the executor; planned p(W) is the solution derived by the corresponding planner.

		Single-node			Parallel nodes	
	voltage (V)	p (W)	planned p (W)	voltage (V)	p (W)	planned p (W)
Feeder A	398.8	-6978	-6996	397.3	-6952	-6993
Feeder B	398.9	-6981	-6995	398.8	-6979	-6994
Feeder C	400.0	-7000	-7000	400.0	-7000	-6999
load_4	397.6	2788	2788	396.1	7121	7121
load_5	397.1	8250	8250	396.1	7133	7133
load_6	397.4	4835	4835	396.2	5866	5866
load_7	396.9	4997	4997	396.0	7788	8292
load_8	397.7	7661	7661	397.7	5536	5536
load_9	397.4	6457	6457	397.1	5537	5537
load_10	398.2	3669	3669	397.9	6742	6742
load_11	396.8	5708	5708	396.1	5711	5711
load_12	397.6	5680	5680	397.4	5533	5533
load_13	398.8	0	0	398.83	0	0
load_14	398.1	4768	4768	398.2	2552	2552
load_15	398.4	0	0	398.33	0	0
load_16	397.2	5071	7824	397.2	0	2844
DG_17	398.6	-6975	-6996	397.3	-6953	-6993
DG_18	398.2	-6969	-7000	397.3	-6952	-7000
DG_19	398.6	-6975	-6992	398.3	-6970	-6993
DG_20	399.4	-6989	-7000	399.1	-6984	-7000
DG_21	398.8	-6979	-6999	398.5	-6974	-6995

3

ZERO-GRADIENTS IN PREDICT AND OPTIMIZE FOR CONVEX OPTIMIZATION

Many real-world problems — from logistics and scheduling to finance and energy management — can be modeled as constrained optimization problems, a classical area in computer science. In practice, however, key inputs to these problems are often uncertain or unknown. A common solution is to use machine learning (ML) models to predict these inputs and plug the predictions into the optimization problem. This raises a natural question: how should we train these ML models? The standard approach is to optimize for prediction accuracy. However, what usually truly matters is not how accurate the predictions are, but how good the resulting decisions turn out to be. This idea underpins the field of predict-and-optimize, or decision-focused learning, which reformulates the learning problem to directly optimize decision quality.

In this chapter, we explore how this can be done when decisions are made via convex optimization. We identify a key technical issue: in many settings, backpropagation through a convex optimization solver yields zero gradients, causing learning to get stuck. We explain the origin of this problem, show when and why it occurs, and propose a practical solution that enables more reliable gradient flow. Our approach improves learning in several applications, paving the way for more robust decision-aware ML systems.

This chapter is currently under revision for a publication and is available at Arxiv: *Veviurko, G., Böhmer, W., de Weerdt, M. (2023). You Shall Pass: Dealing with the Zero-Gradient Problem in Predict and Optimize for Convex Optimization. arXiv preprint arXiv:2307.16304.*.

ABSTRACT

In predict and optimize (P&O), machine learning models predict parameters of constrained optimization problems, using task performance as the learning objective. This approach requires differentiating the optimization problem's solution with respect to the predicted parameters. Unlike in linear and combinatorial problems, where the gradient is always zero or undefined, exact differentiation is possible and widely used for non-linear convex problems. However, we demonstrate that this approach suffers from the zero-gradient problem, leading to suboptimal solutions. Through formal proofs, we show that the severity of this phenomenon depends on the number of active constraints. To address this issue, we introduce a constraint smoothing technique. By combining this with existing ideas from the literature, we develop a theoretically sound algorithm to approximately differentiate convex optimization problems without encountering the zero-gradient problem. Our experiments confirm that the zero-gradient problem indeed occurs in practice and demonstrate that our proposed solution resolves it efficiently. The code for the paper is publicly available [1].

3.1. INTRODUCTION

Predict and optimize (P&O) [2], also referred to as *decision-focused learning* [3], is a decision-making paradigm that combines machine learning (ML) with constrained optimization. It considers a setup where an ML model first outputs the parameters of an optimization problem, and then the solution is computed using a suitable solver. The distinctive feature of P&O is that it uses the solution quality as the learning objective, rather than optimizing for auxiliary metrics such as prediction accuracy.

Gradient-based optimization methods, such as stochastic gradient descent [4], are by far the most common approach to training ML models. To apply them within the P&O context, we need to differentiate through the solution of an optimization problem with respect to its parameters. The properties of the corresponding Jacobian matrix are known to depend on the problem class. For linear and combinatorial problems, it is always either a zero matrix or undefined. To address this, various approximations have been introduced [2, 5, 6]. For the non-linear case, Agrawal *et al.* [7] developed a method to compute the exact Jacobian for a large class of convex optimization problems. Their approach has found numerous applications and is now considered the standard choice for non-linear convex problems.

In this paper, we demonstrate that using the exact Jacobian of non-linear convex optimization problems may lead to poor results. Specifically, we show that the size of the null space of this Jacobian depends on the number of active constraints. Consequently, the

gradient of the loss function can be zero far from the optimal solution, thereby causing the *zero-gradient problem*. We mathematically investigate the nature of this phenomenon and conclude that it may occur independently of the problem class, essentially for *any* application of the differential optimization method developed by Agrawal *et al.* [7].

Existing methods to compute an approximation of the Jacobian are designed for linear and combinatorial optimization problems, where the ill properties of the Jacobian are known and acknowledged. In this work, we propose a method that combines quadratic approximation, similar to the method by Wilder, Dilkina, and Tambe [8], with projection distance regularization [9] and a novel idea of *constraint smoothing*. The resulting algorithm has a simple geometric interpretation and offers theoretical guarantees on performance. Through extensive benchmarks, we demonstrate that this method resolves the zero-gradient problem for both linear and non-linear problems. In the former case, it performs on par with existing methods for linear problems. In the latter case, it significantly outperforms the sole existing approach of using the exact Jacobian.

3.2. PREDICT AND OPTIMIZE

In this section, we first give an overview of the related work in the predict and optimize domain. Then, we define the P&O problem, introduce the mathematical terminology, and provide the necessary background.

3.2.1. RELATED WORK

The predict and optimize (P&O) framework was introduced by Elmachtoub and Grigas [2]. The authors considered combinatorial optimization problems for which the exact Jacobian is always zero or undefined and derived a convex, sub-differentiable approximation of the task performance function to enable training. Subsequent studies have proposed various alternative approximations: Vlastelica *et al.* [5] derived a differentiable, piecewise-linear approximation for task performance; Berthet *et al.* [10] used stochastic perturbations to approximate the Jacobian of combinatorial problems; and Sahoo *et al.* [6] showed that using simple projections on top of the predictor enables the use of the identity matrix as an approximation of the Jacobian.

For convex optimization, exact differentiation through the optimization problem is possible. Amos and Kolter [11] initially developed an approach to compute the Jacobian of quadratic programs (QPs) analytically by implicit differentiation of the KKT conditions. Later, Agrawal *et al.* [7] used a similar technique and developed a differentiable solver, *cvxpylayers*, for disciplined convex programs [12].

The differentiable optimization method by Agrawal *et al.* [7] has found several applications within P&O: Wilder, Dilkina, and Tambe [8] used it to differentiate a QP relaxation of linear problems; [13] differentiated through logarithmic relaxations; and Ferber *et al.* [14] combined the QP relaxation with the cutting-plane method to differentiate combinatorial optimization problems. In the continuous domain, Uysal, Li, and Mulvey [15] applied differentiable optimization to the risk-budgeting portfolio optimization problem, while Wang *et al.* [16] utilized it to learn surrogate models for P&O.

Beyond P&O, differentiable optimization has found applications in other fields: Chen *et al.* [9] used it as an action projection layer in reinforcement learning; Dawson, Gao, and Fan [17] employed it for safe optimal control; and Lee *et al.* [18] applied it to meta-supervised learning.

In summary, analytic differentiation through convex optimization problems is immensely useful and can be utilized in various ways. However, the properties of this process have not been thoroughly studied. Specifically, the question of whether and under what conditions the Jacobian of non-linear convex problems can be zero has not been addressed, even though it is a well-known issue for linear convex problems. In this paper, we aim to investigate this question and demonstrate that there are inherent challenges associated with using the exact Jacobian that are not addressed in the existing literature.

3.2.2. PROBLEM FORMULATION

In this section, we introduce the P&O problem. We refer readers to the work by Elmachtoub and Grigas [2] for further details. In predict and optimize, the *true problem* we are attempting to solve is of the following form:

$$\arg\max_x f(x, w) \text{ s. t. } x \in \mathcal{C}, \tag{3.1}$$

where $x \in \mathbb{R}^n$ is the decision variable, $w \in \mathbb{R}^u$ is a vector of unknown parameters, $f : \mathbb{R}^n \times \mathbb{R}^u \to \mathbb{R}$ is the objective function, and $\mathcal{C}$ is the feasible set. The defining feature of this problem is that the parameters w are unknown at the moment when the decision must be made. Therefore, the true optimization problem is under-defined and cannot be solved directly.

One way to deal with the unknown parameters w is to use a prediction $\hat{w}$ instead. Then, the decision can be computed by solving the *decision problem*:

$$x^*(\hat{w}) = \arg\max_x f(x, \hat{w}) \text{ s. t. } x \in \mathcal{C}. \tag{3.2}$$

In P&O, we assume that instead of the unknown parameters w, we observe a feature vector o that contains some information about w. Besides, we assume a dataset $\mathcal{D} = \{(o_k, w_k)\}$, e.g., of historical data,

which we can use to learn the relation between w and o. This setup enables the use of ML models for making predictions. We denote the prediction model by ϕ_θ, and thus we have $\hat{w} = \phi_\theta(o)$.

The problem described above is not specific to predict and optimize. What separates the P&O paradigm from earlier works is the approach to training the model ϕ_θ. In the past, machine learning models would be trained to predict w as accurately as possible [19]. However, the parameter prediction error is merely an artificial objective and our true goal is to derive a decision x that maximizes the task performance $f(x, w)$. The main goal of the P&O approach is to utilize this objective for training the model ϕ_θ. The task performance achieved by ϕ_θ on the dataset $\mathcal{D}$ can be quantified by the following loss function:

$$L(\theta) = -\frac{1}{|\mathcal{D}|} \sum_{(o,w)\in\mathcal{D}} f\Big(x^*\big(\phi_\theta(o)\big), w\Big) \tag{3.3}$$

To train ϕ_θ with a gradient-based algorithm, we need to differentiate L over θ, and hence we need to compute the gradient $\nabla_\theta f\Big(x^*(\hat{w}), w\Big)$, where $\hat{w} = \phi_\theta(o)$. Applying the chain rule, it can be decomposed into three terms:

$$\nabla_\theta f\Big(x^*(\hat{w}), w\Big) = \nabla_x f\big(x^*(\hat{w}), w\big)\, \nabla_{\hat{w}} x^*(\hat{w})\, \nabla_\theta \hat{w} \tag{3.4}$$

The second term, $\nabla_{\hat{w}} x^*(\hat{w})$, is the Jacobian of the solution of the optimization problem with respect to the prediction $\hat{w}$. As discussed in the previous section, this term is the primary object of study in most P&O works. For combinatorial problems, it is undefined, and for linear convex problems, it is zero at all points except for a measure-zero set, where it is undefined. Hence, approximations are used for these two cases. For strictly convex optimization problems, it is common to compute this Jacobian exactly [7]. In the next section, we study the Jacobian $\nabla_{\hat{w}} x^*(\hat{w})$ and demonstrate that even for strictly convex problems, it can have a large null space, thereby causing the total gradient in Eq. (3.4) to be zero in arbitrary regions outside of the optimum.

3.3. DIFFERENTIABLE OPTIMIZATION

Without loss of generality, we consider a single instance of the problem, i.e., one sample $(o, w) \in \mathcal{D}$, for the theoretical analysis. Everywhere in this section, we denote the prediction by $\hat{w} = \phi_\theta(o)$. Then, the decision is computed as a solution of the decision problem in Eq. (3.2). We use $\hat{x}$ to denote the value of $x^*(\hat{w})$ for a given prediction $\hat{w}$. As we are interested in convex optimization problems, we make the following assumptions:

Assumption 3.3.1. The objective function $f(x, w)$ is concave and twice continuously differentiable in x for any w.

Assumption 3.3.2. The feasible set $\mathcal{C}$ is convex, i.e., $\{\mathcal{C} = \{x | g_i(x) \leq 0, i = 1, \ldots, l\}$, where $g_i(x)$ are convex differentiable functions. Moreover, for any $x \in \mathcal{C}$, the gradients $\{\nabla_x g_i(x) | g_i(x) = 0\}$ of the active constraints are linearly independent. [1]

Additionally, we make an assumption about how f depends on w, which holds for many real-world problems, including linear and quadratic optimization problems.

Assumption 3.3.3. The objective function $f(x, w)$ is twice continuously differentiable in w.

Throughout this paper, we use derivatives of different objects: the gradient of the true objective function with respect to the decision, $\nabla_x f(\hat{x}, w)$; the Jacobian of the decision with respect to the prediction, $\nabla_{\hat{w}} x^*(\hat{w})$; the Jacobian of the prediction with respect to the ML model parameters, $\nabla_\theta \hat{w}$; and the gradient of the predicted objective in the decision problem, $\nabla_x f(x, \hat{w})$. In the next section, we demonstrate that the null space of $\nabla_{\hat{w}} x^*(\hat{w})$ depends on the number of active constraints at $x^*(\hat{w})$ and hence can result in gradient-based optimization methods getting stuck in suboptimal solutions.

3.3.1. THE ZERO-GRADIENT THEOREM

We begin by investigating the relation between the values of the function $x^*(\hat{w})$ and the gradient of the internal objective, $\nabla_x f(x, \hat{w})$. Let $n_i := \nabla_x g_i(\hat{x})$, $i = 1, \ldots, l$ be the normal vectors of the constraints at $\hat{x}$, Then, the KKT conditions [20] at $\hat{x}$ state that there exist real values $\alpha_1, \ldots, \alpha_l$ such that the following holds:

$$\nabla_x f(\hat{x}, \hat{w}) = \sum_{i=1}^{l} \alpha_i n_i, \quad \alpha_i g_i(\hat{x}) = 0,$$

$$\alpha_i \geq 0, \quad g_i(\hat{x}) \leq 0, \quad i = 1, \ldots, l.$$

Under Assumptions 3.3.1 and 3.3.2, the KKT multipliers α_i are uniquely defined by $\hat{w}$ and $\hat{x}$. Thus, as $\hat{x}$ is defined by $\hat{w}$, we sometimes write $\alpha_i(\hat{w})$ to emphasize that it is, in fact, a function of $\hat{w}$. To provide a geometrical perspective on the KKT conditions, we introduce the following definition:

[1] As is, Assumption 3.3.2 does not allow equality constraints since they would violate linear independence property. For clarity, we use this formulation in the main body of the paper. In the appendix, we show that our results hold for the equality constraints as well.

Definition 3.3.4. Let $x \in \mathcal{C}$ and let $I(x) = \{i | g_i(x) = 0\}$ be the set of indices of the constraints active at x. Let $n_i = \nabla_x g_i(x)$, $\forall i \in I(x)$, be the normal vectors of these constraints. The gradient cone, $G(x) := \left\{ \sum_{i \in I} \alpha_i n_i | \alpha_i \geq 0 \right\}$, is the positive linear span of normal vectors n_i.

Combining the KKT conditions with Definition 3.3.4, we immediately arrive at the following property:

Property 3.3.5. *Let $x \in \mathcal{C}$ and let $\nabla_x f(x, \hat{w})$ be the internal gradient at x. Then, x is a solution to the problem in Eq. (3.2) if and only if $\exists \alpha_i \geq 0$, $i \in I(x)$ such that $\nabla_x f(x, \hat{w}) = \sum_{i \in I(x)} \alpha_i n_i \in G(x)$, where $I(x)$ is the set of indices of active constraints, $I(x) = \{i | g_i(x) = 0\}$.*

This property provides a geometrical interpretation of the problem. Effectively, a point x is a solution to the problem in Eq. (3.2) if and only if the internal gradient at this point is inside its gradient cone. Figure 3.1 illustrates this property.

Before studying the Jacobian $\nabla_{\hat{w}} x^*(\hat{w})$, we first need to address the question of when this Jacobian exists. Sufficient conditions for existence are given by [21]. Under Assumptions 3.3.1-3.3.3, these conditions can be reformulated as follows:

Lemma 3.3.6 (Theorem 2.1 by [21])**.** *Let Assumptions 3.3.1-3.3.3 hold and let*

$$\nabla_x f(\hat{x}, \hat{w}) = \sum_{i \in I(\hat{x})} \alpha_i(\hat{w}) n_i$$

be the representation of the internal gradient with the normals of the active constraints. Let the strict complementary slackness (SCS) conditions *hold, i.e., $\alpha_i(\hat{w}) > 0$, $\forall i \in I(\hat{x})$. Then, the Jacobian $\nabla_{\hat{w}} x^*(\hat{w})$ exists at $\hat{w}$. Moreover, $\alpha_i(\cdot)$ is continuous around $\hat{w}$ for any $i \in I(\hat{x})$.*

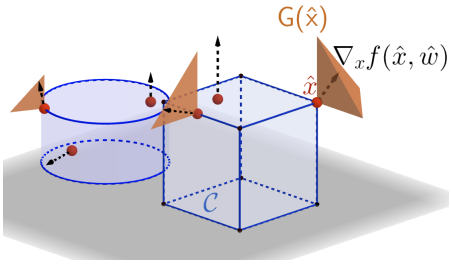


Figure 3.1: Gradient cones $\hat{x} + G(\hat{x})$ (orange cones) and internal gradients $\nabla_x f(\hat{x}, \hat{w})$ (black arrows) at different points $\hat{x}$ (red dots) in different feasible sets $\mathcal{C}$ (blue cube and cylinder). The points $\hat{x}$ can not be moved in the dimensions spanned by the cones.

The proof can be found in the work by Fiacco [21]. This result establishes that SCS is sufficient for the Jacobian $\nabla_{\hat{w}} x^*(\hat{w})$ to exist. In most cases, the points that violate strict complementary slackness form a measure-zero set and hence can be neglected in practice.

We have all the necessary tools to describe the structure of the Jacobian $\nabla_{\hat{w}} x^*(\hat{w})$. Suppose that the SCS conditions hold at $\hat{x}$ and hence the Jacobian exists. Assume that we perturb $\hat{w}$ and obtain $\hat{w}'$. Let $\hat{x}' = x^*(\hat{w}')$ denote the solution corresponding to $\hat{w}'$. What can be said about $\hat{x}'$? Strict complementary slackness implies that the constraints active at $\hat{x}$ will remain active at $\hat{x}'$ if the difference $\|\hat{w}' - \hat{w}\|_2^2$ is small enough. Therefore, the decision $\hat{x}'$ can only move within the tangent space of $\mathcal{C}$ at $\hat{x}$, i.e., orthogonally to all n_i, $i \in I(\hat{x}.)$ Hence, when more constraints are active, $\hat{x}'$ can move in less directions. Formally, we obtain the following:

Lemma 3.3.7. *Suppose that the SCS conditions hold at $\hat{x}$ and let $\nabla_x f(\hat{x}, \hat{w}) = \sum_{i \in I(\hat{x})} \alpha_i n_i$, $\alpha_i > 0$, $\forall i \in I(\hat{x})$ be the internal gradient. Let $\mathcal{N}(\hat{x}) = span(\{n_i \mid i \in I(\hat{x})\})$ be the linear span of the gradient cone. Then $\mathcal{N}(\hat{x})$ is contained in the left null space of $\nabla_{\hat{w}} x^*(\hat{w})$, i.e., $v \nabla_{\hat{w}} x^*(\hat{w}) = 0$, $\forall v \in \mathcal{N}(\hat{x})$*

The formal proof of this result can be found in the appendix. Lemma 3.3.7 is very important, as it specifies in what directions $x^*(\hat{w})$ *can move* as a consequence of changing $\hat{w}$. Now, the first term in the chain rule in Eq. (3.4), $\nabla_x f(\hat{x}, w)$, specifies in what directions $x^*(\hat{w})$ *should* move in order for the true objective to increase. Naturally, if these directions are contained in the null space of $\nabla_{\hat{w}} x^*(\hat{w})$, then the total gradient in Eq. (3.4) is zero. This observation constitutes the main theorem of this paper – the zero-gradient theorem.

