Training Deep-Parametric Policies Using Lagrangian Duality

Andrew Rosemberg
NSF AI Institute for Advances in Optimization
Georgia Institute of Technology
Atlanta, GA
arosemberg@gatech.edu

Alexandre Street
Laboratory of Applied Mathematical Programming and Statistics
Pontifical Catholic University of Rio de Janeiro
RJ, Brazil
alexandre.street@puc-rio.br

Davi M. Valladão
Laboratory of Applied Mathematical Programming and Statistics
Pontifical Catholic University of Rio de Janeiro
RJ, Brazil
davi.valladao@puc-rio.br

Pascal Van Hentenryck
NSF AI Institute for Advances in Optimization
Georgia Institute of Technology
Atlanta, GA
vanhentenryck@gatech.edu

Abstract

Sequential Decision Making under Uncertainty (SDMU) appears across energy, finance, and supply chains. Stochastic Dual Dynamic Programming (SDDP) is a powerful solution approach to these problems but assumes convexity and stage-wise independence; Two-Stage Linear Decision Rules (TS-LDRs) relax independence and yield fast policies but are limited in non-convex environments. This paper introduces *Two-Stage General Decision Rules* (TS-GDR) and an instantiation, *Two-Stage Deep Decision Rules* (TS-DDR), which train nonlinear, time-invariant policies by combining deterministic optimization in the forward pass with duality-based closed-form gradients in the backward pass. On the Long-Term Hydrothermal Dispatch (LTHD) problem for the Bolivian grid, TS-DDR improves solution quality and reduces training/inference time by orders of magnitude compared to SDDP, Reinforcement Learning (RL) and TS-LDR across linear (DCLL), conic (SOC), and non-convex (AC) implementations; it also outperforms a model-predictive control baseline on the stochastic Goddard rocket control problem.

1 Introduction

Sequential Decision Making under Uncertainty (SDMU) models real systems where decisions unfold over time as uncertainty is revealed and prior actions constrain feasibility [1, 2]. Many finite-horizon SDMU problems are naturally posed as Multistage Stochastic Programs (MSPs), which capture both sequential coupling and application constraints but suffer from the curse of

39th Conference on Neural Information Processing Systems (NeurIPS 2025) Workshop: MLxOR: Mathematical Foundations and Operational Integration of Machine Learning for Uncertainty-Aware Decision-Making.

dimensionality [3]. When problems are convex and uncertainty is stage-wise independent, Stochastic Dual Dynamic Programming (SDDP) is effective and widely used in energy systems [4, 5]. However, both assumptions often fail in practice, where non-convex models and non-Gaussian time-dependent uncertainty preclude direct use of the technique. A concise MSP primer and the connection to Bellman/value recursion are summarized in Appendix A and Appendix A.1.

Two-Stage Linear Decision Rules (TS-LDRs) remove the independence assumption and yield implementable, fast policies [6, 7], but linearity (and the convex relaxations they rely on) is ill-suited to non-convex environments, leading to oversimplifications and suboptimality [8]. Reinforcement Learning (RL) excels in Markov decision processes [9–11] but struggles with sample efficiency and feasibility under hard, high-dimensional constraints; constrained/trust-region variants only partially alleviate these issues [12, 13]. Further details on policy iteration and TS-LDR appear in Appendix A.2.

This paper bridges RL and stochastic programming by introducing *Two-Stage General Decision Rules* (TS-GDR) and a deep instantiation, *TS-DDR*. TS-DDR trains time-invariant policies by (i) solving a multi-step *deterministic* optimization in the forward pass to enforce feasibility and (ii) using dual multipliers as closed-form gradients in the backward pass—avoiding expensive implicit differentiation [14]. A compact decision-making discussion (rolling vs. time-invariant) is provided in Appendix C.

TS-DDR is demonstrated on the Long-Term Hydrothermal Dispatch (LTHD) problem [15], including non-convex AC power-flow constraints, showing improved solution quality and orders-of-magnitude speedups over SDDP and TS-LDR, while model-free RL fails to find feasible solutions under high-penalty constraints.