Theorem 3.3.8 (Zero-gradient theorem)**.** *Let $\hat{w}$ be a prediction, and let $\hat{x}$ be the solution of the decision problem defined in Eq. (3.2). Suppose that the strict complementary slackness conditions hold at $\hat{x}$ and let $\mathcal{N}(\hat{x}) = span(\{n_i \mid i \in I(\hat{x})\})$ be the linear span of the gradient cone at $\hat{x}$. Then, $\nabla_x f(\hat{x}, w) \in \mathcal{N}(\hat{x}) \implies \nabla_\theta f(\hat{x}, w) = 0$.*

The proof of this theorem is obtained by simply applying Lemma 3.3.7 to the chain rule in Eq. (3.4). The theorem claims that the gradient of the P&O loss in Eq. (3.3) can be zero in the points outside of the optimal solution. Hence, any gradient-following method "shall not pass" these points. In particular, the zero-gradient phenomenon occurs in such points $\hat{x}$ where the true gradient $\nabla_x f(\hat{x}, w)$ is contained in the space $\mathcal{N}(\hat{x})$ spanned by the gradient cone $G(\hat{x})$. As the dimensionality of this space grows with the number of active constraints, the zero-gradient issue is particularly important for problems with a large number of constraints. In the worst case, $\mathcal{N}(\hat{x})$ can be as big as the whole decision space $\mathbb{R}^n$, thereby making the total gradient $\nabla_\theta f(\hat{x}, w)$ from Eq. (3.4)

zero for any value of the true gradient $\nabla_x f(\hat{x}, w)$. In the following sections, we introduce a method that resolves the zero-gradient problem and provides theoretical guarantees for its performance.

3.3.2. QUADRATIC APPROXIMATION

The fundamental assumption of the predict and optimize framework is that training ϕ_θ using the task performance loss is better than fitting it to the true values of w. Hence, the models trained with predict and optimize might output $\hat{w}$ that is significantly different from the true w and yet produces good decisions. Taking this argument one step further, we claim that the objective function of the decision problem in Eq. (3.2) does not need to be the same as the true objective $f(x, w)$. In particular, we suggest computing decisions using a simple quadratic problem (QP) resembling the one proposed for linear problems by Wilder, Dilkina, and Tambe [8]:

$$x^*_{QP}(\hat{w}) = \arg\max_x -\|x - \hat{w}\|_2^2 \text{ s.t. } x \in \mathcal{C}. \tag{3.5}$$

The reasons for this choice are manyfold. First, the internal objective $f_{QP}(x, \hat{w}) = -\|x - \hat{w}\|_2^2$, is strictly concave and hence $x^*_{QP}(\hat{w})$ is always uniquely-defined. Moreover, the range of $x_{QP}(\hat{w})$ is $\mathcal{C}$, i.e., $\forall x \in \mathcal{C}$, $\exists \hat{w}$ such that $x = x^*_{QP}(\hat{w})$. Hence, it can represent any optimal solution.

Wilder, Dilkina, and Tambe [8] proposed to use the QP approximation to differentiate through linear problems. However, as shown in Theorem 3.3.8, the Jacobian of $x^*_{QP}(w)$ can still be non-informative. We see the main advantage of using QP in the fact that it uses the smallest reasonable prediction vector $\hat{w}$ (one scalar per decision variable). Besides, QP approximation can represent any solution, and, as we show below, its Jacobian has a simple analytic form, which allows computing it cheaply and enables studying its theoretical properties.

The problem in Eq. (3.5) has a simple geometrical interpretation: the point $x = \hat{w}$ is the unconstrained maximum of $f_{QP}(x, \hat{w})$ and $x^*_{QP}(\hat{w})$ is its Euclidean projection on the feasible set $\mathcal{C}$, see Figure 3.2. To compute the Jacobian $\nabla_{\hat{w}} x^*_{QP}$, we need to understand how perturbations of $\hat{w}$ affect x^*_{QP}. Employing the geometrical intuition above, we obtain the following lemma:

Lemma 3.3.9. *Let $\hat{w}$ be a prediction and $\hat{x}$ be the optimal solution of the QP problem defined in Eq. (3.5). Let the strict complementary slackness condition hold and let $\{n_i | i \in I(\hat{x})\}$ be the normals of the active constraints. Let $\{e_j | j = 1, \dots, n - |I(\hat{x})|\}$ be an orthogonal complement of vectors $\{n_i | i \in I(\hat{x})\}$ to a basis of $\mathbb{R}^n$. Then, representation of the Jacobian $\nabla_{\hat{w}} x_{QP}(\hat{w})$ in the basis $\{n_i\} \cup \{e_j\}$ is a diagonal matrix. Its first $|I(\hat{x})|$ diagonal entries are zero, and the others are one.*

Proof of this lemma can be found in the appendix. Lemma 3.3.9 implies that the Jacobian $\nabla_{\hat{w}} x_{QP}(\hat{w})$ has a simple form and can be easily computed by hand.

QP approximation does not address the zero-gradient problem and we introduce it for computational reasons: it is sufficiently powerful and its simple Jacobian can be computed cheaply. In the next section, we introduce the novel *constraint smoothing* method that computes an approximate Jacobian with the null-space of the maximum size one. Although this method addresses zero-gradient issues in any optimization problem, demonstrating that the resulting approximation aligns with the true Jacobian in general is challenging. For the QP approximations, however, it is possible to derive strong theoretical guarantees.

3.3.3. CONSTRAINT SMOOTHING

We identified that the zero-gradient problem is a fundamental issue of differential optimization that also appears in non-linear convex optimization. We showed that the size of the null space of the Jacobian $\nabla_{\hat{w}} x(\hat{w})$ depends on the number of constraints active at $\hat{x}$. Generally, this number can be as large as the number of optimized variables n, and the gradient-descent algorithms can get stuck in certain points on the boundary of the feasible set.

A potential solution to this issue is to approximate the feasible set $\mathcal{C}$ in such a way that all gradient cones become one-dimensional. It is known that any convex set can be approximated with a convex polytope [22], and a convex polytope can be approximated with a smooth convex set [23]. By combining these results we can approximate $\mathcal{C}$ with a smooth $\mathcal{C}'$. Then, the null space of the Jacobian of the resulting problem over $\mathcal{C}'$ becomes strictly one-dimensional. Therefore, it is possible to derive an arbitrarily close approximation of the problem such that the null space of its Jacobian is at most one-dimensional everywhere. In practice, however, this approach requires a computationally efficient way to derive an approximation of $\mathcal{C}$, and hence we leave it as a potential future work direction. Instead, we propose a simple way to modify the feasible set – we smooth $\mathcal{C}$ locally around the point for which we compute the Jacobian, thereby ensuring that its null space is one-dimensional. Combining this approach with the QP approximation yields a theoretically sound algorithm.

Let $\nabla_x f(\hat{x}, \hat{w}) = \sum_{i \in I(\hat{x})} \alpha_i n_i$ be the internal gradient at $\hat{x}$ for some $\alpha_i \geq 0,\ \forall i \in I(\hat{x})$. Then, we introduce the following definition:

Definition 3.3.10. Let $r > 0$ be a positive real number and let $c = \hat{x} - r \frac{\nabla_x f(\hat{x}, \hat{w})}{\|\nabla_x f(\hat{x}, \hat{w})\|_2}$. Then, the locally smoothed feasible set, $\mathcal{C}_r(\hat{x}, \hat{w}) := \{y | y \in \mathbb{R}^n, \|y - c\|_2 \leq r\}$, is a ball of radius r around c. The corresponding locally smoothed problem $P_r(\hat{x}, \hat{w})$ with parameters $\hat{x}, \hat{w}$ is defined as $x_r^*(\hat{w}) := \arg\max_{x \in \mathcal{C}_r(\hat{x}, \hat{w})} f(x, \hat{w})$.

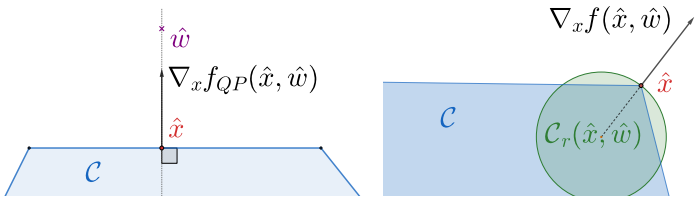


Figure 3.2: *Left*: Illustration of the QP approximation. The internal gradient (black arrow) at the solution of the QP $\hat{x}$ (red point) is orthogonal to the feasible set $\mathcal{C}$ (blue area) and points towards the unconstrained maximum $\hat{w}$ (purple cross). *Right*: Illustration of the smoothed problem. The internal gradient (black arrow) is orthogonal to the smoothed feasible set $\mathcal{C}_r(\hat{x}, \hat{w})$ (green circle) at the decision $\hat{x}$ (red point).

Figure 3.2 shows an example of the local smooth problem. Now, let $\hat{x}_r = x_r^*(\hat{w})$ denote the solution of $P_r(\hat{x}, \hat{w})$. By construction, the internal gradient at $\hat{x}_r$ lies in the one-dimensional gradient cone, and hence, by Property 3.3.5, $\hat{x}_r = \hat{x}$. The main purpose of smoothing is to approximate the gradient in Eq. (3.4) by substituting $\nabla_{\hat{w}} x^*(\hat{w})$ with $\nabla_{\hat{w}} x_r^*(\hat{w})$. We highlight that the decisions are still computed using the non-smoothed problem $x^*(\hat{w})$ and $x_r^*(\hat{x}, \hat{w})$ is used exclusively to perform the gradient update step. In other words, we use the following expression to compute the gradient:

$$\nabla_\theta f(x^*(\hat{w}), w) \approx \nabla_x f\big(\hat{x}, w\big) \nabla_{\hat{w}} x_r^*(\hat{w}) \nabla_\theta \hat{w} \tag{3.6}$$

Applying Lemma 3.3.7 we see that the null-space of $\nabla_{\hat{w}} x_r^*(\hat{w})$ is at most one dimensional.

Next, we demonstrate that the approximate Jacobian $\nabla_{\hat{w}} x_r^*(\hat{w})$ of the locally smoothed problem is consistent with the true Jacobian if QP approximation is used. Specifically, we show that performing gradient steps using locally smoothed QP problem is guaranteed to not decrease the objective. To prove that, we first use Lemma 3.3.9 to describe the Jacobian of the smoothed QP problem:

Property 3.3.11. *Let $\hat{x} = x_{QP}^*(\hat{w})$ be a decision derived via QP. Suppose that the complementary slackness conditions hold for $P_r(\hat{x}, \hat{w})$ and let $e_1 = \nabla_x f_{QP}(\hat{x}, \hat{w})$ be the internal gradient. Let $\{e_2, \ldots, e_n\}$ be a complement of e_1 to an orthogonal basis of $\mathbb{R}^n$. Then, the Jacobian $\nabla_{\hat{w}} x_r^*(\hat{w})$ of the locally smoothed problem expressed in the basis $\{e_1, e_2, \ldots, e_n\}$ is a diagonal matrix. Its first entry is zero, others are ones.*

As we see from Property 3.3.11, the value of r does not affect the Jacobian. We keep it only for notation clarity. Property 3.3.11 reveals a

connection to the work by Sahoo *et al.* [6], where the authors propose to substitute the Jacobian of linear problems by identity matrix. Then, they apply projection operators, chosen by a human, to eliminate certain directions. The constraint smoothing approach can be seen as an extension of this approach to the more general convex case. Finally, we can prove that the Jacobian of the smoothed QP problem yields a "good" direction for the gradient update.

Theorem 3.3.12. *Let $\hat{x} = x^*_{QP}(\hat{w})$ be the decision obtained via QP and let $\nabla_{\hat{w}} x^*_r(\hat{w})$ be the Jacobian of the locally smoothed QP problem. Let $\Delta\hat{w} = \nabla_x f(\hat{x}, w)\, \nabla_{\hat{w}} x^*_r(\hat{w})$ be the prediction perturbation obtained by using this Jacobian and let $\hat{w}'(t) = \hat{w} + t\Delta\hat{w}$ be the updated prediction. Then, for $t \to 0^+$, using $\hat{w}'(t)$ results in a non-decrease in the task performance, i.e., $f\big(x^*_{QP}(\hat{w}'(t)), w\big) \geq f\big(x^*_{QP}(\hat{w}), w\big)$.*

Theorem 3.3.12 shows that using constraint smoothing together with the QP approximation results in an analytically computable Jacobian with a one-dimensional null space that results in good gradient steps. Therefore, we are much less likely to encounter the zero-gradient problem when using this approximation. However, the resulting one-dimensional null space contains the only direction that can move the prediction $\hat{w}$, and hence the decision $\hat{x}$, inside $\mathcal{C}$. This might become crucial, for example, when the optimal solution with respect to the true objective lies in the interior of $\mathcal{C}$. To resolve this problem, we use the projection distance regularization method first suggested by Chen *et al.* [9]. Specifically, we add a penalty term

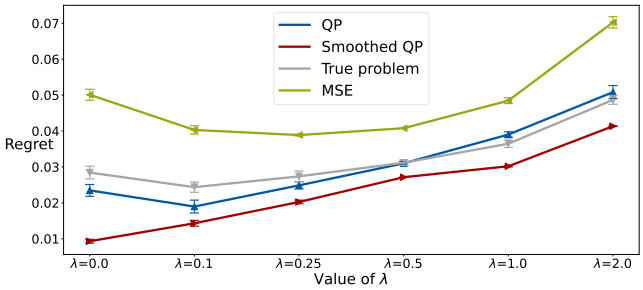
$$p(\hat{w}) = \alpha \|\hat{x} - \hat{w}\|_2^2, \tag{3.7}$$

where $\alpha \in \mathbb{R}^+$ is a hyperparameter. Minimizing this term, we push $\hat{w}$ along the null space of the Jacobian towards the feasible set and eventually move $\hat{x}$ inside $\mathcal{C}$.

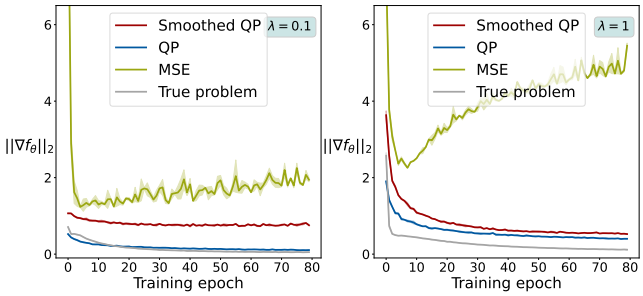
3.4. EXPERIMENTS

We derived the zero-gradient theorem describing when the gradient of the P&O objective for non-linear convex optimization can be zero outside of the optimum. To deal with this, we proposed a solution that combines QP approximation with constraint smoothing and projection distance regularization. Below, we conduct experiments to verify our theoretical results. Specifically, we address the following:

1. Does the zero-gradient indeed occur in practice?
2. Does our method solve the zero-gradient problem?
3. Does QP approximation still work if the original problem is not quadratic?



(**a**)



(**b**)

Figure 3.3: Results on the standard portfolio optimization problem: (a) the final test regret for each of the algorithms for varying λ's. The smoothed QP performs the best. *(b)* Evolution of the l_2 norm of the gradient during training for $\lambda = 0.1$ and $\lambda = 1$. Unlike standalone QP and differentiation of the true problem, smoothed QP does not have the zero-gradient issue.

3.4.1. PORTFOLIO OPTIMIZATION

Following Wang *et al.* [16], we study the portfolio optimization problem, where we aim to maximize the immediate return but minimize the risk under the budget constraint:

$$\arg\max_x \underbrace{p^\top x - \lambda x^\top Q x}_{f(x,p,Q)} \quad \text{s. t.} \quad \sum_{i=1}^{n} x_i = 1,\ x \geq 0. \tag{3.8}$$

The decision $x \in \mathbb{R}^n$ is the investment, $p \in \mathbb{R}^n$ is the immediate return, and $Q \in \mathbb{R}^{n \times n}$ is the covariance matrix. The unknown parameters are defined as $w := (p, Q)$ and $\lambda \geq 0$ is the risk-aversion weight. We refer the readers to the appendix and to the code for further details regarding the implementation of this and all other benchmarks. We consider different values of λ from the set $\{0, 0.1, 0.25, 0.5, 1, 2\}$, in order to generate a spectrum of problems, from the linear ($\lambda = 0$) to the "strongly quadratic" ($\lambda = 2$).

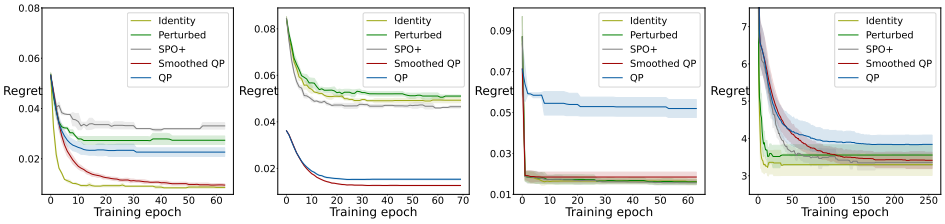


Figure 3.4: The test regret of linear P&O algorithms, QP approximation, and smoothed QP on four benchmark problems on a) Linear portfolio; b) Portfolio with $\lambda = 0.1$; c) OPF; d) Knapsack.

We compare the performance of four methods: naive minimization of the mean-squared error of the prediction (labeled "MSE"); differentiation through the true problem in Eq (3.8) ("True problem"); differentiation through the QP approximation ("QP"); and our approach that combines QP approximation, smoothing, and projection distance regularization ("Smoothed QP"). Importantly, smoothed QP uses different values of the projection distance regularization weight α from Eq (3.7) for different λ's. Specifically, $\alpha = 0$ for $\lambda \in \{0, 0.1\}$, $\alpha = 0.01$ for $\lambda \in \{0.25, 0.5\}$, and $\alpha = 0.1$ for $\lambda \in \{1, 2\}$. The remaining hyperparameter values for all methods are provided in the appendix. For the performance metric, we use *regret* [2], defined as follows:

$$\text{regret}(o, w) = \max_x f(x, w) - f\Big(x^*\big(\phi_\theta(o)\big), w\Big). \tag{3.9}$$

The results in Figure 3.3 demonstrate that the smoothed QP approach is dominant – it outperforms the competitors by a significant margin across all values of λ. Figure 3.3 (b) suggests the reason for this is indeed the zero-gradient problem: for the methods using the exact Jacobian (QP and true problem), the gradient norm decreases rapidly with training.

Figure 3.3 suggests that the QP approximation is sufficient in portfolio optimization. However, it might be explained by the fact that the true problem in Eq. (3.8) is also quadratic. To gain more insight into the performance of our method in non-quadratic cases, we introduce an artificial modification of the problem's objective using the *LogSumExp* function:

$$f_{lse}(x, p, Q) = -\log \sum_i e^{-p_i x_i} - \lambda x^\top Q x \tag{3.10}$$

The LogSumExp function acts as a soft maximum, and the corresponding optimization problem can be interpreted as the maximization of the most profitable investment. The results in Table 3.1 demonstrate that the QP approximation outperforms the true problem in terms of regret

and significantly reduces the computation time. Moreover, smoothed QP again outperforms the other approaches, which suggests that the zero-gradient problem occurs in the LogSumExp case as well.

3.4.2. COMPARISON TO LINEAR METHODS

As discussed earlier, the non-linear convex P&O problems were considered to be solvable solely by exact differentiation. Above, we show that this is not the case as the zero-gradient phenomenon causes the training process to get stuck. In the linear case, on the other hand, the zero-gradient is a well-known issue and there exist various methods to approximate the Jacobian of linear optimization problems.

We implement several state-of-the-art methods for linear P&O and compare them to the smoothed/standalone QP approaches. Specifically, we use the SPO+ loss [2], the perturbed optimizers approach [10], and the identity-with-projection method [6]. Besides, using the QP approximation alone is equivalent to the approach suggested by Wilder, Dilkina, and Tambe [8]. We ran these methods on the linear ($\lambda = 0$) and "almost linear" ($\lambda = 0.1$) portfolio optimization problems, on the linear optimal power flow (OPF) problem, and on the continuous knapsack problem. The OPF problem is based on the linear model for the power flow in DC grids ([24, 25]; the knapsack is adapted from the PyEPO package [26]. For further details regarding these problems, we refer the readers to the appendix. The results in Figure 3.4 show that our smoothed QP approach can compete with the existing algorithms in linear problems.

3.4.3. META SUPERVISED LEARNING

In the last experiment, we use the meta-learning setup by Lee *et al.* [18]. In this setup, we have a feature extractor, ϕ (typically, a large CNN), and a base classifier class $\mathcal{A}$ (in our case, SVM). Then, we have a *meta-training* set, $\{\mathcal{D}_i^{train}, \mathcal{D}_i^{test}\}_{i=0}^{N}$, sampled from the task distribution. Each tuple $\left(\mathcal{D}_i^{train}, \mathcal{D}_i^{test}\right)$ describes a training and a test task. Typically, $\mathcal{D}_i^{train}$ is small, as we are interested in the few-shot learning. For each pair of tasks, we train an instance of $\mathcal{A}$ on $\phi(\mathcal{D}_i^{train})$ and test it on $\phi(\mathcal{D}_i^{test})$ to measure the test performance. The meta learning objective is to derive such ϕ, that the features it extracts lead good test performance. The approach used by [18] builds upon the fact that SVM training is a convex optimization problem. Therefore, this process can be differentiated and the feature extractor ϕ can be trained end-to-end.

In our experiment, we rerun the code provided by Lee *et al.* [18] for CIFAR-FS dataset [27] with and without our constraint smoothing method. The results in Table 3.2 demonstrate that smoothing improves the performance, thereby indicating that the zero-gradient problems

	Regret	Runtime (sec)
True problem	0.834 ± 0.120	7965 ± 52
QP	0.506 ± 0.009	762 ± 52
Smoothed QP	**0.438** ± **0.009**	801 ± 54

Table 3.1: Final (normalized) test regret and training time for the different methods on the LogSumExp portfolio problem.

occurs in this setup as well. Importantly, in the reported runs, we do not use QP approximation and apply smoothing to the original SVM problem, which is a quadratic program. The reason for this is that the QP approximation method learns slower than the true SVM problem, as the latter is specifically designed for classification. For more details on the experiment details, we refer the readers to the appendix and the paper by Lee *et al.* [18].

In summary, the experiments provide significant evidence that the zero-gradient problem occurs in non-linear convex P&O problems, such as portfolio optimization and meta supervised learning. The proposed solution is shown to be effective in resolving the problem in these cases as well as to be competitive in in the case of linear P&O.

3.5. CONCLUSION

In this work, we provide theoretical evidence for the zero-gradient problem in the non-linear convex P&O setting. We show that this problem can occur for any convex problem, as long as constraints are activated during training. This result affects various applications of the differential optimization method – the main tool for the non-linear convex P&O problems. To resolve this issue, we introduce a method to approximate of the Jacobian. It is done by smoothing the feasible set around the current solution, thereby reducing the null space's dimensionality to one. We prove that the combination of smoothing with the QP approximation results in the gradient update steps that do not decrease the task performance, but often allow to escape the zero-gradient cones.

To support our theoretical findings, we conduct various experiments. Using the portfolio optimization problem, we demonstrate that the zero-gradient indeed occurs in practice and our solution consistently improves upon the exact differentiation. We also show that our approach works for linear problems and matches the performance of the state-of-the-art methods. Moreover, we showed that our smoothing techniques helps meta-learning with differentiable optimization.

In future work, we aim to investigate alternative approaches to smoothing, such as global smoothing of the constraints. Besides, we want to study what theoretical guarantees can be obtained for

	SVM	Smoothed SVM
1-shot accuracy	58.74% ± 0.88%	**65.15%** ± **0.54%**
5-shot accuracy	70.95% ± 0.33%	**76.42%** ± **0.26%**

Table 3.2: Test accuracy of meta-learning with differentiable optimization using exact/smoothed Jacobian of the SVM

.

smoothing of non-quadratic problems. Moreover, we want to further investigate the applicability of QP approximation to general convex problems

AUTHORS' CONTRIBUTIONS:

G.V. conceived the research, developed the methodology, conducted all experiments, and prepared the initial manuscript. W.B. and M.d.W. provided critical feedback throughout the research process and contributed significantly to the revision and refinement of the manuscript.

REFERENCES

[1] G. Veviurko. *You Shall Pass: Dealing with the Zero-Gradient Problem in Predict and Optimize for Convex Optimization*. Jan. 2025. doi: 10.5281/zenodo.14645375. url: https://doi.org/10.5281/zenodo.14645375.

[2] A. N. Elmachtoub and P. Grigas. “Smart" Predict, then Optimize"”. In: *arXiv preprint arXiv:1710.08005* (2017).

[3] J. Mandi, J. Kotary, S. Berden, M. Mulamba, V. Bucarey, T. Guns, and F. Fioretto. *Decision-Focused Learning: Foundations, State of the Art, Benchmark and Future Opportunities*. 2024. arXiv: 2307.13565 [cs.LG]. url: https://arxiv.org/abs/2307.13565.

[4] J. Kiefer and J. Wolfowitz. “Stochastic estimation of the maximum of a regression function”. In: *The Annals of Mathematical Statistics* (1952), pp. 462–466.

[5] M. Vlastelica, A. Paulus, V. Musil, G. Martius, and M. Rolínek. “Differentiation of Blackbox Combinatorial Solvers”. In: (2019), pp. 1–19.

[6] S. S. Sahoo, A. Paulus, M. Vlastelica, V. Musil, V. Kuleshov, and G. Martius. *Backpropagation through Combinatorial Algorithms: Identity with Projection Works*. 2023. arXiv: 2205.15213 [cs.LG].

[7] A. Agrawal, B. Amos, S. Barratt, S. Boyd, S. Diamond, and J. Z. Kolter. “Differentiable convex optimization layers”. In: *Adv. Neural Inf. Process. Syst.* 32 (2019).

[8] B. Wilder, B. Dilkina, and M. Tambe. “Melding the Data-Decisions Pipeline: Decision-Focused Learning for Combinatorial Optimization”. en. In: *AAAI* 33.01 (July 2019), pp. 1658–1665.

[9] B. Chen, P. L. Donti, K. Baker, J. Z. Kolter, and M. Bergés. “Enforcing Policy Feasibility Constraints through Differentiable Projection for Energy Optimization”. In: *e-Energy 2021 - Proceedings of the 2021 12th ACM International Conference on Future Energy Systems* (2021), pp. 199–210.

[10] Q. Berthet, M. Blondel, O. Teboul, M. Cuturi, J. P. Vert, and F. Bach. “Learning with differentiable perturbed optimizers”. In: *Adv. Neural Inf. Process. Syst.* 2020-Decem (2020), pp. 1–24.

[11] B. Amos and J. Z. Kolter. “OptNet: Differentiable optimization as a layer in neural networks”. In: *34th International Conference on Machine Learning, ICML 2017* 1 (2017), pp. 179–191.

[12] M. Grant, S. Boyd, and Y. Ye. “Disciplined Convex Programming”. In: *Global Optimization: From Theory to Implementation*. Ed. by L. Liberti and N. Maculan. Boston, MA: Springer US, 2006, pp. 155–210. isbn: 978-0-387-30528-8.

[13] J. Mandi and T. Guns. “Interior Point Solving for LP-based prediction+optimisation”. In: (Oct. 2020). arXiv: 2010.13943 [cs.NE].

[14] A. Ferber, B. Wilder, B. Dilkina, and M. Tambe. “MIPaaL: Mixed integer program as a layer”. In: *AAAI 2020 - 34th AAAI Conference on Artificial Intelligence* (2020), pp. 1504–1511.

[15] A. S. Uysal, X. Li, and J. M. Mulvey. “End-to-End Risk Budgeting Portfolio Optimization with Neural Networks”. In: (July 2021). arXiv: 2107.04636 [q-fin.PM].

[16] X. Wang, C. Sun, R. Wang, and T. Wei. “Two-Stage Optimal Scheduling Strategy for Large-Scale Electric Vehicles”. In: *IEEE Access* 8 (2020), pp. 13821–13832. doi: 10.1109/ACCESS.2020.2966825.

[17] C. Dawson, S. Gao, and C. Fan. “Safe Control With Learned Certificates: A Survey of Neural Lyapunov, Barrier, and Contraction Methods for Robotics and Control”. In: *IEEE Transactions on Robotics* 39.3 (2023), pp. 1749–1767. doi: 10.1109/TRO.2022.3232542.

[18] K. Lee, S. Maji, A. Ravichandran, and S. Soatto. *Meta-Learning with Differentiable Convex Optimization*. 2019. arXiv: 1904.03758 [cs.CV]. url: https://arxiv.org/abs/1904.03758.

[19] A. Mukhopadhyay and Y. Vorobeychik. “Prioritized allocation of emergency responders based on a continuous-time incident prediction model”. In: *International Conference on Autonomous Agents and MultiAgent Systems*. 2017.

[20] H. W. Kuhn and A. W. Tucker. “Nonlinear programming”. In: *In Berkeley Symposium on Mathematical Statistics and Probability* 2 (1951), pp. 481–492.

[21] A. V. Fiacco. “Sensitivity analysis for nonlinear programming using penalty methods”. In: *Mathematical programming* 10.1 (1976), pp. 287–311.

[22] E. M. Bronstein. “Approximation of convex sets by polytopes”. In: *Journal of Mathematical Sciences* 153.6 (2008), pp. 727–762.

[23] M. Ghomi. “Optimal Smoothing for Convex Polytopes”. In: *Bulletin of the London Mathematical Society* 36.4 (2004), pp. 483–492. doi: https://doi.org/10.1112/S0024609303003059. eprint: https://londmathsoc.onlinelibrary.wiley.com/doi/pdf/10.1112/S0024609303003059.

[24] J. Li, F. Liu, Z. Wang, S. H. Low, and S. Mei. “Optimal Power Flow in Stand-Alone DC Microgrids”. In: *IEEE Transactions on Power Systems* 33.5 (2018), pp. 5496–5506. issn: 08858950. doi: 10.1109/TPWRS.2018.2801280. arXiv: 1708.05140.

[25] G. Veviurko, W. Böhmer, L. Mackay, and M. de Weerdt. “Surrogate DC Microgrid Models for Optimization of Charging Electric Vehicles under Partial Observability”. In: *Energies* 15.4 (2022), p. 1389.

[26] B. Tang and E. B. Khalil. “PyEPO: A PyTorch-based End-to-End Predict-then-Optimize Library for Linear and Integer Programming”. In: *arXiv preprint arXiv:2206.14234* (2022).

[27] L. Bertinetto, J. F. Henriques, P. H. S. Torr, and A. Vedaldi. *Meta-learning with differentiable closed-form solvers*. 2019. arXiv: 1805.08136 [cs.CV]. url: https://arxiv.org/abs/1805.08136.

[28] QUANDL. *Quandl WIKI prices, 2020.* 2020.

3.6. APPENDIX

3.6.1. PROOFS

Proof of Lemma 3.3.7. Let $\Delta\hat{w}$ denote an arbitrary direction and let $d = \nabla_{\hat{w}} x^*(\hat{w})\Delta\hat{w}$ be the corresponding directional derivative of the decision. The existence of d is guaranteed by the strict complementary slackness conditions and Lemma 3.3.6. Let $t \to 0^+$. Then, we have

$$\hat{x}'(t) := x^*(\hat{w} + t\Delta\hat{w}) = \hat{x} + td + o_x(t),$$

where $o_x(t)$ is the "little o" notation, i.e., $\lim_{t\to 0^+} \frac{\|o_x(t)\|_2}{t} = 0$. To prove the lemma, we first want to show that $d^\top n_i = 0,\ \forall i \in I(\hat{x})$. Then, we will show that it implies the lemma's claim.

By definition, $n_i = \nabla_x g_i(\hat{x})$. Then, since $g_i(\cdot)$ is differentiable and $g_i(\hat{x}) = 0,\ \forall i \in I(\hat{x})$, we have the following first-order approximation for $g_i\big(\hat{x}'(t)\big)$:

$$g_i\big(\hat{x}'(t)\big) = g_i\big(\hat{x} + td + o(t)\big) =$$
$$g_i(\hat{x}) + tn_i^\top d + o_g(t) = tn_i^\top d + o_g(t).$$

Since $\hat{x}'$ is the solution of the internal optimization problem, the inequality $g_i(\hat{x}'(t)) \le 0$ holds. Hence, the equation above implies that $n_i^\top d \le 0$. Now, we want to show that, in fact, $n_i^\top d = 0$. For a proof by contradiction, suppose that $n_i^\top d < 0$. Then, by definition of $o_g(t)$, there exists $\epsilon > 0$, such that

$$0 < t < \epsilon \implies g_i\big(\hat{x}'(t)\big) < 0.$$

Now, we will to show that $g_i\big(\hat{x}'(t)\big) < 0$ contradicts the complementary slackness condition at $\hat{x}$. From Lemma 3.3.6, we know the KKT multiplier, $\alpha_i'(t) := \alpha_i(\hat{w} + t\Delta\hat{w})$, is a continuous function of t. On the one hand, from the KKT conditions, we know that $g_i\big(\hat{x}'(t)\big) < 0 \implies \alpha_i'(t) = 0$. Therefore, $\alpha_i'(t) = 0$ for $t < \epsilon$. Hence, we have

$$\lim_{t\to 0^+} \alpha_i'(t) = 0.$$

On the other hand, the continuity implies that $\lim_{t\to 0^+} \alpha_i'(t) = \alpha_i'(0) = \alpha_i$ and, due to strict complementary slackness, $\alpha_i > 0$. Hence, we also have

$$\lim_{t\to 0^+} \alpha_i'(t) > 0.$$

We arrived at a contradiction and therefore can claim that $d^\top n_i = 0$ for all n_i. Since $\{n_i | i \in I(\hat{x})\}$ is a basis of $\mathcal{N}(\hat{x})$, this implies that for any direction $v \in \mathcal{N}(\hat{x})$ and for any $\Delta\hat{w}$, we have $v^\top \nabla_{\hat{w}} x^*(\hat{w})\Delta\hat{w} = 0$. In other words, vector $v^\top \nabla_{\hat{w}} x^*(\hat{w})$ is orthogonal to the whole space of $\hat{w}$ and hence it must be zero, $v^\top \nabla_{\hat{w}} x^*(\hat{w}) = 0,\ \forall v \in \mathcal{N}(\hat{x})$. Hence $\mathcal{N}(\hat{x})$ is contained in the left null space of $\nabla_{\hat{w}} x^*(\hat{w})$. □

Proof of Lemma 3.3.9. First, consider the case when the unconstrained maximum $\hat{w}$ is in the interior of $\mathcal{C}$. By definition of x^*_{QP}, it means that $\hat{x} = x^*_{QP}(\hat{w})$ is also in the interior of $\mathcal{C}$ and $\hat{x} = \hat{w}$. Then, x^*_{QP} is the identity function around $\hat{w}$, and hence $x^*_{QP}(\hat{w} + \Delta\hat{w}) = x(\hat{w}) + \Delta\hat{w}$ for small enough $\Delta\hat{w}$. Hence, $\nabla_{\hat{w}} x^*_{QP}(\hat{w}) = I$. Since no constraints are active in this case ($I(\hat{x}) = \emptyset$), the lemma's claim holds.

Now, consider the case when some constraints are active, and thus $\hat{x}$ lies on the boundary of $\mathcal{C}$. To get the exact form of the Jacobian $\nabla_x x^*_{QP}(\hat{w})$, we will compute $\lim_{t\to 0} x^*_{QP}(\hat{w} + t\Delta\hat{w})$ for all possible $\Delta\hat{w}$. As in the QP case the predictions $\hat{w}$ lie in the same space as $\hat{x}$, we can do it first for $\Delta\hat{w} \in \mathcal{N}(\hat{x})$ and then for $\Delta\hat{w} \perp \mathcal{N}(\hat{x})$.

1. $\Delta\hat{w} \in \mathcal{N}(\hat{x})$. For $\Delta\hat{w} \in \mathcal{N}(\hat{x})$, we want to show that the corresponding directional derivative is zero. We begin by computing the internal gradient $\nabla_x f_{QP}(\hat{x}, \hat{w})$:

$$\nabla_x f_{QP}(\hat{x}, \hat{w}) = -\nabla_x \|x - w\|_2^2 = 2(\hat{w} - \hat{x}).$$

Using this formula, we can write the internal gradient for the perturbed prediction $\hat{w} + t\Delta\hat{w}$ at the same point $\hat{x}$:

$$\nabla_x f_{QP}(\hat{x}, \hat{w} + t\Delta\hat{w}) = \nabla_x f_{QP}(\hat{x}, \hat{w}) + 2t\Delta\hat{w}.$$

By definition, $\mathcal{N}(\hat{x})$ is a linear span of the vectors $\{n_i | i \in I(\hat{x})\}$. Hence, since $\Delta\hat{w} \in \mathcal{N}(\hat{x})$, it can be expressed as

$$\Delta\hat{w} = \sum_{i \in I(\hat{x})} \delta_i n_i, \quad \delta_i \in \mathbb{R}. \tag{$*$}$$

By Property 3.3.5, the internal gradient has the following representation:

$$\nabla_x f_{QP}(\hat{x}, \hat{w}) = \sum_{i \in I(\hat{x})} \alpha_i n_i, \quad \alpha_i > 0. \tag{$**$}$$

Then, combining $(*)$ and $(**)$, we obtain

$$\nabla_x f_{QP}(\hat{x}, \hat{w} + t\Delta\hat{w}) = \nabla_x f_{QP}(\hat{x}, \hat{w}) + 2t\Delta\hat{w} = \sum_{i \in I(\hat{x})} (\alpha_i + 2t\delta_i) n_i$$

Since $\alpha_i > 0$, $\forall i \in I(\hat{x})$, there exists $\epsilon > 0$, such that $\alpha_i - 2t\delta_i > 0$ for $|t| < \epsilon$. Therefore, $\nabla_x f_{QP}(\hat{x}, \hat{w} + t\Delta\hat{w})$ lies in the gradient cone of $\hat{x}$, and hence, by Property 3.5, $x^*_{QP}(\hat{w} + t\Delta\hat{w}) = \hat{x}$ for $|t| < \epsilon$. Therefore, the directional derivative of $x^*_{QP}(\hat{w})$ along $\Delta\hat{w} \in \mathcal{N}(\hat{x})$ is zero.

2. $\Delta\hat{w} \perp \mathcal{N}(\hat{x})$. Next, let $\Delta\hat{w}$ be orthogonal to $\mathcal{N}(\hat{x})$. We begin with the first order approximation of $\hat{x}'(t)$:

$$\hat{x}'(t) = \hat{x} + td + o(t).$$

From the proof of Lemma 3.3.7, we know that $d \perp \mathcal{N}$. By definition of x^*_{QP}, we know that $\hat{x}$ is the point on $\mathcal{C}$ closest to $\hat{w}$. Likewise, $\hat{x}'(t)$ is the point on $\mathcal{C}$ closest to $\hat{w} + t\Delta\hat{w}$. Hence, $d = \Delta\hat{w}$. Therefore, for any $\Delta\hat{w} \perp \mathcal{N}$, the directional derivative of $x_{QP}(\hat{w})$ along $\Delta\hat{w}$ is one. So, we have shown that

$$\nabla_{\hat{w}} x^*_{QP}(\hat{w})\Delta\hat{w} = \begin{cases} 0 & \text{for } \Delta\hat{w} \in \mathcal{N}(\hat{x}) \\ \Delta\hat{w} & \text{for } \Delta\hat{w} \perp \mathcal{N}(\hat{x}). \end{cases}$$

Therefore, the lemma is proven. □

Proof of Theorem 3.3.8. First, we want to construct an orthogonal basis $\{e_1, \dots e_n\}$ of $\mathbb{R}^n$ that will greatly simplify the calculations. We start by including the internal gradient in this basis, i.e., we define $e_1 = \nabla_x f_{QP}(\hat{x}, \hat{w})$. Then, let $I(\hat{x}) = \{i | g_i(\hat{x}) = 0\}$ be the set of indices of the active constraints of the original problem and let $\mathcal{N}(\hat{x}) = span(\{n_i | i \in I(\hat{x})\})$ be a linear span of their normals. By the liner independence condition from Assumption 3.3.2, $dim\big(\mathcal{N}(\hat{x})\big) = |I(\hat{x})|$. Moreover, by Property 3.3.5, we know that $e_1 \in \mathcal{N}(\hat{x})$. Then, we can choose vectors $e_2, \dots, e_{|I(\hat{x})|}$ that complement e_1 to an orthogonal basis of $\mathcal{N}(\hat{x})$. The remaining vectors $e_{|I(\hat{x})|+1}, \dots, e_n$, are chosen to complement $e_1, \dots, e_{|I(\hat{x})|}$ to an orthogonal basis of $\mathbb{R}^n$. The choice of this basis is motivated by Lemma 3.9: e_1 is a basis of the null-space of the $r-$smoothed Jacobian, $e_1, \dots, e_{|I(\hat{x})|}$ form a basis of the null space of the true QP Jacobian, and the remaining vectors form a basis of space in which we can move $x^*_{QP}(\hat{w})$.