2 Technical Approach

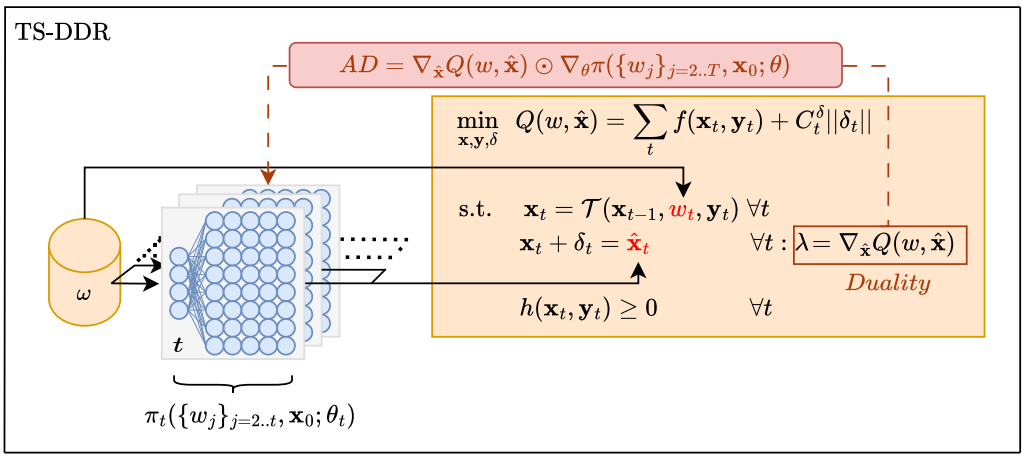

Figure 1: Training TS-DDR using Lagrangian Duality

2.1 Two-Stage General Decision Rules

The first contribution of this paper is to generalize TS-LDR to Two-Stage General Decision Rules (TS-GDR), which supports arbitrary decision rules. In TS-GDR, the second stage problem $\mathcal{Q}(w; \theta)$ becomes

$$
\begin{aligned}
\min_{\mathbf{x},\mathbf{y},\delta} \quad & \sum_t f(\mathbf{x}_t, \mathbf{y}_t) + C_t^\delta ||\delta_t|| \\
\text{s.t.} \quad & \mathbf{x}_t = \mathcal{T}(\mathbf{x}_{t-1}, w_t, \mathbf{y}_t) && \forall t \\
& \mathbf{x}_t + \delta_t = \hat{\mathbf{x}}_t \quad : \lambda_t && \forall t \\
& h(\mathbf{x}_t, \mathbf{y}_t) \geq 0 && \forall t
\end{aligned}
$$

For specific parameter values, the target state is specified by the policy, i.e., $\hat{\mathbf{x}}_t = \pi_t(\{w_j\}_{j=2..t}, \mathbf{x}_0; \theta_t)$, thus *decoupling the next state prediction from the optimization of decision variables* $\mathbf{x}, \mathbf{y}, \delta$. *Importantly, the second stage is a deterministic multi-period optimization.*

The second contribution of this paper is to define Two-Stage Deep Decision Rules (TS-DDR) where the policy π_t^d is specified by a deep neural network.

2.2 Training of TS-DDR

This section presents the third contribution of the paper: the training procedure for TS-DDR, as depicted in Figure 1. While a TS-LDR policy can be trained by solving a large-scale two-stage mathematical programming problem (e.g., using Benders decomposition), the training of the TS-DDR uses a combination of machine learning and mathematical programming techniques. More specifically, *the training of the policy parameters uses deterministic optimization technology in the forward phase (depicted if Fig. 1 by the direction of the solid arrows) to satisfy the problem constraints and ensure feasibility, and stochastic gradient descent in the backward phase (depicted if Fig. 1 by the direction of the dashed arrows).* The link between the two phases exploit *duality theory*.