For brevity, let $f_x = \nabla_x f(\hat{x}, w)$ denote the true gradient vector. By definition, $\Delta\hat{w} = f_x \nabla_{\hat{w}} x^*_r(\hat{x}, \hat{w})$ is obtained via the $r-$smoothed problem. From Property 3.3.11, we know that $\Delta\hat{w}$ is a projection of f_x on the vectors $e_2, \dots, e_n$. Then, since $e_1, \dots, e_n$ is an orthogonal basis, we have

$$\Delta\hat{w} = \sum_{i=2}^{n} \beta_i e_i, \quad \beta_i = f_x^\top e_i,\ i = 2, \dots, n.$$

Now, let's see how this $\Delta\hat{w}$ affects the true decision $x^*_{QP}(\hat{w} + t\Delta\hat{w})$ for $t \to 0^+$. First, we have a first-order approximation

$$x^*_{QP}(\hat{w} + t\Delta\hat{w}) = \hat{x} + td + o(t),$$

for some $d \in \mathbb{R}$. From Lemma 3.3.9, we know that d is actually a

projection of $\Delta\hat{w}$ onto the vectors $e_{|I(\hat{x})|+1}, \ldots, e_n$. Therefore, we have

$$x^*_{QP}(\hat{w} + t\Delta\hat{w}) = \hat{x} + \sum_{i=|I(\hat{x})|+1}^{n} \beta_i e_i + o(t).$$

Finally, the change in the true objective can be expressed as

$$\begin{aligned}
&f\big(x^*_{QP}(\hat{w} + t\Delta\hat{w}), w\big) - f\big(x^*_{QP}(\hat{w}), w\big) = \\
&t f_x^\top \Big(\sum_{i=|I(\hat{x})|+1}^{n} \beta_i e_i \Big) + o(t) = \\
&t \sum_{i=|I(\hat{x})|+1}^{n} \beta_i f_x^\top e_i + o(t) \\
&= t \sum_{i=|I(\hat{x})|+1}^{n} \beta_i^2 + o(t) \geq 0.
\end{aligned}$$

Therefore, perturbing prediction along $\Delta\hat{w}$ does not decrease the true objective $f(\hat{x}, w)$, and hence

$$f\big(x^*_{QP}(\hat{w} + t\Delta\hat{w}), w\big) \geq f\big(x^*_{QP}(\hat{w}), w\big)$$

for $t \to 0^+$. □

3.7. EQUALITY CONSTRAINTS

Assumption 3.3.2 postulates that for any $x \in \mathcal{C}$, the gradients of active constraints, $\{\nabla_x g_i(x) | g_i(x) = 0\}$, are linearly independent. Now, suppose we include equality constraints in our problem. e.g., we have a constraint $g^{eq}(x) \leq 0$ and $-g^{eq}(x) \leq 0$ for some g. Clearly, the gradients of $g^{eq}(x)$ and $-g^{eq}(x)$ violate the independence assumption. However, we claim that it does not affect our results. Let $\hat{w}$ and $\hat{x}$ be a prediction and a corresponding decision and let $n^{eq} = \nabla_x g^{eq}(\hat{x})$. Suppose the equality constraint $g^{eq}(\hat{x}) = 0$ is active. Let $I(\hat{x})$ be the set of indices of the active constraints *not including* $g^{eq}(x)$. Then, we have a representation of the internal gradient,

$$\nabla_x f(\hat{x}, \hat{w}) = \alpha_1^{eq} n^{eq} - \alpha_2^{eq} n^{eq} + \sum_{i \in I(\hat{x})} \alpha_i n_i.$$

Suppose that $\alpha_1^{eq} \neq \alpha_2^{eq}$, e.g., without loss of generality, $\alpha_1^{eq} > \alpha_2^{eq}$. Then,

$$\nabla_x f(\hat{x}, \hat{w}) = (\alpha_1^{eq} - \alpha_2^{eq}) n^{eq} + \sum_{i \in I(\hat{x})} \alpha_i n_i$$

and hence removing the constraint $-g^{eq}(x) \leq 0$ would not change the optimality of $\hat{x}$. The remaining problem would satisfy complementary slackness and hence would have all the properties demonstrated in Section 3. Therefore, for the case with equality constraints, we need to extend the complementary slackness conditions by demanding $\alpha_1^{eq} \neq \alpha_2^{eq}$.

3.7.1. EXPERIMENTAL DETAILS

In this section, we provide the details of the experiments reported in the paper. All experiments were conducted on a single machine with 32gb RAM and NVIDIA GeForce RTX 3070. The code is written in Python 3.8, and neural networks are implemented in *PyTorch* 1.11. For methods requiring differentiation of optimization problems, we use the implementation by Agrawal *et al.* [7]. The linear methods (SPO+, perturbed optimizers, identity-with-projection) were re-implemented by us.

Experimental results reported in Figures 3.3 and 3.4 show the average and the standard deviation (shaded region) of the measured quantities across 4 random seeds. For each seed, we randomly split data into train, validation, and test sets by using 70%, 20%, and 10% of the whole dataset respectively. In Figure 3a, for each method and at each run, we take the model version corresponding to the best performance on the validation set and report its performance on the test set. In Figure 4, we do the same procedure at each training iteration.

For all experiments except for meta learning, the predictor ϕ_θ is represented by a fully connected neural network with two hidden layers of 256 neurons each, and *LeakyReLU* activation functions. The output layer has no activation function. Instead, the output of the neural network is scaled by the factor x_{scale} and shifted by x_{shift}. For linear methods and methods using QP approximation, the output layer predicts vector $\hat{w}$ of the same dimensionality as the decision variable. For the method using the true model, the prediction size is defined by the number of unknown parameters in the true objective function. For training, we used the *Adam* optimizer from PyTorch, with custom learning rate and otherwise default parameters. Hyperparameters of all methods were chosen based on the results of the grid search reported in Tables 2-6. Configuration files to reproduce the experiments and the code are publicly available [1].

For the meta learning experiments, we used the official code by Lee *et al.* [18]. We modified it by implementing a the constraint smoothing method and otherwise left unchanged. We did not tune the hyperparameters and used value provided by the authors.

Parameter	Search space	Value
Learning rate	$\{0.5, 1, 5, 10\} \times 10^{-5}$	5×10^{-5}
Training epochs	$\{40, 80, 160\}$	80
Batch size	$\{1, 4, 8, 32\}$	1
x_{shift}	$\{0, 0.1, 1\}$	0.1
x_{scale}	$\{0.1, 1\}$	1

Table 3.3: Hyperparameters for methods from Figure 3 for standard portfolio optimization problem with different λ's.

PORTFOLIO OPTIMIZATION PROBLEM

Following Wang *et al.* [16], we use historical data from QUANDL WIKI prices [28] for 505 largest companies on the American market for the period 2014-2017. The dataset is processed and for every day we obtain a feature vector summarizing the recent price dynamic. For further details on the processing, we refer readers to the code and to the original paper by Wang *et al.* [16]. The processed dataset contained historical data describing the past price dynamics for each of the 505 securities. For every random seed, 50 securities (thus, 50 decision variables) were chosen randomly. The experiments on the LogSumExp variation of the portfolio optimization problem were conducted similarly. The hyperparameters for normal and LogSumExp portfolio problems are reported in Tables 2, 3, and 4.

OPTIMAL POWER FLOW PROBLEM

We considered a linearized DC-OPF problem that represents a DC grid without power losses. The decision variable is the vector of nodal voltages $v \in \mathbb{R}^n$, and the unknown parameter w represents either the value gained by serving power to a customer or the price paid for utilizing a generator. The reference voltage $v_0 \in \mathbb{R}$, the admittance matrix Y, and the constraint bounds represent the physical properties of the grid.

$$\begin{aligned} \max_{v} \quad & f(v, w) = -v_0 w^\top (Yv) \\ \text{subject to:} \quad & \underline{V} \le v \le \bar{V} \\ & \underline{P} \le -v_0 Yv \le \bar{P} \\ & \underline{I} \le Y_{ij}(v_i - v_j) \le \bar{I} \end{aligned}$$

Data for the DC OPF problem is generated artificially. First, we randomly generate a grid topology, see Figure 3.5 for an example. For each line, its admittance is set to $6S$. Nodal voltages are bounded between 325V and 375V, and the reference node has a fixed voltage of $v_0 = 350V$. The demand in loads (power upper-bound), generators capacity (power lower-bound), and line current limits are sampled

Parameter	Search space	Value
Learning rate	$\{0.5, 1, 5, 10\} \times 10^{-5}$	5×10^{-5}
Batch size	$\{1, 4, 8, 32\}$	1
x_{shift}	$\{0, 0.1, 1\}$	0.1
x_{scale}	$\{0.1, 1\}$	0.1

Table 3.4: Hyperparameters for methods from the LogSumExp portfolio optimization problem.

randomly from the following normal distributions: $\mathcal{N}(8000, 2500)\times$watt-hour, $\mathcal{N}(-14000, 2500)\times$watt-hour, $\mathcal{N}(25, 5)\times$ampere. The coefficients w are also sampled form the normal distributions: $\mathcal{N}(1.2, 1)$ for loads, and $\mathcal{N}(0.8, 0.1)$ for generators. Finally, all values are normalized such that v_0 becomes $7V$ (surprisingly, it performed better numerically than scaling v_0 to $1V$). The observations o is a concatenation of the true coefficients w, demand of the loads, the capacity of the generators, and line current limits plus normally distributed noise with mean 0 and standard deviation 0.5.

KNAPSACK PROBLEM

For the knapsack problem, we used the data generation process from the PyEPO package [26] The true problem is defined by the number of decision variables n, objective coefficients $w \in \mathbb{R}^n$, and m resource constraints represented by $W \in \mathbb{R}^{n\times n}, b \in \mathbb{R}^m$:

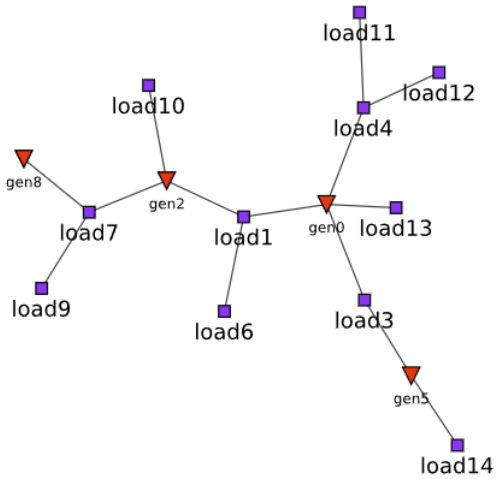


Figure 3.5: Example of randomly generated grid topology. Red triangles represent generator nodes, and purple squares represent loads.

Parameter	Search space	Value
Learning rate	$\{0.5, 1, 5, 10\} \times 10^{-5}$	5×10^{-5}
Batch size	$\{1, 4, 8, 32\}$	1
x_{shift}	$\{0, 0.1, 1\}$	0.1
x_{scale}	$\{0.1, 1\}$	0.1
ID: projection	{mean, norm, both}	norm
Perturbed: n samples	$\{1, 4, 8, 32\}$	4
Perturbed: σ	$\{0.05, 0.1, 0.5\}$	0.05

Table 3.5: Hyperparameters for linear methods from Figure 4c for power flow problem.

Parameter	Search space	Value
Learning rate	$\{0.5, 1, 5, 10\} \times 10^{-5}$	5×10^{-5}
Batch size	$\{1, 4, 8, 32\}$	1
x_{shift}	$\{0, 0.1, 1\}$	0.1
x_{scale}	$\{0.1, 1\}$	0.1
ID: projection	{mean, norm, both}	norm
Perturbed: n samples	$\{1, 4, 8, 32\}$	4
Perturbed: σ	$\{0.05, 0.1, 0.5\}$	0.05

Table 3.6: Hyperparameters for linear methods from Figure 4a, b for standard portfolio optimization problem with $\lambda = 0$ and $\lambda = 0.1$.

$$\begin{aligned} \max_x \quad & f(x, w) = w^\top x \\ \text{subject to:} \quad & 0 \leq x \leq 1 \\ & Wx \leq b \end{aligned}$$

In the experiments, we used $n = 20, m = 15$. Weights W were sampled uniformly from the interval $(3, 8)$ and b was sampled uniformly from $(20, 80)$. The constraints are fixed for all instances of the dataset (i.e., defined by the random seed). For each seed, we construct a dataset of 512 samples. To do so we first sample the observation, $o \sim \mathcal{N}(0, 1)$, $o \in \mathbb{R}^{512 \times 16}$. Then, for each sample $i = \{1, \ldots, 512\}$, we compute the coefficients w as

$$w^i = \left((\frac{1}{4} B o^i + 3)^2 + 1\right) * 0.4 + \epsilon^i,$$

where $B \in \mathbb{R}^{n \times p}$ is sampled from the Bernoulli distribution with $k = 1, p = 0.5$ (fixed across samples) and $\epsilon \in \mathbb{R}^{512 \times 20}$ is noise uniformly sampled from $(0.5, 1.5)$ (different for each sample).

META SUPERVISED LEARNING

For completeness, we provide a more detailed description of the meta learning experiment. For further details, we refer reader to the original paper by Lee *et al.* [18].

We have a feature extractor network, ϕ, that we train on the *meta-training* set, $\{\mathcal{D}_i^{train}, \mathcal{D}_i^{test}\}_{i=0}^{N}$. Assuming for simplicity batch size of one, the training process goes as follows:

1. A pair, $(\mathcal{D}_i^{train}, \mathcal{D}_i^{test})$, of train and test tasks is sampled
2. All samples within each task are embedded using ϕ to obtain $(\phi(\mathcal{D}_i^{train}), \phi(\mathcal{D}_i^{test}))$.
3. An SVM base learner is trained on $\phi(\mathcal{D}_i^{train})$, by solving the following problem:
$$\theta^* = \arg\min_{\theta} L^{SVM}(\phi(\mathcal{D}_i^{train}))$$
Importantly, it is a convex quadratic problem, and hence is differentiable.
4. The trained SVM θ^* is applied to the test task $\mathcal{D}_i^{test}$ to obtain the cross-entropy loss $L^{meta}\left(\mathcal{D}_i^{test}, \theta^*, \phi\right)$
5. The weights of ϕ are updated by performing a gradient step on L^{meta}.

Similarly to other experiments, we use *meta-training, meta-validation*, and *meta-test* sets. The training lasts for 30 epochs and the best model is chosen based on the accuracy on the meta-validation set. Then, the accuracy on the meta-test set is reported.

Parameter	Search space	Value
Learning rate	$\{0.5, 1, 5, 10\} \times 10^{-5}$	5×10^{-5}
Batch size	$\{1, 4, 8, 32\}$	1
x_{shift}	$\{0, 0.1, 1\}$	0.1
x_{scale}	$\{0.1, 1\}$	0.1
ID: projection	{mean, norm, both}	norm
Perturbed: n samples	$\{1, 4, 8, 32\}$	4
Perturbed: σ	$\{0.05, 0.1, 0.5\}$	0.05

Table 3.7: Hyperparameters for linear methods from Figure 4d for the knapsack problem

4

MAXIMUM REWARD REINFORCEMENT LEARNING

Reinforcement learning (RL) is a framework for training agents to make decisions by interacting with their environment and learning from feedback, or rewards. The learning objective is to accumulate as much reward as possible during each interaction. While elegant in theory, RL often struggles in practice when rewards are sparse or delayed, making it hard to link actions to outcomes. A common workaround is to design a surrogate reward that provides more regular feedback. However, this solution is fragile: even small design errors can lead to reward "hacking" and unintended behavior.

In this chapter, we explore a fundamentally different problem reformulation of RL. Instead of focusing on maximizing the total reward collected over time, we study how to optimize the maximum reward an agent obtains within an episode. This objective shifts the focus from cumulative reward to discovering and reliably achieving the best possible outcome. It can be particularly natural in settings where a single high-reward event is the main goal – for example, a robot reaching a destination. Although this idea has intuitive appeal, prior attempts to formalize it have encountered major technical challenges, particularly in settings involving uncertainty in the environment or the agent's policy. This chapter presents the first rigorous and general formulation of max-reward RL, along with algorithms and insights that make it practical.

This chapter is based on the publication in the conference proceedings: *Veviurko, G., Boehmer, W., de Weerdt, M. To the Max: Reinventing Reward in Reinforcement Learning. In Forty-first International Conference on Machine Learning. Compared to the published version, Section 4.1 is adjusted for broader audience, minor adjustments made in Sections 4.2, 4.3.*

ABSTRACT

In reinforcement learning (RL), different reward functions can define the same optimal policy but result in drastically different learning performance. For some, the agent gets stuck with a suboptimal behavior, and for others, it solves the task efficiently. Choosing a good reward function is hence an extremely important yet challenging problem. In this paper, we explore an alternative approach for using rewards for learning. We introduce max-reward RL, where an agent optimizes the maximum rather than the cumulative reward. Unlike earlier works, our approach works for both deterministic and stochastic environments and can be easily combined with state-of-the-art RL algorithms. In the experiments, we study the performance of max-reward RL algorithms in two goal-reaching environments from Gymnasium-Robotics and demonstrate its benefits over standard RL. The code is publicly available [1].

4.1. INTRODUCTION

Reinforcement Learning (RL) is a machine learning paradigm in which an agent learns to make decisions by interacting with its environment. The agent observes the state of the environment, takes actions, and receives feedback in the form of *rewards*. Over time, it uses this feedback to learn a policy – a strategy for choosing actions – that maximizes its performance. The standard objective in RL is to maximize the *cumulative return*, which is the total reward the agent expects to collect over time, typically with future rewards discounted to reflect their decreasing importance. This makes the reward a crucial element of the problem, as it defines the optimal decision-making policy that the agent will try to learn.

It is well known [2] that there are infinitely many ways to define the reward function under which a desired policy is optimal. Practically, however, these rewards often result in drastically different learning processes. For example, many major successes of RL required meticulous engineering of the reward: by hand [3] or by learning it from a human example [4]. Hence, designing a reward function that enables learning and corresponds to a certain optimal policy is a challenging problem in modern reinforcement learning.

Many tasks in reinforcement learning (RL) involve reaching a specific goal state – for example, navigating a maze, grasping an object, or completing a level in a game. These are known as *goal-reaching* tasks, and the most intuitive reward in such problems is sparse: the agent receives a reward only upon reaching the goal, and zero otherwise. [5–7]. Sparse reward problems are notoriously hard to solve with standard RL. A popular and simple solution is to introduce a dense surrogate reward that represents some sort of distance between the

agent and the goal [8, 9]. However, this approach is very sensitive and should be carefully tailored to each problem individually, in order to not change the induced optimal policy. Specifically, this dense artificial dense reward should *a)* increase when the agent gets closer to the goal, and *b)* not distract the agent from the reaching the goal. Designing a function that satisfies both criteria can be tricky for a human expert, as it requires estimating the (discounted) cumulative returns in various states.

In this work, we propose *max-reward RL*, where the agent optimizes the maximum reward achieved in the episode rather than the cumulative return. This paradigms makes the reward design process much more intuitive and straightforward, as it only requires that "better" states correspond to larger rewards. Hence, as long as the goal-reaching action has the highest reward, the optimal policy does not change. Besides simplifying the reward design, the maximum reward objective can also be easier to optimize for. In standard RL, learning a value of a non-terminal state involves bootstrapping, and hence has a moving target. On the contrary, in max-reward RL, bootstrapping only happens when encountered reward is larger than what was expected. Therefore, max-reward RL bootstraps less and hence, potentially, learns better.

One of the key properties of the cumulative return is that it satisfies the Bellman equation [10] and hence can be efficiently approximated and optimized by iteratively applying the Bellman operator. To make the max-reward RL approach viable, an analogous learning rule is required. However, Cui and Yu [11] prove that naively changing summation into a max operator in the standard Bellman update rule works *only* in a deterministic setting and hence cannot be used in most RL problems and algorithms.

Inspired by results from stochastic optimal control theory [12], this paper introduces a theoretically justified framework for max-reward RL in the general stochastic setting. We introduce a Bellman-like equation, prove the stochastic and deterministic policy gradient theorems, and reformulate some of the state-of-the-art algorithms (PPO, TD3) for the max-reward case. Using the Maze environment [9] with different surrogate dense rewards, we experimentally demonstrate that max-reward algorithms outperform their cumulative counterparts. Finally, experiments with a challenging Fetch environment [9] show the promise of max-reward RL in more realistic goal-reaching problems.

4.2. RELATED WORK

The first attempt to formulate max-reward RL was made by Quah and Quek [13], where the authors derived a learning rule for the maximum reward state-action value function. However, as it was shown later [14], that work made a technical error of interchanging expectation

and maximum operators. Gottipati *et al.* [14] corrected this error, but the value functions learned via their approach differ from the expected maximum reward if stochasticity is present. Independently, Wang *et al.* [15] derived a similar method in the context of planning in deterministic Markov Decision Processes (MDPs). Later, Cui and Yu [11] demonstrated that the presence of stochasticity poses a problem not only for the max-reward RL but also for other non-cumulative rewards.

There exists a parallel branch of research that (re)discovered maximum reward value functions in the context of safe RL for reach-avoid problems [16]. In their work, Fisac *et al.* [17] considered a deterministic open-loop dynamic system, where the agent's goal is to avoid constraint violations. The authors derived a contraction operator, similar to the one by Gottipati *et al.* [14], to learn the max-cost safe value function. Hsu *et al.* [18] extended this approach to reach-avoid problems, where the goal is to reach the goal while not violating constraints. Later, max-cost value functions were utilized within the safe RL context to learn the best-performing policy that does not violate the constraints [19]. The main limitation of the three aforementioned works is the same as for Gottipati *et al.* [14] – their methods only apply to deterministic environments and policies.

Effective reward design is a long-standing challenge in reinforcement learning which dates back to at least as early as 1994 [20]. In this paragraph, we briefly summarize the existing work related to the reward design problem. For further reading, we refer the reader to Eschmann [21]. Some of the big successes of RL utilize a hand-designed reward function, e.g., in the game of DOTA [3] or robots playing soccer [22]. However, manually designed rewards often lead to undesirable behavior [23]. Alternatively, the reward can be designed in an automated fashion. For example, based on state novelty to encourage exploration [24–26], by learning it from the experiences [27], or by using human data [28].

To conclude, reward design and reward shaping remain challenging topics. In this work, we propose a new way to think about the reward – the max-reward RL framework. While self-sufficient in some cases, this approach can also be combined with various existing methods for reward design.

4.3. BACKGROUND

We consider a standard reinforcement learning setup for continuous environments. An agent interacts with an MDP defined by a tuple $(\mathcal{S}, \mathcal{A}, R, P, p_0, \gamma)$, where $\mathcal{S}$ is the continuous *state space*, $\mathcal{A}$ is the continuous *action space*, and $R : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, \bar{R}]$ is a non-negative and bounded *reward function*.[1] For each state-action pair, $(s, a) \in \mathcal{S} \times \mathcal{A}$,

[1] Non-negativity of reward can be achieved in any MDP with bounded reward function.

the transition function $P(\cdot|s,a) \in \mathcal{P}(\mathcal{S})$ is a probability density function (PDF) of the next state s' and $p_0(\cdot) \in \mathcal{P}(\mathcal{S})$ is the PDF of the initial state s_0. Scalar $0 \leq \gamma < 1$ is the *discount factor*. We use $\pi : \mathcal{S} \to \mathcal{P}(\mathcal{A})$ to denote a stochastic policy and $\mu : \mathcal{S} \to \mathcal{A}$ to denote a deterministic policy. The time is discrete and starts at zero, i.e., $t \in \mathbb{N} \cup \{0\}$. For each timestep t, the state is denoted by s_t, the action by a_t, and the reward by $r_{t+1} := R(s_t, a_t, s_{t+1})$. Everywhere in the text, the expectation over policy, $\mathbb{E}_\pi$, denotes the expectation over the joint distribution of s_t, a_t, r_{t+1} for $t \in \mathbb{N} \cup \{0\}$ induced by π, P, and p_0. Sometimes, we use such notation as $\mathbb{E}_{x\sim\pi}$ (or just $\mathbb{E}_x$) to emphasize that the expectation is taken only over x.

In standard RL, the main quantity being optimized is the *cumulative return*, defined as $G_t = \sum_{i=0}^{\infty} \gamma^i r_{t+1+i}$. To maximize $\mathbb{E}_\pi[G_t]$, most RL algorithms learn state and/or state-action value functions defined as follows:

$$v^\pi(s) = \mathbb{E}_\pi\left[G_t \middle| s_t = s\right], \quad v^*(s) = \max_\pi v^\pi(s).$$

$$q^\pi(s,a) = \mathbb{E}_\pi\left[G_t \middle| \substack{s_t=s \\ a_t=a}\right], \quad q^*(s,a) = \max_\pi q^\pi(s,a).$$

Crucially, these functions are solutions to the corresponding *Bellman equations*:

$$v^\pi(s) = \mathbb{E}_{\substack{a_t \\ s_{t+1}}}\left[r_{t+1} + \gamma v^\pi(s_{t+1}) \middle| s_t = s\right]$$

$$q^\pi(s,a) = \mathbb{E}_{\substack{s_{t+1} \\ a_{t+1}}}\left[r_{t+1} + \gamma q^\pi(s_{t+1}, a_{t+1}) \middle| \substack{s_t=s \\ a_t=a}\right]$$

$$q^*(s,a) = \mathbb{E}_{s_{t+1}}\left[r_{t+1} + \gamma \max_{a'} q^*(s_{t+1}, a') \middle| \substack{s_t=s \\ a_t=a}\right]$$

The defining feature of these equations is that they can be solved by repeatedly applying *Bellman operators.* These operators are contractions and hence each of them has a unique fixed point that corresponds to one of the value functions above. For example, the optimal state-action value function $q^*(s,a)$ is the fixed point of the *Bellman optimality operator* $\mathcal{T}^*$:

$$\left(\mathbb{T}^* q\right)(s,a) = \mathbb{E}_{s_{t+1}}\left[r_{t+1} + \gamma \max_{a'} q(s_{t+1}, a') \middle| \substack{s_t=s \\ a_t=a}\right] \tag{4.1}$$

The Bellman equation is foundational for all state-of-the-art RL algorithms as it allows training neural networks to approximate value functions. Therefore, for the max-reward framework to be usable, it is necessary to derive an analog of the Bellman equation. Below, we describe such an attempt made by Gottipati *et al.* [14] and demonstrate that it is limited to purely deterministic problems.

4.3.1. DETERMINISTIC MAX-REWARD RL

Instead of cumulative return, max-reward RL aims at optimizing the *max-reward return*:

$$\hat{G}_t = \max\{r_{t+1}, \gamma r_{t+2}, \gamma^2 r_{t+3} \dots\} \tag{4.2}$$

Similarly to cumulative returns, $\hat{G}_t$ uses the discount factor γ which is necessary for learning with Bellman-like updates, as we show later. To approximate $\mathbb{E}_\pi[\hat{G}_t]$, Gottipati *et al.* [14] introduced the following definition of the state-action value functions:

$$\hat{q}^\pi_{det}(s,a) = \mathbb{E}_{\substack{s_{t+1}\\a_{t+1}}}\left[r_{t+1} \vee \gamma q(s_{t+1}, a_{t+1}) \middle|_{a_t=a}^{s_t=s}\right]$$

$$\hat{q}^*_{det}(s,a) = \mathbb{E}_{s_{t+1}}\left[r_{t+1} \vee \gamma \max_{a'} q(s_{t+1}, a') \middle|_{a_t=a}^{s_t=s}\right]$$

where $\vee$ denotes the binary max operator, i.e., $a \vee b := \max\{a, b\}$. By construction, $\hat{q}^*_{det}$ and $\hat{q}^\pi_{det}$ satisfy Bellman-like recursive equations. In their work, Gottipati *et al.* [14] proved that the following operator is a contraction:

$$(\hat{\mathbb{T}}^*_{det} q)(s,a) = \mathop{\mathbb{E}}_{s_{t+1}}\left[r_{t+1} \vee \gamma \max_{a'} q(s_{t+1}, a') \middle|_{a_t=a}^{s_t=s}\right] \tag{4.3}$$

Therefore, $\hat{q}^*_{det}$ is the unique fixed point of $\hat{\mathcal{T}}^*_{det}$ and can be learned, e.g., with Q-learning.

Chain environment example. Before going into the limitations of the approach above, we conduct a simple experiment to motivate the use of max-reward reinforcement learning. We show that max-reward RL is a better approach in a goal-reaching problem where the agent needs to learn to reach the goal state. Specifically, it dominates the standard cumulative RL when transitions into the goal state occur infrequently in the training data, which is often the case in larger-scale goal-reaching problems.

Consider the five-state chain environment in Figure 4.1. Transitions leading into s_4 have reward of 1, transitions into s_2 have a reward parametrized by $x \in (0, 1)$, and other rewards are zero. Hence, the optimal policy, concerning both max-reward and cumulative returns, is to go to s_4 and stay there. We run tabular $Q-$value iteration algorithm using standard (Eq. (4.1)) and max-reward (Eq. (4.3)) Bellman operators for different values of the intermediate reward x. In each training epoch, we iterate over all possible transitions. For each transition, we compute the target value using one of the Bellman operators and update the Q-table. Crucially, we *randomly skip some of the transitions into* s_4 *with a certain probability*. In the experiment, we consider four values

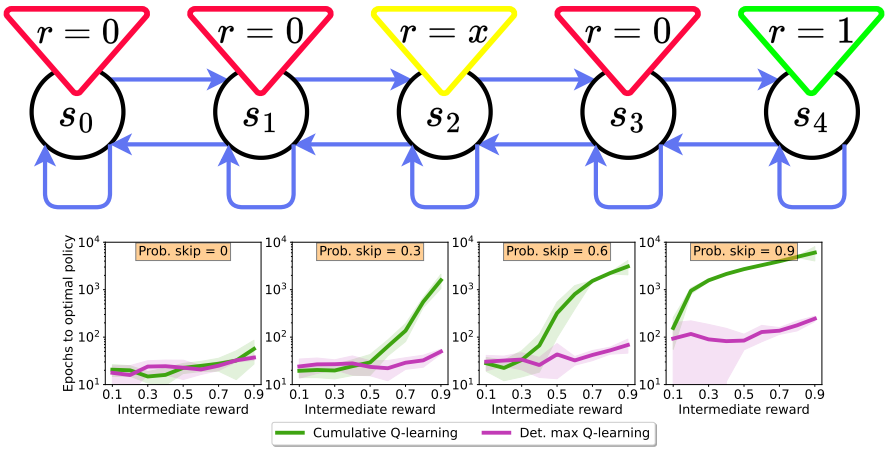


Figure 4.1: Five-state chain MDP with three actions (*left, stay, right*) in each state and the results for cumulative (in green) and max-reward (in violet) value iteration. The $y-$axis is the number of training epochs to recover the optimal policy; the $x-$axis are the values of the intermediate reward x. The four panels correspond to different probabilities of skipping transitions into s_4.

for the skip probability – $p_{skip} \in \{0, 0.3, 0.6, 0.9\}$. During training, when a transition into s_4 is sampled, the Q-table is updated with probability $1 - p_{skip}$ and otherwise left unchanged. Transitions into other states are never skipped. In this way, we can control how often the agent is exposed to the transitions into the optimal state and thereby simulate problems where goal-reaching transitions are rarely encountered.

The results in Figure 4.1 indicate that for larger values of the skip probability, the max-reward approach converges to the optimal policy significantly faster than the cumulative approach. We believe that this phenomenon can be explained by differences in bootstrapping. In standard RL, the target for the $q-$value is a sum of the immediate reward and the $q-$value at the next timestep. Therefore, this target changes in each epoch until convergence. In the max-reward case, on the other hand, the target in the max-reward state is just the reward and does not change with time. This example suggests that the max-reward approach is a better choice in environments where the task of the agent is to reach the goal state.

Issues when stochasticity is present. Unfortunately, the max-reward approach described above has a serious theoretical drawback. Expanding the definition of $\hat{q}^*_{det}$ for more timesteps, we obtain a nested sequence of non-interchangeable $\vee$ and $\mathbb{E}$:

$$\hat{q}^*_{det}(s,a) = \mathbb{E}_{\pi^*}\Big[r_{t+1} \vee \gamma \mathbb{E}_{\pi^*}\Big[r_{t+2} \vee \dots\Big]\Big|^{s_t=s}_{a_t=a}\Big]$$

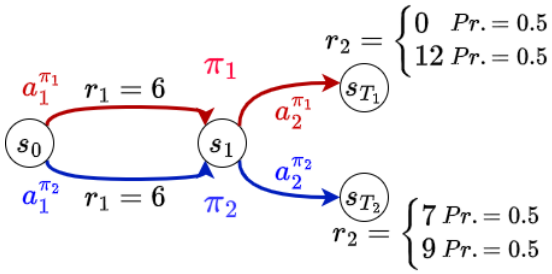


Figure 4.2: A three-state MDP with deterministic transitions and stochastic rewards. Two policies, π_1 and π_2, perform first action a_1, but then have different a_2, resulting in different reward distributions.

4

Using Jensen's inequality [29], we conclude the following:

$$\hat{q}^*_{det}(s,a) \leq \mathbb{E}_{\pi^*}\left[\hat{G}_t | ^{s_t=s}_{a_t=a}\right] \tag{4.4}$$

When both the policy and the transition model are deterministic, Eq. (4.4) becomes an equality. However, if stochasticity is present, the value of $\hat{q}^*_{det}(s,a)$ is merely a lower bound of the expected return. Hence, it can induce suboptimal policies.

In Figure 4.2, we show an example where the policy maximizing $\hat{q}^*_{det}$ is suboptimal. The figure demonstrates a three-state MDP and two policies, π_1 (red arrows) and π_2 (blue arrows). Let $\gamma = 1$ for simplicity. For the state s_0, the expected max-reward return is higher for the policy π_1 :

$$\mathbb{E}_{\pi_1}[\hat{G}_0] = \mathbb{E}_{\pi_1}[r_1 \vee r_2] = 9 >$$
$$\mathbb{E}_{\pi_2}[\hat{G}_0] = \mathbb{E}_{\pi_2}[r_1 \vee r_2] = 8$$

So π_1 is better in terms of the expected max-reward return, but the value functions have the following values:

$$\hat{q}^{\pi_1}_{det}(s_0) = \mathbb{E}_{\pi_1}\left[r_1 \vee \mathbb{E}_{\pi_1}[r_2]\right] = \mathbb{E}_{\pi_1}[r_1 \vee 6] = 6$$

$$\hat{q}^{\pi_2}_{det}(s_0) = \mathbb{E}_{\pi_2}\left[r_1 \vee \mathbb{E}_{\pi_2}[r_2]\right] = \mathbb{E}_{\pi_2}[r_1 \vee 8] = 8$$

Based on the values of $\hat{q}^{\pi}_{det}$, we would conclude that π_2 is better, which we already showed to be incorrect. This example demonstrates that even in a simple stochastic environment, the operator $\hat{T}^*_{det}$ can lead to incorrect policies. Therefore, it is an open question whether there exists a Bellman-like operator that would enable learning max-reward returns in the stochastic setting.

4.4. MAX-REWARD RL

In this section, we introduce a novel approach to max-reward RL that is theoretically sound, works for both stochastic and deterministic cases,

and can be combined with state-of-the-art RL algorithms. First, we expand the definition of the max-reward return given in Eq. (4.2):

$$\mathbb{E}_\pi[\hat{G}_t] = \mathbb{E}_\pi\big[r_{t+1} \vee \gamma \hat{G}_{t+1}\big] \tag{4.5}$$

Since $\mathbb{E}$ and $\vee$ do not commute, it is impossible to extract the term $\mathbb{E}_\pi\big[G_{t+1}\big]$ on the right-hand side of Eq. (4.5). Because of that, we cannot obtain an equation involving only $\mathbb{E}_\pi\big[G_t\big]$, $\mathbb{E}_\pi\big[G_{t+1}\big]$, and r_{t+1}. Instead, we will utilize an approach from stochastic optimal control theory [12] and define the max-reward value function using an auxiliary variable that allows propagating information between timesteps:

Definition 4.4.1. Let $y \in \mathbb{R}$ be an auxiliary real variable. The *max-reward value functions* are defined as follows:

$$\hat{v}^\pi(s, y) = \mathbb{E}_\pi[y \vee \hat{G}_t \big|_{s_t=s}]$$

$$\hat{q}^\pi(s, a, y) = \mathbb{E}_\pi[y \vee \hat{G}_t \big|^{s_t=s}_{a_t=a}]$$

Since reward is lower-bounded, $r_{t+1} \geq 0$, we can always recover the expected max-reward return $\mathbb{E}_\pi\big[\hat{G}_t\big]$ by substituting $y = 0$ into the value functions:

$$\begin{aligned} \hat{v}^\pi(s, 0) &= \mathbb{E}_\pi[\hat{G}_t \big|_{s_t=s}] \\ \hat{q}^\pi(s, a, 0) &= \mathbb{E}_\pi[\hat{G}_t \big|^{s_t=s}_{a_t=a}] \end{aligned} \tag{4.6}$$

Hence, if we find an efficient method of learning the max-reward value functions, we will be able to optimize $\mathbb{E}_\pi[\hat{G}_t]$.

The auxiliary variable y is crucial when dealing with the max-reward returns. When we look at the value of the state s' from the perspective of the state s, we must consider the immediate reward $r = r(s, a, s')$. Specifically, we should treat low reward trajectories from s' as if they still yield the reward of r. Expanding upon this observation, we conclude that maximization of the maximum reward requires propagating information about the past rewards. This is achieved via the auxiliary variable y.

By combining the definition of the max-reward value functions with Eq. (4.5), we obtain the following recursive equations:

Lemma 4.4.2. *Let $y \in \mathbb{R}$ and let $y' := \frac{R(s,a,s_{t+1}) \vee y}{\gamma}$. Then, the max-reward value functions are subject to the following Bellman-like equations:*

$$\hat{v}^\pi(s, y) = \gamma \mathbb{E}_{\substack{a_t \\ s_{t+1}}}\big[y' \vee \hat{v}^\pi(s_{t+1}, y') \big|_{s_t=s}\big]$$

$$\hat{q}^\pi(s, a, y) = \gamma \mathbb{E}_{\substack{s_{t+1} \\ a_{t+1}}}\big[y' \vee \hat{q}^\pi(s_{t+1}, a_{t+1}, y') \big|^{s_t=s}_{a_t=a}\big]$$

Proof of this lemma, as well as all other proofs, can be found in Appendix 4.6.1. The extra term $y' \vee$ might seem redundant, but it is important since it enforces the boundary conditions. Without it, the

functions $v \equiv 0$ and $q \equiv 0$ would be solutions to these equations. Using Lemma 4.4.2, we can define Bellman-like operators for the max-reward value functions:

Definition 4.4.3. Let $v : \mathcal{S} \times \mathbb{R} \to \mathbb{R}$, $q : \mathcal{S} \times \mathcal{A} \times \mathbb{R} \to \mathbb{R}$ be real-valued functions and let $y' := \frac{R(s,a,s_{t+1}) \vee y}{\gamma}$. Then, the *max-reward Bellman operator* $\hat{\mathcal{T}}^\pi$ is defined as follows:

$$\hat{\mathcal{T}}^\pi v(s, y) := \gamma \mathbb{E}_{\substack{a_t \\ s_{t+1}}} \left[y' \vee v(s_{t+1}, y') \middle| s_t = s \right]$$

$$\hat{\mathcal{T}}^\pi q(s, a, y) := \gamma \mathbb{E}_{\substack{s_{t+1} \\ a_{t+1}}} \left[y' \vee q(s_{t+1}, a_{t+1}, y') \middle| \substack{s_t = s \\ a_t = a} \right]$$

4

In the following theorem, we prove that this operator is a contraction and that the max-reward state and state-action value functions are its fixed points.

Theorem 4.4.4. *$\hat{\mathcal{T}}^\pi$ is a γ−contraction with respect to the L_∞ norm, and $\hat{v}^\pi$(or $\hat{q}^\pi$) is its fixed point.*

Theorem 4.4.4 implies that the max-reward value functions can be learned in the same way as the standard value functions – by sampling from the environment and applying Bellman operators. In the next section, we define the objective function of the max-reward RL problem and discuss how the presence of the auxiliary variable y impacts the notion of optimal policy.