At each iteration, the training procedure predicts the target state $\hat{\mathbf{x}}_t$ and solves the second stage optimization. To train the policy π parameterized by θ, the backward phase differentiates through $\mathcal{Q}(w; \theta)$ and updates the policy parameters accordingly using Automatic Differentiation (AD). Since the target $\hat{\mathbf{x}}_t = \pi_t(\{w_j\}_{j=2..t}, \mathbf{x}_0; \theta_t)$ is a right-hand-side (rhs) parameter of the inner problem Q, by duality theory, the subgradient of a subproblem objective with respect to $\hat{\mathbf{x}}_t$ is given by the dual variable of its associated constraint. If the problem is non-convex, the dual is the subgradient of the local optimum. As a result,

$$\nabla_\theta \mathbf{E}[\mathcal{Q}(w;\theta)] \approx \frac{1}{S}\sum_{s=1}^{S} \nabla_\theta \mathcal{Q}(w^s;\theta) = \frac{1}{S}\sum_{s=1}^{S} \overbrace{\nabla_{\hat{\mathbf{x}}} Q(w^s, \hat{\mathbf{x}})}^{\lambda^s} \odot \nabla_\theta \pi(\{w_j^s\}_{j=2..T}, \mathbf{x}_0; \theta)$$

where S is the number of samples. A full training pseudocode and time-invariant policy specification are given in Appendix B, Algorithm 1.

By representing the policy in terms of only the realized uncertainties, the proposed method can avoid the implicit function differentiation of the KKT conditions, as done in [14], to back-propagate rewards through the state transition towards the parametric policy. This is a major advantage for applications with strict training times.

3 Results

Main Experiment. This study examines the Long-Term Hydrothermal Dispatch (LTHD) problem [15] with real data from Bolivia (28 buses, 26 loads, 34 generators—11 hydro, 31 branches, horizon 96; 165 scenarios per stage). The LTHD problem refers to the optimization of sequential decision strategies (policies) to manage (Hydro) energy storage systems and minimize energy dispatch costs. The LTHD is a complicated MSP problem as it needs to take into account the complex power flow equations that represent the physics of electricity transmission. The non-convex nature of AC power flow inhibits the use of classical optimization methods such as SDDP. Moreover, other control and ML methods (such as RL) have difficulties scaling with decision and scenario dimensions. It is an important case study to evaluate the benefits of any novel method. The second-stage policy implementation is described in Appendix D.1. Implementation models include a linear DC with line losses (DCLL), the conic SOC relaxation, and the non-convex AC power flow. Formulations are detailed in Appendices G, G.1, and H; the overall Bolivian case description is in Appendix D. The study compares TS-DDR (time-invariant), TS-LDR, and SDDP; model-free RL baselines are also examined. Hyperparameters, solvers, and power-flow formulations appear in the appendix.

Quality and speed. On convex implementations where SDDP is optimal (stage-independent uncertainty), TS-DDR is within ∼0.6% (DCLL) (Table 1) and ∼0.5% (SOC) (Table 2) of optimum while achieving *4 orders of magnitude faster* execution than SDDP; training is also substantially faster. In the non-convex AC implementation (Table 3), TS-DDR attains the best implementation cost, improving over SDDP by ∼0.3% while remaining orders of magnitude faster at both training and inference. TS-DDR also consistently outperforms TS-LDR in cost. Complete tables (means, standard deviations, wall times) are in the appendix.

Table 1: Comparison of SDDP and ML Decision Rule for Bolivia with DCLL Implementations.

Model	Plan	Imp Cost (USD)	GAP (%)	Training (Min)	Execution (Min)
SDDP	DCLL	$295,879(\pm 5,667)$	-	-	10.421
TS-DDR	DCLL	$297,757(\pm 5,169)$	$0.63(\pm 2.71)$	3	0.002
TS-LDR	DCLL	$300,806(\pm 8,054)$	$1.66(\pm 0.72)$	15	0.002

Table 2: Comparison of SDDP and ML Decision Rule for Bolivia with SOC Implementations.

Model	Plan	Imp Cost (USD)	GAP (%)	Training (Min)	Execution (Min)
SDDP	SOC	$300,219(\pm 5,176)$	-	-	482.699
TS-DDR	SOC	$301,694(\pm 4,856)$	$0.49(\pm 2.38)$	40	0.025
TS-LDR	SOC	$313,205(\pm 5,295)$	$4.32(\pm 2.42)$	150	0.025

Table 3: Comparison of SDDP and ML Decision Rule for Bolivia with AC Implementation.