4.4.1. MAX-REWARD OBJECTIVE

Similarly to standard RL, the main objective in the max-reward RL problems is to maximize the expected (max-reward) return from the initial state, defined as follows:

$$\hat{J}(\pi) = \mathbb{E}_{s_0 \sim p_0} \left[\hat{v}^\pi(s_0, 0) \right] \tag{4.7}$$

Then, the optimal policy is naturally defined as :

$$\pi^* = \arg\max_\pi \hat{J}(\pi). \tag{4.8}$$

To better understand the properties of the max-reward optimal policy, consider again the MDP in Figure 4.2. Let $\gamma = 1$. Then, the values of the objective function for π_1 and π_2 can be computed as follows:

$$\hat{J}(\pi_1) = \mathbb{E}_{\pi_1}[6 \vee r_2] = 9$$

$$\hat{J}(\pi_2) = \mathbb{E}_{\pi_2}[6 \vee r_2] = 8$$

Hence, π_1 is optimal. However, if we consider the max-reward return from $t = 1$, we have

$$\mathbb{E}_{\pi_1}[G_1] = \frac{12+0}{2} = 6 \quad \mathbb{E}_{\pi_2}[G_1] = \frac{9+7}{2} = 8$$

and hence π_2 obtains higher expected max-reward return starting at $s = s_1$. Seemingly, there is a contradiction: π_1 is optimal but π_2 is better from the state s_1. However, the explanation is simple: the maximum reward is the highest reward encountered anywhere along the trajectory. An optimal decision thus not only depends on the current state, as with the cumulative reward, but also on the maximum reward that has been acquired thus far. In the example, if we start from s_1, then we haven't encountered any reward yet. Hence, following π_1, we will have $r_2 = 0$ as the maximum reward half of the time. If we start from s_0, we receive a reward of $r_1 = 6$ when going to s_1. Then, the maximum reward will not be lower than 6, even if we get $r_2 = 0$. Thus, we conclude:

In max-reward RL, the optimal policy π^* maximizing $\hat{J}(\cdot)$ should depend not only on the current state, but also on the rewards obtained so far.

To formalize this observation, we introduce additional notation. We define the *extended state space* as $\hat{\mathcal{S}} := \mathcal{S} \times \mathbb{R}$ and we denote extended states by $\hat{s} = (s, y)$, $s \in \mathcal{S}$, $y \in \mathbb{R}$. Then, for an extended state $(s, y) \in \hat{\mathcal{S}}$ and for an action $a \in \mathcal{A}$, the *extended transition model* $\hat{P}(\cdot, \cdot|s, y, a)$ is a PDF over $(s', y') \in \hat{\mathcal{S}}$, defined as

$$\hat{P}(s', y'|s, y, a) = P(s'|s, a)\, \delta\Big(y' - \frac{R(s, a, s') \vee y}{\gamma}\Big)$$

where $\delta(\cdot)$ is the Dirac delta function. The initial distribution of (s_0, y_0) is given by $\hat{p}_0(s_0, y_0) = p(s_0)\delta(y_0)$ thereby ensuring $y_0 \equiv 0$. Combining everything, we introduce the following definition:

Definition 4.4.5. Let $M = (\mathcal{S}, \mathcal{A}, R, P, p_0, \gamma)$ be an MDP. Then, the *extended max-reward MDP* is an MDP $\hat{M}$ given by the tuple $(\hat{\mathcal{S}}, \mathcal{A}, R, \hat{P}, \hat{p_0}, \gamma)$.

Essentially, the extended MDP defined above tracks the (inversely) discounted maximum reward obtained so far. For example, if the maximum reward so far is r_1, then the extended state at timestep t is $(s_t, \frac{r_1}{\gamma^t})$. Hence, to improve $\hat{J}$, we need $r_{t+1} > \frac{r_1}{\gamma^t}$.

Using the notion of extended MDP, we can redefine policy for the max-reward RL:

Definition 4.4.6. Let M be an MDP and let $\hat{M}$ be its induced extended max-reward MDP. Then, any policy $\hat{\pi}$ in $\hat{M}$ is an *extended max-reward policy*.

After we have defined optimality in the max-reward sense, we can introduce the max-reward Bellman optimality operator:

Definition 4.4.7. Let $q : \mathcal{S} \times \mathcal{A} \times \mathbb{R} \to \mathbb{R}$ be a real-valued function and let $y' := \frac{R(s,a,s_{t+1}) \vee y}{\gamma}$. Then, the *max-reward Bellman optimality operator* $\hat{\mathcal{T}}^*$ is defined as follows:

$$\hat{\mathcal{T}}^* q(s, a, y) := \gamma \mathbb{E}_{s_{t+1}}\left[y' \vee \max_{a'} q(s_{t+1}, a', y') \Big|_{a_t=a}^{s_t=s}\right]$$

Similarly to $\hat{T}^\pi$, this operator is also a contraction:

Theorem 4.4.8. *$\hat{\mathcal{T}}^*$ is a $\gamma-$contraction with respect to the L_∞ norm, and $\hat{q}^*$ is its fixed point.*

We have most of the pieces of the max-reward RL framework. We established that it operates on the extended max-reward MDP $\hat{M}$, where the extended states preserve information about the past rewards. Then, both the max-reward optimal and on-policy value functions can be learned by sampling transitions from $\hat{M}$. Therefore, all DQN-based methods [30] can be used under the max-reward RL paradigm directly. However, most state-of-the-art RL algorithms utilize policies parametrized by neural networks. This is possible due to the policy gradient theorems [31, 32], as they allow estimating the objective function gradient with respect to the policy parameters via sampling. In the next section, we formulate and prove max-reward policy gradient theorems for both deterministic and stochastic extended max-reward policies.

4.4.2. POLICY GRADIENT THEOREMS

First, we define $\hat{p}_t^{\hat{\pi}}(s_0, y_0, s, y)$ – the probability measure of arriving in the extended state (s, y) after t timesteps, starting from (s_0, y_0) and executing the extended policy $\hat{\pi}$. Let

$$\hat{P}^{\hat{\pi}}(s', y'|s, y) = \int_a \hat{\pi}(a|s, y)\hat{P}(s', y'|s, y, a)da$$

be the "on-policy" transition model. Then, $\hat{p}_t^{\hat{\pi}}(s_0, y_0, s, y)$ is defined as follows:

$$\hat{p}_0(s_0, y_0, s, y) = \delta(s - s_0)\delta(y - y_0)$$

$$\hat{p}_t^{\hat{\pi}}(s_0, y_0, s, y) = \int_{\tilde{s},\tilde{y}} \hat{p}_{t-1}^{\hat{\pi}}(s_0, y_0, \tilde{s}, \tilde{y})\, \hat{P}^{\hat{\pi}}(s, y|\tilde{s}, \tilde{y})\, d\tilde{s}\, d\tilde{y}$$

The discounted stationary state distribution of an extended max-reward MDP is then given by

$$\hat{d}^{\hat{\pi}}(s,y) = \int_{s_0,y_0} \hat{p}_0(s_0,y_0) \sum_{t=0}^{\infty} \gamma^t \hat{p}_t^{\hat{\pi}}(s_0,y_0,s,y)\, ds_0\, dy_0.$$

As such, $\hat{d}^{\hat{\pi}}$ is not a distribution. However, it can be normalized into one by dividing it by $C = \int_{s,y} \hat{d}^{\hat{\pi}}(s,y)\, ds\, dy$.

Finally, we can formulate and prove the max-reward policy gradient theorems. Consider a neural network with weights θ that represents a stochastic policy. Then, we have the following result:

Theorem 4.4.9. *Let $\hat{\pi}_\theta : \mathcal{S} \times \mathbb{R} \to \mathcal{P}(\mathcal{A})$ be a stochastic extended max-reward policy parameterized with θ. Then, the following holds for $\nabla_\theta \hat{J}(\theta)$:*

$$\nabla_\theta \hat{J}(\theta) \propto \mathbb{E}_{\substack{(s,y)\sim\hat{d}^{\hat{\pi}} \\ a\sim\hat{\pi}_\theta}} \left[\hat{q}^{\hat{\pi}_\theta}(s,a,y) \nabla_\theta \ln \hat{\pi}_\theta(a|s,y) \right]$$

The deterministic max-reward policy gradient follows from the stochastic version:

Corollary 4.4.10. *Let $\hat{\mu}_\theta : \mathcal{S} \times \mathbb{R} \to \mathcal{A}$ be a deterministic extended max-reward policy parameterized with θ. Then $\nabla_\theta \hat{J}(\theta)$ can be computed as follows:*

$$\nabla_\theta \hat{J}(\theta) \propto \mathbb{E}_{\hat{d}^{\hat{\mu}}} \left[\nabla_\theta \hat{\mu}_\theta(s,y) \nabla_a \hat{q}^{\hat{\mu}_\theta}(s,a,y)|_{a=\mu_\theta(s,y)} \right]$$

The policy gradient theorems allow us to use various algorithms from standard RL, such as REINFORCE [33], A2C [31], A3C [34], TRPO [35], PPO [36], DDPG [37], and TD3 [38], to optimize maximum rewards. In this work, we focus on PPO and TD3, as they are considered to be the best-performing algorithms within their corresponding families. For max-reward PPO, the only difference compared to the standard version is that the advantage estimation uses max-reward returns. For max-reward TD3, the target value for the Q functions is computed using the max-reward Bellman optimality operator (4.4.7). In Appendix, we provide descriptions of max-reward TD3 and PPO in pseudocode.

Figure 4.3: *Left:* Single-goal maze, where the goal (red ball) is always in the same location. *Right:* Two-goals maze with two spawn locations of the goal (red balls).

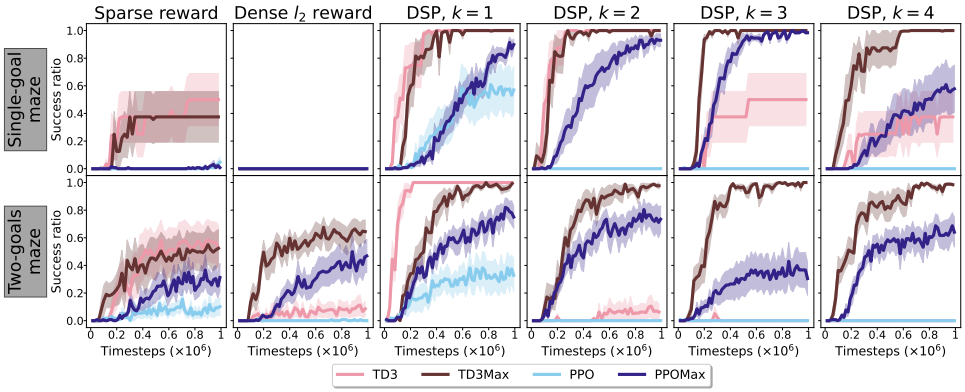


Figure 4.4: Learning curves of TD3, max-reward TD3, PPO, and max-reward PPO on two different mazes. The y—axis is the success ratio, the x—axis is the number of environmental timesteps. The shaded area is the standard error of the mean. Columns correspond to different reward functions.

4.5. EXPERIMENTS

To empirically evaluate the benefits of using the maximum instead of the cumulative reward, we compare the max-reward TD3 and PPO with their cumulative counterparts using two goal-reaching environments from Gymnasium-Robotics [9] under different dense reward functions.

4.5.1. MAZE WITH SHORTEST PATH REWARDS

First, we consider the Maze environment from Gym Robotics [9] illustrated in Figure 4.3, where the agent controls a ball by applying acceleration in two dimensions. The objective is to reach the goal position in the maze. Episodes last 1000 timesteps and there are no terminal states. We use two mazes: *single-goal* maze, where the goal is always in the same location, and the *two-goals* maze where at each episode the goal location is chosen randomly from the two possible options. The main metric in this environment is *success ratio* – a binary value indicating whether the goal was reached during the episode.

We consider several reward functions that induce the same optimal policy of reaching and then remaining in the goal state (due to absence of terminal states):

1. *Sparse reward* – only reaching the goal is rewarded with $r = 1$.
2. *Dense l_2 reward* – default dense reward, defined as the exponent of the negative of the l_2-distance to the goal. Reaching the goal is rewarded with $r = 1$.

3. *Discrete shortest path (DSP)* – our custom reward that represents the true, topology aware distance to the goal. To compute it, the maze is split into $n \times m$ cells. Then, the distance matrix $D \in \mathbb{R}^{n \times m}$ is computed such that for each cell (i, j), $D[i, j]$ is the number of cells between (i, j) and the goal-containing cell. The DSP reward with parameter $k \in \mathbb{N}$ is then defined as

$$r^k_{dsp}(i, j) = \begin{cases} \beta^{D[i,j]+1}, & \text{if } D[i, j] = 0 \mod k \\ 0, & \text{otherwise} \end{cases}$$

where $\beta \in (0, 1)$ is a hyperparameter. The value of k controls the sparsity of the reward, i.e., for larger k fewer cells have a non-zero reward. Reaching the goal is rewarded with $r = 1$.

For the DSP reward, we first tune the value of β by running standard TD3 and PPO on the single-goal maze. We set $k = 1$ and run 10 random seeds for each algorithm for $\beta \in \{0.65, 0.7, 0.75, 0.8, 0.85, 0.9, 0.95\}$. Additionally, we test the negative version of the DSP reward, $r^k_{dsp}(i, j) - 1$, which, in theory, should cause better exploration. For TD3, the best-performing reward was the negative DSP with $\beta_{TD3} = 0.9$, and for PPO – negative DSP with $\beta_{PPO} = 0.95$. In all other runs involving DSP reward we use these values of β.

Finally, we compare TD3, PPO, max-reward TD3, and max-reward PPO on the single-goal and two-goals mazes using sparse, dense l_2, and DSP reward for $k = 1, 2, 3, 4$ (for cumulative methods, negative DSP reward is used). Figure 4.4 demonstrates the learning curves. The sparse reward performs inconsistently due to insufficient exploration. Dense l_2 reward has local maximums (especially in the single-goal maze) and its performance greatly depends on the maze topology. The DSP reward, which represents the true distance to the goal, overall performs better.

Importantly, we see that the max-reward approaches work for all values of k, while the standard RL methods do not. For larger k, the reward becomes sparser, and cumulative approaches tend to converge to suboptimal policies. We believe that the nature of this phenomenon is the same as in the chain environment example discussed in Section 4.3.1. Specifically, the Maze environment can be seen as a larger chain with multiple intermediate rewards. During training, all methods quickly learn to stay in one of the cells with non-zero reward. Then, to update the policy, samples of transitions to a better state are needed. For larger k, these transitions become less frequent, as the cells with non-zero reward become further from one another. In line with the chain environment results, max-reward methods require fewer such transitions and therefore perform more efficiently.

Another potential reason for the superiority of max-reward methods lies in the way how they handle local optima. Since max-reward policy is conditioned on the discounted max-reward so far, y, it has no incentive

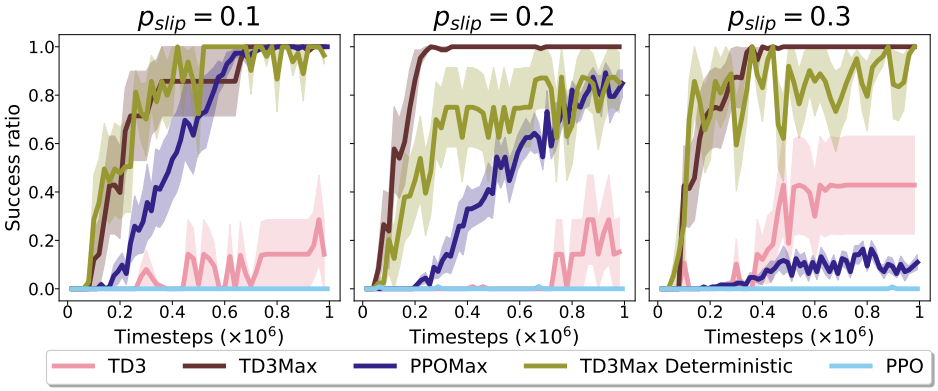


Figure 4.5: Learning curves of TD3, max-reward TD3, deterministic max-reward TD3, PPO, and max-reward PPO on a stochastic version of the single-goal maze with DSP reward, $k = 3$. The $y-$axis is the success ratio, the $x-$axis is the number of environmental timesteps. The shaded area is the standard error of the mean.

to stay in the local optima. As y "remembers" the reward at a local optimum, any trajectory leaving this optimum is at least as good as staying in the optimum. Combined with exploration techniques, e.g., entropy regularization in PPO, this causes the agent to leave local optima after visiting them.

Stochastic Maze. One of the strengths of our RL formulation is that it works with stochastic environments and/or policies. To experimentally verify that, we conduct an additional experiment using a stochastic variation of the single-goal maze. Specifically, we introduce a parameter p_{slip} which regulates the level of stochasticity. Whenever the agent makes an action, it is replaced with a random action with probability p_{slip}. We compare max-reward and standard versions of TD3 and PPO on this environment. Additionally, we implement and test *deterministic* max-reward TD3 [14]. In this experiment, we use the DSP reward with $k = 3$, as it is a case where the max-reward paradigm demonstrates improvement over standard RL in a deterministic Maze. The results presented in Figure 4.5 confirm the theory: our max-reward TD3 solves this stochastic environment while the deterministic max-reward TD3 is highly inconsistent. Therefore, we conclude that our method indeed can be used for stochastic environments.

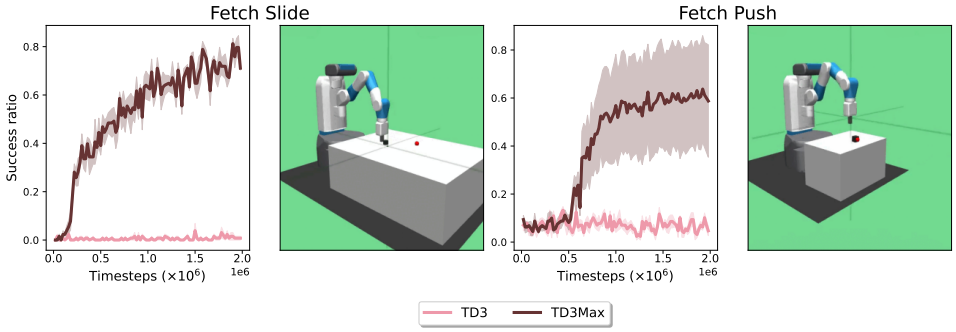


Figure 4.6: Success ratio for standard (light red) and max-reward (dark-red) TD3 in *Fetch Slide* (left) and *Fetch Push* (right) environments.

4.5.2. FETCH ENVIRONMENT

In the second experiment, we consider more challenging robotics problems. Specifically, we study the *Fetch-Slide* and *Fetch-Push* environments depicted in Figure 4.6. The agent controls a 7-DoF manipulator and its goal is to move the puck into the target location. In *Fetch-Slide*, the goal is located beyond agent's reach and hence it needs to slide the puck into the goal. In *Fetch-Push*, the agent needs to push the puck into the goal which can be anywhere on the platform. Each episode is truncated after 100 timesteps and there are no terminal states. We use the standard dense reward for this problem defined as negative of the l_2−distance between the puck and the goal. The performance metric for this environment is again the success ratio – a binary value that indicates whether the goal was reached during the episode. This environment is known to be challenging for standard RL, and it cannot be solved without special approaches [5].

The plot in Figure 4.6 demonstrates that the max-reward TD3 achieves a goal-reaching policy in both environments, while the standard version, in line with the prior work, fails to learn completely. We believe that this happens due to the difference in bootstrapping mentioned earlier. The environment is complex and multidimensional and the goal-reaching transitions are rare which makes the learning problem really hard for the standard methods. We believe that this experiment shows the great potential of max-reward RL in more realistic goal-reaching environments.

4.6. CONCLUSIONS AND FUTURE WORK

In this work, we provide a theoretical description of the max-reward reinforcement learning paradigm and verify it experimentally. Our theoretical contributions include a novel formulation of the max-reward value functions and a Bellman-like contraction operator that enables

efficient learning. Besides, we prove the policy gradient theorems for max-reward policies and hence enable using the state-of-the-art RL algorithms in the context of max-reward RL.

In the experiments with two robotic environments, we show that max-reward RL works better for sparse reward problems with surrogate dense reward. This result confirms our intuition that maximum reward is a better choice for goal-reaching environments. Moreover, we demonstrate that our max-reward RL, unlike prior work, is also consistent in stochastic environments.

Qualitatively, we believe that the main strengths of the max-reward algorithms can be summarized as follows:

1. In max-reward RL, bootstrapping works differently than in the standard RL. Specifically, it allows for more efficient propagation of reward from the goal states.

2. Max-reward agents are less prone to getting stuck in local optima. Since the maximum reward obtained so far is a part of the extended state space, the agents do not have any incentive to stay in these optima.

3. Due to the auxiliary variable y, max-reward value functions are of distributional nature. Specifically, differences of the form $\hat{v}(s, y + \Delta) - \hat{v}(s, y)$ can be used to approximate the distribution of the max-reward return. As reported in prior work [39], learning distributional value functions can have positive impact even in deterministic problems.

In future work, we aim to study how max-reward RL can be combined with the existing methods for automated reward design and explore its potential in other problems.

ACKNOWLEDGEMENTS

We would like to acknowledge Delft University of Technology for providing the resources and support necessary for this research. The collaborative environment greatly contributed to the development and success of this work. In particular, we are very thankful to Jinke He for reading the draft of this paper and providing helpful feedback.

Additionally, we thank the anonymous reviewers for their thorough and constructive feedback. Their insightful comments and suggestions have significantly improved the quality of this paper.

AUTHORS' CONTRIBUTIONS:

G.V. conceived the research, developed the methodology, conducted all experiments, and prepared the initial manuscript. W.B. and

M.d.W. provided critical feedback throughout the research process and contributed significantly to the revision and refinement of the manuscript.

REFERENCES

[1] G. Veviurko. *veviurko/To-the-Max: To the Max: Reinventing Reward in Reinforcement Learning*. Version publication. Jan. 2025. doi: 10.5281/zenodo.14645337. url: https://doi.org/10.5281/zenodo.14645337.

[2] A. Y. Ng, D. Harada, and S. Russell. "Policy invariance under reward transformations: Theory and application to reward shaping". In: *Icml*. Vol. 99. Citeseer. 1999, pp. 278–287.

[3] C. Berner, G. Brockman, B. Chan, V. Cheung, P. Debiak, C. Dennison, D. Farhi, Q. Fischer, S. Hashme, C. Hesse, *et al.* "Dota 2 with large scale deep reinforcement learning". In: *arXiv preprint arXiv:1912.06680* (2019).

[4] O. Vinyals, I. Babuschkin, W. M. Czarnecki, M. Mathieu, A. Dudzik, J. Chung, D. H. Choi, R. Powell, T. Ewalds, P. Georgiev, *et al.* "Grandmaster level in StarCraft II using multi-agent reinforcement learning". In: *Nature* 575.7782 (2019), pp. 350–354.

[5] M. Plappert, M. Andrychowicz, A. Ray, B. McGrew, B. Baker, G. Powell, J. Schneider, J. Tobin, M. Chociej, P. Welinder, *et al.* "Multi-goal reinforcement learning: Challenging robotics environments and request for research". In: *arXiv preprint arXiv:1802.09464* (2018).

[6] C. Florensa, D. Held, X. Geng, and P. Abbeel. *Automatic Goal Generation for Reinforcement Learning Agents*. 2018. arXiv: 1705.06366 [cs.LG].

[7] D. Ghosh, A. Gupta, A. Reddy, J. Fu, C. Devin, B. Eysenbach, and S. Levine. *Learning to Reach Goals via Iterated Supervised Learning*. 2020. arXiv: 1912.06088 [cs.LG].

[8] M. Towers, J. K. Terry, A. Kwiatkowski, J. U. Balis, G. d. Cola, T. Deleu, M. Goulão, A. Kallinteris, A. KG, M. Krimmel, R. Perez-Vicente, A. Pierré, S. Schulhoff, J. J. Tai, A. T. J. Shen, and O. G. Younis. *Gymnasium*. Mar. 2023. doi: 10.5281/zenodo.8127026. url: https://zenodo.org/record/8127025 (visited on 07/08/2023).

[9] R. de Lazcano, K. Andreas, J. J. Tai, S. R. Lee, and J. Terry. *Gymnasium Robotics*. Version 1.2.4. 2023. url: http://github.com/Farama-Foundation/Gymnasium-Robotics.

[10] R. Bellman. "Some applications of the theory of dynamic programming—a review". In: *Journal of the Operations Research Society of America* 2.3 (1954), pp. 275–288.

[11] W. Cui and W. Yu. "Reinforcement Learning with Non-Cumulative Objective". In: *IEEE Transactions on Machine Learning in Communications and Networking* (2023).

[12] A. Kröner, A. Picarelli, and H. Zidani. “Infinite Horizon Stochastic Optimal Control Problems with Running Maximum Cost”. In: *SIAM J. Control Optim.* 56.5 (Jan. 2018), pp. 3296–3319.

[13] K. Quah and C. Quek. “Maximum reward reinforcement learning: A non-cumulative reward criterion”. In: *Expert Systems with Applications* 31.2 (2006), pp. 351–359. issn: 0957-4174. doi: https://doi.org/10.1016/j.eswa.2005.09.054.

[14] S. K. Gottipati, Y. Pathak, R. Nuttall, Sahir, R. Chunduru, A. Touati, S. G. Subramanian, M. E. Taylor, and S. Chandar. *Maximum Reward Formulation In Reinforcement Learning*. 2020. arXiv: 2010.03744 [cs.LG].

[15] R. Wang, P. Zhong, S. S. Du, R. R. Salakhutdinov, and L. Yang. “Planning with general objective functions: Going beyond total rewards”. In: *Advances in Neural Information Processing Systems* 33 (2020), pp. 14486–14497.

[16] J. F. Fisac, M. Chen, C. J. Tomlin, and S. S. Sastry. *Reach-Avoid Problems with Time-Varying Dynamics, Targets and Constraints*. 2014. arXiv: 1410.6445 [math.OC].

[17] J. F. Fisac, N. F. Lugovoy, V. Rubies-Royo, S. Ghosh, and C. J. Tomlin. “Bridging Hamilton-Jacobi Safety Analysis and Reinforcement Learning”. In: *2019 International Conference on Robotics and Automation (ICRA)*. May 2019, pp. 8550–8556.

[18] K.-C. Hsu, V. Rubies-Royo, C. J. Tomlin, and J. F. Fisac. “Safety and Liveness Guarantees through Reach-Avoid Reinforcement Learning”. In: (Dec. 2021). arXiv: 2112.12288 [cs.LG].

[19] D. Yu, H. Ma, S. E. Li, and J. Chen. “Reachability Constrained Reinforcement Learning”. In: (May 2022). arXiv: 2205.07536 [cs.LG].

[20] M. J. Mataric. “Reward functions for accelerated learning”. In: *Machine learning proceedings 1994*. Elsevier, 1994, pp. 181–189.

[21] J. Eschmann. “Reward function design in reinforcement learning”. In: *Reinforcement Learning Algorithms: Analysis and Applications* (2021), pp. 25–33.

[22] T. Haarnoja, B. Moran, G. Lever, S. H. Huang, D. Tirumala, M. Wulfmeier, J. Humplik, S. Tunyasuvunakool, N. Y. Siegel, R. Hafner, M. Bloesch, K. Hartikainen, A. Byravan, L. Hasenclever, Y. Tassa, F. Sadeghi, N. Batchelor, F. Casarini, S. Saliceti, C. Game, N. Sreendra, K. Patel, M. Gwira, A. Huber, N. Hurley, F. Nori, R. Hadsell, and N. Heess. *Learning Agile Soccer Skills for a Bipedal Robot with Deep Reinforcement Learning*. 2023. arXiv: 2304.13653 [cs.RO].

[23] V. Krakovna, J. Uesato, V. Mikulik, M. Rahtz, T. Everitt, R. Kumar, Z. Kenton, J. Leike, and S. Legg. “Specification gaming: the flip side of AI ingenuity”. In: (2020).

[24] H. Tang, R. Houthooft, D. Foote, A. Stooke, X. Chen, Y. Duan, J. Schulman, F. D. Turck, and P. Abbeel. *#Exploration: A Study of Count-Based Exploration for Deep Reinforcement Learning*. 2017. arXiv: 1611.04717 [cs.AI].

[25] D. Pathak, P. Agrawal, A. A. Efros, and T. Darrell. *Curiosity-driven Exploration by Self-supervised Prediction*. 2017. arXiv: 1705.05363 [cs.LG].

[26] Y. Burda, H. Edwards, A. Storkey, and O. Klimov. *Exploration by Random Network Distillation*. 2018. arXiv: 1810.12894 [cs.LG].

[27] A. Trott, S. Zheng, C. Xiong, and R. Socher. *Keeping Your Distance: Solving Sparse Reward Tasks Using Self-Balancing Shaped Rewards*. 2019. arXiv: 1911.01417 [cs.AI].

[28] B. Ibarz, J. Leike, T. Pohlen, G. Irving, S. Legg, and D. Amodei. *Reward learning from human preferences and demonstrations in Atari*. 2018. arXiv: 1811.06521 [cs.LG].

[29] J. L. W. V. Jensen. “Sur les fonctions convexes et les inégalités entre les valeurs moyennes”. In: *Acta mathematica* 30.1 (1906), pp. 175–193.

[30] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller. “Playing atari with deep reinforcement learning”. In: *arXiv preprint arXiv:1312.5602* (2013).

[31] R. S. Sutton, D. McAllester, S. Singh, and Y. Mansour. “Policy gradient methods for reinforcement learning with function approximation”. In: *Advances in neural information processing systems* 12 (1999).

[32] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, and M. Riedmiller. “Deterministic policy gradient algorithms”. In: *International conference on machine learning*. Pmlr. 2014, pp. 387–395.

[33] R. J. Williams. “Simple Statistical Gradient Algorithms for Connectionist Reinforcement Learning”. In: *Machine Learning* 8 (1992), pp. 229–256.

[34] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu. “Asynchronous Methods for Deep Reinforcement Learning”. In: *Proceedings of The 33rd International Conference on Machine Learning*. Vol. 48. Proceedings of Machine Learning Research. PMLR, 2016, pp. 1928–1937.

[35] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz. “Trust Region Policy Optimization”. In: *Proceedings of Machine Learning Research (ICML)*. Vol. 37. 2015, pp. 1889–1897.

[36] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. “Proximal Policy Optimization Algorithms”. In: *CoRR* abs/1707.06347 (2017). arXiv: 1707.06347.

[37] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra. “Continuous control with deep reinforcement learning.” In: *International Conference on Learning Representations (ICLR)*. 2016.

[38] S. Fujimoto, H. van Hoof, and D. Meger. “Addressing Function Approximation Error in Actor-Critic Methods”. In: *Proceedings of Machine Learning Research (ICML)*. Vol. 80. 2018, pp. 1587–1596.

[39] M. G. Bellemare, W. Dabney, and R. Munos. *A Distributional Perspective on Reinforcement Learning*. 2017. arXiv: 1707.06887 [cs.LG].

APPENDIX

4.6.1. PROOFS

Proof of Lemma 4.4.2. First, we prove the equation for the state value function $\hat{v}^\pi$:

$$\begin{aligned}\hat{v}^\pi(s,y) &= \mathbb{E}_\pi\big[y \vee \hat{G}_t \big| s_t{=}s\big] = \mathbb{E}_\pi\big[y \vee r_{t+1} \vee \gamma\hat{G}_{t+1}\big| s_t{=}s\big] \\ &= \gamma\mathbb{E}_\pi\big[y' \vee \hat{G}_{t+1}\big| s_t{=}s\big] = \gamma\mathbb{E}_\pi\big[y' \vee y' \vee \hat{G}_{t+1}\big| s_t{=}s\big] \\ &= \gamma\mathbb{E}_{\substack{a_t \\ s_{t+1}}}\big[y' \vee \hat{v}^\pi(s_{t+1},y')\big| s_t{=}s\big]\end{aligned}$$

Then, for the state-action value function $\hat{q}^\pi$:

$$\begin{aligned}\hat{q}^\pi(s,a,y) &= \mathbb{E}_\pi\big[y \vee \hat{G}_t \big|_{a_t=a}^{s_t=s}\big] = \mathbb{E}_\pi\big[y \vee r_{t+1} \vee \gamma\hat{G}_{t+1}\big|_{a_t=a}^{s_t=s}\big] \\ &= \gamma\mathbb{E}_\pi\big[y' \vee \hat{G}_{t+1}\big|_{a_t=a}^{s_t=s}\big] = \gamma\mathbb{E}_\pi\big[y' \vee y' \vee \hat{G}_{t+1}\big|_{a_t=a}^{s_t=s}\big] \\ &= \gamma\mathbb{E}_{\substack{s_{t+1} \\ a_{t+1}}}\big[y' \vee \hat{q}^\pi(s_{t+1}a_{t+1},y')\big|_{a_t=a}^{s_t=s}\big]\end{aligned}$$

□

Proof of Theorem 4.4.4. First, we demonstrate a simple property of the $\vee$ operator that we will use later. Let $a,x,y \in \mathbb{R}$. Then, using equation $x \vee y = 0.5(x+y+|x-y|)$, we obtain the following:

$$\begin{aligned}a \vee x - a \vee y &= 0.5\big(a + x + |x-a| - a - y - |y-a|\big) \\ &= 0.5(x - y + |x-a| - |y-a|) \\ &\le 0.5(x - y + |x-a-(y-a)|) \\ &= 0.5(x - y + |x-y|) \\ &\le |x-y|\end{aligned} \tag{4.9}$$

Now, we can prove that $\hat{\mathcal{T}}^\pi$ is a contraction. We begin with the state-action case. Let $q,z : \mathcal{S}\times\mathcal{A}\times\mathbb{R}$ be two-real valued functions. Then, we can expand $\|\hat{\mathcal{T}}^\pi q - \hat{\mathcal{T}}^\pi z\|_\infty$ as follows:

$$\begin{aligned}&\|\hat{\mathcal{T}}^\pi q - \hat{\mathcal{T}}^\pi z\|_\infty \\ &= \gamma \sup_{s\in\mathbb{S},a\in\mathbb{A},y\in\mathbb{R}} \Big|\mathbb{E}_{\substack{s_{t+1} \\ a_{t+1}}}\big[y' \vee q(s_{t+1},a_{t+1},y') - y' \vee z(s_{t+1},a_{t+1},y')\big|_{a_t=a}^{s_t=s}\big]\Big| \\ &\le \gamma \sup_{s\in\mathcal{S},a\in\mathcal{A},y\in\mathbb{R}} \mathbb{E}_{\substack{s_{t+1} \\ a_{t+1}}}\Big[\Big|y' \vee q(s_{t+1},a_{t+1},y') - y' \vee z(s_{t+1},a_{t+1},y')\Big|\Big|_{a_t=a}^{s_t=s}\Big]| \\ &\overset{(*)}{\le} \gamma \sup_{s_{t+1}\in\mathcal{S},a_{t+1}\in\mathcal{A},y'\in\mathbb{R}} \Big|y' \vee q(s_{t+1},a_{t+1},y') - y' \vee z(s_{t+1},a_{t+1},y')\Big| \\ &= \gamma \sup_{s\in\mathcal{S},a\in\mathcal{A},y\in\mathbb{R}} \Big|y \vee q(s,a,y) - y \vee z(s,a,y)\Big| \\ &\le \gamma \sup_{s\in\mathcal{S},a\in\mathcal{A},y\in\mathbb{R}} \Big|q(s,a,y) - z(s,a,y)\Big| = \|q-z\|_\infty\end{aligned}$$

The first inequality follows from the fact that $\left|\mathbb{E}_x[x]\right| \leq \mathbb{E}_x\left[|x|\right]$ and the last inequality follows from Eq. (4.9). In $(*)$, we use the following property of the expectation: $\sup_y \{\mathbb{E}[x|y]\} \leq \sup_x\{x\}$. Now, we demonstrate the contraction property for the state value function: Let $v, u : \mathcal{S} \times \mathbb{R}$ be two-real valued functions. Then, we expand $\|\hat{\mathcal{T}}^\pi v - \hat{\mathcal{T}}^\pi u\|_\infty$ as follows:

$$
\begin{aligned}
\|\hat{\mathcal{T}}^\pi v - \hat{\mathcal{T}}^\pi u\|_\infty &= \gamma \sup_{s\in\mathbb{S}, y\in\mathbb{R}} \left|\mathbb{E}_{\substack{a_t \\ s_{t+1}}}\left[y' \vee v(s_{t+1}, y') - y' \vee u(s_{t+1}, y')\middle|^{s_t=s}\right]\right| \\
&\leq \gamma \sup_{s\in\mathcal{S}, y\in\mathbb{R}} \mathbb{E}_{\substack{a_t \\ s_{t+1}}}\left[\left|y' \vee v(s_{t+1}, y') - y' \vee u(s_{t+1}, y')\right|^{s_t=s}\right] \\
&\leq \gamma \sup_{s_{t+1}\in\mathcal{S}, y'\in\mathbb{R}} \left|y' \vee v(s_{t+1}, y') - y' \vee u(s_{t+1}, y')\right| \\
&= \gamma \sup_{s\in\mathcal{S}, y\in\mathbb{R}} \left|y \vee v(s, y) - y \vee u(s, y)\right| \\
&\leq \gamma \sup_{s\in\mathcal{S}, y\in\mathbb{R}} \left|v(s, y) - u(s, y)\right| = \|v - u\|_\infty
\end{aligned}
$$

Therefore, the max-reward Bellman operator is a contraction. Hence, by the Banach fixed-point theorem, it has unique fixed-point(s). From Lemma 4.4.2, we conclude that this is the max-reward value function(s). □

Proof of Lemma 4.4.8.

$$
\begin{aligned}
&\|\hat{\mathcal{T}}^* q - \hat{\mathcal{T}}^* z\|_\infty \\
&= \gamma \sup_{s\in\mathbb{S}, a\in\mathbb{A}, y\in\mathbb{R}} \left|\mathbb{E}_{s_{t+1}}\left[y' \vee \max_{a'} q(s_{t+1}, a', y') - y' \vee \max_{a'} z(s_{t+1}, a', y')\middle|^{s_t=s}_{a_t=a}\right]\right| \\
&\leq \gamma \sup_{s\in\mathcal{S}, a\in\mathcal{A}, y\in\mathbb{R}} \mathbb{E}_{s_{t+1}}\left[\left|y' \vee \max_{a'} q(s_{t+1}, a', y') - y' \vee \max_{a'} z(s_{t+1}, a', y')\right|^{s_t=s}_{a_t=a}\right] \\
&\leq \gamma \sup_{s\in\mathcal{S}, a\in\mathcal{A}, y\in\mathbb{R}} \mathbb{E}_{s_{t+1}}\left[\max_{a'} \left|y' \vee q(s_{t+1}, a', y') - y' \vee z(s_{t+1}, a', y')\right|^{s_t=s}_{a_t=a}\right] \\
&\leq \gamma \sup_{s_{t+1}\in\mathcal{S}, a_{t+1}\in\mathcal{A}, y'\in\mathbb{R}} \left|y' \vee q(s_{t+1}, a_{t+1}, y') - y' \vee z(s_{t+1}, a_{t+1}, y')\right| \\
&= \gamma \sup_{s\in\mathcal{S}, a\in\mathcal{A}, y\in\mathbb{R}} \left|y \vee q(s, a, y) - y \vee z(s, a, y)\right| \\
&\leq \gamma \sup_{s\in\mathcal{S}, a\in\mathcal{A}, y\in\mathbb{R}} \left|q(s, a, y) - z(s, a, y)\right| = \|q - z\|_\infty
\end{aligned}
\tag{4.10}
$$

□

Proof of Theorem 4.4.9. First of all, we notice that the max-reward

Bellman equation implies another recursive equation for $\hat{v}^\pi$ and $\hat{q}^\pi$:

$$\hat{q}^\pi(s,a,y) = \gamma \mathbb{E}_{s_{t+1}}\big[y' \vee \hat{v}^\pi(s_{t+1},y')\big|_{s_t=s}\big] = \gamma \mathbb{E}_{\substack{a_t \\ s_{t+1}}}\big[\hat{v}^\pi(s_{t+1},y')\big|_{a_t=a}^{s_t=s}\big] \tag{4.11}$$

As discussed in the main paper, we use the version with extra $\vee y'$ to enforce boundary conditions. However, we can still use the equation above as it is a property of the max-reward value function.

Before proving the theorem, we introduce simplified notation to improve readability – for all functions, we use subscripts to denote the input variables. For example, $\hat{v}_t := \hat{v}^{\hat{\pi}}(s_t, y_t)$. The proof follows the one for the standard policy gradient theorem. We begin by obtaining a recurrent equation for $\nabla_\theta \hat{v}_0$:

$$\begin{aligned}
\nabla_\theta \hat{v}_0 &= \nabla_\theta\Big(\int_{a_0} \hat{\pi}_0 \hat{q}_0 da_0\Big) = \underbrace{\int_{a_0} (\nabla_\theta \hat{\pi}_0)\hat{q}_0 da_0}_{\phi_0} + \int_{a_0} \hat{\pi}_0 (\nabla_\theta q)_0 da_0 \\
&\overset{\text{Eq.(4.11)}}{=} \phi_0 + \gamma \int_{a_0} \hat{\pi}_0 \Big(\nabla_\theta \int_{s_1,y_1} \hat{p}(s_1,y_1|s_0,y_0,a_0)\hat{v}_1 ds_1 dy_1\Big) da_0 \\
&= \phi_0 + \gamma \int_{s_1,y_1}\int_{a_0} \hat{\pi}_0 \hat{p}(s_1,y_1|s_0,y_0,a_0)(\nabla_\theta \hat{v}_1) ds_1 dy_1 da_0 \\
&= \phi_0 + \gamma \int_{s_1,y_1} \hat{p}_1^{\hat{\pi}}(s_0,y_0,s_1,y_1)(\nabla_\theta \hat{v}_1) ds_1 dy_1\,,
\end{aligned}$$

where we introduced the shorthand $\phi_t = \phi(s_t,y_t) = \int_a \nabla_\theta \hat{\pi}(a|s_t,y_t)\, q(s_t,a,y_t)\, da$. Expanding this recurrence further, we obtain

$$\begin{aligned}
\nabla_\theta \hat{v}_0 &= \sum_{t=0}^{\infty} \int_{s_t,y_t} \gamma^t \hat{p}_t^{\hat{\pi}}(s_0,y_0,s_t,y_t)\phi(s_t,y_t) ds_t dy_t \\
&= \int_{s,y} \Big(\sum_{t=0}^{\infty} \gamma^t \hat{p}_t^{\hat{\pi}}(s_0,y_0,s,y)\Big)\phi(s,y) ds dy \\
&\propto \int_{s,y} \hat{d}^{\hat{\pi}}(s,y|s_0,y_0)\phi(s,y) ds dy \\
&= \int_{s,y} \hat{d}^{\hat{\pi}}(s,y|s_0,y_0)\Big(\int_a (\nabla_\theta \hat{\pi}(a|s,y))\hat{q}(s,a,y) da\Big) ds dy
\end{aligned}$$

Above, $\hat{d}^{\hat{\pi}}(s,y|s_0,y_0)$ is the discounted stationary distribution of s,y for policy π given s_0,y_0. Finally, we substitute this formula for $\nabla_\theta \hat{v}_0$ into the

definition of $\hat{J}(\theta)$ and conclude the proof:

$$
\begin{aligned}
&\nabla_\theta \hat{J}(\theta) \\
&= \int_{s_0,y_0} \hat{p}_0(s_0,y_0) \int_{s,y} \hat{d}^{\hat{\pi}}(s,y|s_0,y_0) \Big(\int_a \big(\nabla_\theta \hat{\pi}(a|s,y)\big) \hat{q}(s,a,y) da \Big) ds dy ds_0 dy_0 \\
&= \int_{s,y} \hat{d}^{\hat{\pi}}(s,y) \Big(\int_a \big(\nabla_\theta \hat{\pi}(a|s,y)\big) \hat{q}(s,a,y) da \Big) ds dy \\
&= \mathbb{E}_{\substack{s,y\sim\hat{d}^{\hat{\pi}} \\ a\sim\hat{\pi}(\cdot|s,y)}} \left[\hat{q}^{\hat{\pi}}(s,a,y) \nabla_\theta \ln \hat{\pi}(a|s,y) \right]
\end{aligned}
\tag{4.12}
$$

□

4

4.6.2. EXPERIMENTAL DETAILS

For all experiments, we used our implementation of TD3 and PPO that we verified on several MuJoco domains. The implementation of the max-reward algorithms is similar to their cumulative versions except for the following differences:

1. The input layer of all neural networks has an extra dimension to work with the extended states (s, y).
2. The output layer of the value networks uses *Tanh* activation and is rescaled to $u \in [0, \bar{R}]$. Then, it is transformed with $ReLU(u-y)+y$ to enforce $\hat{v}^\pi(s,y) \geq y$.

Hyperparameters of all runs are reported in Tables 4.1-4.2.

4.6.3. ALGORITHMS

Parameter	PPO	PPOMax	TD3	TD3Max
Parallel environments	16	16	16	16
Discount factor γ	0.99	0.999	0.99	0.995
Learning rate	3e-4	3e-4	3e-4	3e-4
Lr. annealing	No	No	No	No
Entropy weight	5e-2	5e-2		
Value loss weight	0.5	0.5		
Clip coef.	0.2	0.2		
GAE λ	0.95	1		
Policy update freq.			2	2
Target soft update τ			0.005	0.005
Expl. noise type			pink	pink
Expl. noise std			0.7	0.7
Expl. noise clip			0.5	0.5
Target noise scale			0.2	0.2
Initial expl. steps			25000	25000
Tr. epochs per rollout	10	10		
Rollout length	1024	2048		
Minibatch size	32	32	256	256

Table 4.1: Hyperparameters for the experiments with Maze environment.

Parameter	TD3	TD3Max
Parallel environments	16	16
Discount factor γ	0.99	0.995
Learning rate	3e-4	3e-4
Lr. annealing	No	No
Policy update freq.	2	2
Target soft update τ	0.005	0.005
Expl. noise type	pink	pink
Expl. noise std	0.1	0.1
Expl. noise clip	0.5	0.5
Target noise scale	0.2	0.2
Initial expl. steps	25000	25000
Minibatch size	256	256

Table 4.2: Hyperparameters for the experiments with Fetch environment.

Algorithm 2 Max-reward TD3

Initialize critic networks $\hat{q}_{\phi_1}, \hat{q}_{\phi_2}$ and actor network μ_θ
Initialize target networks $\phi'_1 \leftarrow \phi_1$, $\phi'_2 \leftarrow \phi_2$, $\theta' \leftarrow \theta$
Initialize replay buffer $\mathcal{D}$
for *episode* $= 1, 2, \ldots$ **do**
    Initialize $s_0, y_0 \sim \hat{p}_0$
    for $t = 0, 1, \ldots, T-1$ **do**
        Sample exploration noise ϵ_t
        Execute $a_t = \mu(s_t, y_t) + \epsilon_t$ and get s_{t+1}, r_{t+1}
        Update $y_{t+1} = (y_t \vee r_{t+1})/\gamma$
        Save $(s_t, y_t, a_t, s_{t+1}, r_{t+1}, y_{t+1})$ into $\mathcal{D}$
        if initial exploration is over **then**
            Sample a mini-batch of size N from $\mathcal{D}$
            Sample target actions noise η
            $\tilde{a} \leftarrow \mu_{\theta'}(s', y') + \eta$
            $z \leftarrow y' \vee \gamma \min_{i=1,2} \hat{q}_{\phi'_i}(s', \tilde{a}, y')$
            Critic loss $L_c = \frac{1}{N} \sum_{i=1}^{2} (z - \hat{q}_{\phi_i}(s, a, y))^2$
            Perform gradient update step on L_c
            if time to update policy **then**
                $L_a \leftarrow \mathbb{E}\big[\hat{q}_{\phi_1}(s, \mu_\theta(s, y), y)\big]$
                Perform gradient update step on L_a
                $\phi'_i \leftarrow \tau\phi_i + (1-\tau)\phi'_i$, $i = 1, 2$
                $\theta' \leftarrow \tau\theta + (1-\tau)\theta$
            end if
        end if
    end for
end for

Algorithm 3 Max-reward PPO

Initialize actor $\hat{\pi}_\theta$ and critic $\hat{v}_\phi$
for *iteration* $= 1, 2, \ldots$ **do**
  Initialize trajectories buffer $\mathcal{D}$
  for *actor* $= 1, 2, \ldots, N$ **do**
    Initialize $s_0, y_0 \sim \hat{p}_0$
    for $t = 0, 1, \ldots, T-1$ **do**
      Execute $a_t \sim \pi_\theta(\cdot|s_t, y_t)$ and get s_{t+1}, r_{t+1}
      Update $y_{t+1} = (y_t \vee r_{t+1})/\gamma$
      Save $(s_t, y_t, a_t, s_{t+1}, r_{t+1}, y_{t+1})$ into $\mathcal{D}$
    end for
    $\hat{G}_t^n \leftarrow \gamma^n \hat{v}_\phi(s_{t+n}, y_{t+n}),\ n = 1, \ldots, T-t$
    $\hat{G}_t(\lambda) = (1-\lambda)\sum_{n=1}^{T-t} \lambda^{n-1} \hat{G}_t^n$
    Compute advantages $\hat{A}_t = \hat{G}_t(\lambda) - \hat{v}_\phi(s_t, y_t)$
  end for
  for k=1, 2 . . . K **do**
    Critic loss: $L_c \leftarrow \frac{1}{T}\sum_t (\hat{G}_t^{T-t} - \hat{v}_\phi(s, y))^2$
    Actor loss: $L_a \leftarrow L_{PPO}(\pi_\theta, \{\hat{A}_t\}_{t=1}^T)$
    Perform gradient update step on $L_a + L_c$
  end for
end for

5

CONCLUSION

Throughout this thesis, we explore various challenges in optimization under uncertainty and propose novel approaches to address them. While our contributions span diverse domains, they are unified by a focus on efficient problem formulations. The research presented in Chapters 2–4 illustrates how these formulations can be developed for power systems, decision-focused learning, and reinforcement learning, demonstrating their theoretical and experimental advantages.

In this concluding chapter, we review the key results from each chapter and discuss their implications for future research. We then synthesize these findings to draw broader conclusions about the pivotal role of problem formulation in optimization under uncertainty and consider how the research presented in this thesis can contribute to advancing the field.

5.1. SURROGATE MODELS AND PARTIAL OBSERVABILITY

In Chapter 2, we address the challenge of optimizing electric vehicle (EV) charging in DC microgrids under partial observability, where EV locations within the grid are known only at the cable level. This setup is particularly relevant for preserving privacy, when EV owners wish to maintain location confidentiality. Additionally, it can also be relevant when communication between microgrid nodes is limited due to technical constraints.

Beyond the challenges posed by partial observability, the equations governing power flows in DC microgrids are inherently non-convex, which further increases computational complexity. A direct approach to this problem would involve explicitly modeling all possible assignments of EVs to charging locations, making the problem intractable even for moderately sized grids. The key challenge, therefore, is to handle both the aleatoric uncertainty in EV locations and the computational complexity of the optimization problem.

The core contributions of our work are two novel surrogate problem formulations that redefine the microgrid model in an uncertainty-indifferent manner. Both reformulations address the issue of partial observability, albeit in different ways: one represents each cable as a single large node, while the other models each cable as multiple nodes connected in parallel. Both reformulations are invariant to permutations of EVs within cables, making them uncertainty-agnostic. By combining these surrogates with an existing convex relaxation technique and introducing a planning-execution separation, we obtain solutions that address partial observability while ensuring that the grid constraints are never violated. This approach allows us to manage aleatoric uncertainty in EV locations without explicitly modeling it, relying instead on specifically designed reformulations.

Experiments conducted on various simulated microgrids demonstrate that our surrogate models significantly outperform the baseline. Specifically, the parallel nodes model produces high-quality solutions but incurs

a higher computational cost, while the single-node approach is computationally lightweight and still yields good solutions. Thus, our contributions highlight how designing reformulations can involve a trade-off between solution quality and computational requirements.

The surrogate models proposed in Chapter 2 have direct applications in privacy-respecting EV charging scheduling and can inspire the development of surrogate-based approaches for other power system challenges. For instance, problems like multi-timestep optimal power flow or unit commitment often become intractable when explicitly modeling uncertainty. These problems could benefit from surrogate models that resolve uncertainty while ensuring the satisfaction of system constraints.

In a broader sense, Chapter 2 demonstrates how problem reformulations can provide practical solutions for managing uncertainty in computationally challenging problems, where neither explicit uncertainty modeling nor learning-based approaches are feasible. This work highlights the potential of leveraging domain expertise to address problems characterized by high degrees of uncertainty, such as resource allocation or task scheduling. It shows that targeted reformulations can reduce the reliance on complex modeling while still delivering reliable and effective results.

5.2. ADDRESSING ZERO GRADIENTS

The core idea of decision-focused learning is to train machine learning models based on decision quality rather than prediction accuracy when estimating missing parameters in optimization problems. In Chapter 3, we address a key technical challenge in applying this framework to convex optimization: the zero-gradient issue that arises when differentiating through such problems. Previously identified only in the linear case, this issue severely limits the practical application of decision-focused learning, as it causes convergence to suboptimal solutions.

We tackle this challenge by proposing a problem reformulation that remains consistent with the original formulation while avoiding the zero-gradient issue. This reformulation involves two key steps: quadratic approximation and local smoothing. Quadratic approximation provides the problem with a universal and tractable structure, offering computational simplicity and facilitating theoretical analysis. Local smoothing modifies the gradient computation by smoothing regions where the gradient would otherwise vanish, enabling proper gradient propagation even in areas where traditional methods fail.

Our extensive experiments and theoretical analysis underscore the practical significance of the zero-gradient problem and demonstrate the effectiveness of the proposed solutions. These experiments highlight the limitations of conventional decision-focused learning methods, where the zero-gradient issue severely restricts performance. In contrast, our refor-

mulation successfully addresses these limitations, offering a reliable gradient propagation framework that improves model training and decision quality. By resolving these challenges, we extend the practical scope of decision-focused learning to a broader range of convex optimization problems.

This work suggests that analyzing the loss function landscape and identifying regions with poor behavior, such as vanishing gradients, is a promising approach to improving decision-focused learning. Furthermore, our use of surrogate reformulation demonstrates the potential to address more complex applications, such as non-convex optimization, using similar techniques. By smoothing or approximating problem structures, decision-focused learning could potentially be extended beyond the convex case, opening new opportunities for tackling challenging optimization problems.

Looking ahead, an exciting direction for future research lies in applying decision-focused learning to sequential decision-making tasks. In many real-world applications, such as energy management or robotics, decisions are interconnected and significantly influence subsequent choices. These dependencies often result in computationally intractable problems, which we believe could benefit from novel reformulations that simplify and streamline the solution process. Extending decision-focused learning to this domain represents a promising avenue for research, offering substantial value for practitioners.

5.3. MAX-REWARD REINFORCEMENT LEARNING

In Chapter 4, we address the challenge of aligning the optimization objective in reinforcement learning with the desired behavior, particularly in goal-reaching tasks. We begin by examining the limitations of using cumulative return as the standard objective for these tasks. Although policies optimized for cumulative return are mathematically guaranteed to be optimal goal-reaching policies, learning such policies in practice often presents significant challenges. In particular, agents may converge to suboptimal policies that exploit the reward function rather than exhibiting the intended goal-reaching behavior.

To address these challenges, we introduce max-reward reinforcement learning as a more intuitive and mathematically advantageous formulation for goal-reaching tasks. Unlike cumulative return, max-reward RL focuses solely on the singular goal of reaching the target state in the fewest possible steps, aligning more closely with how humans naturally approach such tasks. Furthermore, max-reward RL offers computational benefits, enabling agents to learn goal-reaching behaviors more efficiently. As demonstrated through extensive experiments, this formulation results in more stable learning of goal-reaching policies.

While the max-reward formulation proves highly effective for certain

problems, it is not universally applicable and may be unsuitable for many tasks. Nevertheless, it presents exciting opportunities as a foundational building block for improved reinforcement learning algorithms. For example, future research could explore combining max-reward with cumulative return to leverage the strengths of both approaches. Another promising application of max-reward lies in safe RL, where safety constraints are often defined in terms of distance to obstacles. In such scenarios, safety can be expressed through max-reward—or in this context, max-cost—value functions.

In a broader sense, Chapter 4 underscores the importance of careful problem formulation in reinforcement learning, demonstrating that even small modifications to the optimization objective can have a substantial impact on an agent's performance and learning process. The max-reward RL framework offers a practical approach for goal-reaching tasks and motivates researchers to explore the applicability of other non-cumulative objective functions across diverse tasks.

5.4. FUTURE DIRECTIONS

The contributions of each chapter lay a foundation for further exploration into effective problem formulations in optimization under uncertainty. In this final section of the thesis, we summarize the key lessons learned and outline directions for future research.

The surrogate models for EV charging presented in Chapter 2 suggest that similar approaches could be extended to other domains, such as decentralized systems with limited communication or privacy constraints. Specifically, these methods could facilitate the development of lightweight models, informed by human domain expertise, that effectively handle aleatoric uncertainty in complex environments. In Chapter 3, the proposed approach to decision-focused learning demonstrates the potential for extending these ideas to more complex scenarios where exact differentiation of constrained optimization problems is not feasible. For example, this approach could benefit decision-focused learning for non-convex and/or sequential problems that arise in various applications, such as robotics or job scheduling. Developing tractable formulations that enable efficient learning in such settings could significantly broaden the applicability of decision-focused approaches. Similarly, the max-reward reinforcement learning framework introduced in Chapter 4 provides a novel perspective on task-specific formulations in RL. This framework shows particular promise for tasks where aligning learning objectives with desired outcomes is challenging. Further exploration of max-reward RL, including its applications in safety and exploration, as well as its integration with cumulative returns, could offer valuable insights and drive meaningful advancements in the field.

Collectively, the contributions of this thesis emphasize a broader prin-

ciple: effective optimization under uncertainty often depends on how the problem is framed. Across power systems, decision-focused learning, and reinforcement learning, this work demonstrates that carefully crafted problem formulations can address diverse challenges in optimization under uncertainty, leading to more tractable and effective solutions. These findings suggest that placing greater emphasis on problem formulation could drive significant progress in the field. The key lessons distilled from this thesis are as follows:

1. **Formulations can address uncertainty**: Instead of explicitly modeling uncertainty, well-designed formulations can efficiently manage it, as demonstrated by the surrogate models in Chapter 2.
2. **Balancing competing objectives**: Problem reformulations can balance competing metrics, such as scalability and solution quality, as illustrated by the surrogate models in Chapter 2.
3. **Solution-aware formulations**: Effective formulations should consider the solution method, as seen in Chapter 3, where the quadratic approximation and local smoothing enable efficient gradient-based training.
4. **Task-aligned objectives**: Aligning formulations with the specific structure of a task, rather than relying on generic approaches, can improve performance, as demonstrated by decision-focused learning and max-reward RL in Chapters 3 and 4.

Looking ahead, structured research into effective problem formulations offers intriguing possibilities. While this thesis focused on deriving reformulations through human knowledge and intuition, future work could investigate automated techniques for identifying effective formulations, drawing inspiration from machine learning methods such as automated hyperparameter tuning or neural architecture search. For example, learning-based approaches could identify patterns common to effective formulations, facilitating their systematic discovery. Formal methods might also be employed to verify the equivalence of alternative formulations or rigorously analyze their differences. Additionally, recent advances in language models could enable tools that translate natural language problem descriptions into precise mathematical representations. Such tools could help craft formulations that more accurately reflect the nature of the task or seamlessly integrate domain knowledge and heuristics into problem definitions and solutions.

In conclusion, this thesis demonstrates that problem formulation is not merely a preliminary step in optimization under uncertainty but a critical tool for managing complexity and improving solution quality. As real-world challenges grow increasingly intricate, the insights and approaches developed here advocate for a continued search for innovative formulations that can drive more effective and robust solutions.

CURRICULUM VITAE

Grigorii Veviurko

19-12-1995	Born in Ufa, Russia

EDUCATION

2020–2024	PhD in Computer Sceince Delft University of Technology, Delft, Netherlands
2017–2019	MSc in Computational Neuroscience Technical University Berlin, Berlin, Germany
2013–2017	BSc in Mathematics Higher School of Economics, Moscow, Russia
2002–2013	General education Independent school "Moomintroll", Moscow, Russia

WORK EXPERIENCE

2025 onward	ML & Optimization Software Engineer, Apple, Munich, Germany
2024	Research intern Nokia Bell Labs, Stuttgart, Germany
2017–2020	Research assistant Max Delbrück Center for Molecular Medicine, Berlin, Germany

LIST OF CONTRIBUTIONS

RELATED PUBLICATIONS

3. G. Veviurko, W. Boehmer, and M. de Weerdt. “To the Max: Reinventing Reward in Reinforcement Learning”. In: *Forty-first International Conference on Machine Learning*. 2024
2. G. Veviurko, W. Böhmer, and M. de Weerdt. “You Shall not Pass: the Zero-Gradient Problem in Predict and Optimize for Convex Optimization”. In: *arXiv preprint arXiv:2307.16304* (2023)
1. G. Veviurko, W. Böhmer, L. Mackay, and M. de Weerdt. “Surrogate DC Microgrid Models for Optimization of Charging Electric Vehicles under Partial Observability”. In: *Energies* 15.4 (2022), p. 1389

UNRELATED PUBLICATIONS

1. A. J. Barker, G. Veviurko, N. C. Bennett, D. W. Hart, L. Mograby, and G. R. Lewin. “Cultural transmission of vocal dialect in the naked mole-rat”. In: *Science* 371.6528 (2021), pp. 503–507

OTHER CONTRIBUTIONS

2. Prototype of a Web Navigation Agent, Nokia Bell Labs
1. Prototype of an adaptive DC microgrid controller, DC Opportunities