Model	Plan	Imp Cost (USD)	GAP (%)	Training (Min)	Execution (Min)
TS-DDR	AC	$301,851(\pm 4,876)$	-	60	0.067
SDDP	SOC	$302,816(\pm 5,431)$	$0.32(\pm 2.34)$	-	478.699
TS-LDR	AC	$319,326(\pm 4,715)$	$5.79(\pm 1.30)$	226	0.067
SDDP	DCLL	$323,895(\pm 3,944)$	$7.30(\pm 2.27)$	-	320.113

Behavioral insights. Under AC, TS-DDR stores less water while avoiding expensive thermal dispatch, whereas SDDP-based policies tend to require higher stored energy, consistent with prior observations on time-inconsistency [8]. See Appendix J, Figures 3 and 4 (AC), with SOC/DCLL counterparts in the same appendix.

RL baselines. Model-free methods (e.g., PPO, DDPG, TD3, SAC) improved performance compared to random policies but stagnated and did not find fully feasible policies under high penalty regimes, reflecting known challenges with hard constraints and conflicting gradient signals in multi-term objectives. Details and learning curves are provided in the appendix. Algorithmic choices and learning-curve details appear in Appendix L; see also Figure 2.

Secondary Experiment: Goddard rocket. On a stochastic Goddard rocket control task (1,200 stages), TS-DDR achieves strong solutions with training in tens of minutes and sub-second inference per rollout, whereas an MPC baseline (deterministic rolling horizon) is $\sim$23% worse on average and orders of magnitude slower at execution. Full specifications and additional results are in Appendix E.

4 Conclusion

This paper introduced TS-GDR and its instantiation TS-DDR, a novel approach that integrates machine learning with stochastic optimization to solve Multistage Stochastic Optimization Problems. TS-DDR can be trained effectively by solving deterministic optimization problems in the forward pass and using closed-form gradients in the backward pass thanks to duality theory. At inference time, TS-DDR relies on the evaluation of a deep learning network. was validated on the Long-Term Hydrothermal Dispatch Problem (LTHD) and actual instances for Bolivia, a problem of significant societal impact, as well as the Goddard rocket problem. The results demonstrated the effectiveness of TS-DDR in improving solution quality while significantly reducing computation times.

TS-DDR potentially represents a significant step forward in sequential decision-making under uncertainty. By leveraging the complementary strengths of machine learning and stochastic optimization, TS-DDR offers a versatile and scalable framework. Investigating other hybrid approaches merging RL and stochastic optimization is certainly a promising direction. Other opportunities include the study of alternative parametric families for policies, applying TS-DDR in other domains, addressing the challenges posed by discrete variables, and alternative risk measures.

Limitations and avenues for future work are discussed in Appendix F.

Acknowledgments

Partly funded by NSF award 2112533. The work of Alexandre Street and Davi Valladão was partially supported by CAPES, CNPq and FAPERJ. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the Funding agencies.

References

[1] Alexander Shapiro, Darinka Dentcheva, and Andrzej Ruszczyński. *Lectures on stochastic programming: modeling and theory*. SIAM, 2009.

[2] Warren B Powell. A unified framework for optimization under uncertainty. In *Optimization challenges in complex, networked and risky systems*, pages 45–83. INFORMS, 2016.

[3] Warren B Powell. *Approximate Dynamic Programming: Solving the curses of dimensionality*, volume 703. John Wiley & Sons, 2007.

[4] Mario VF Pereira and Leontina MVG Pinto. Multi-stage stochastic optimization applied to energy planning. *Mathematical programming*, 52:359–375, 1991.

[5] PSR. Software | PSR, 2019. URL `http://www.psr-inc.com/softwares-en/`. [Online; accessed 2019-07-06].

[6] Merve Bodur and James R Luedtke. Two-stage linear decision rules for multi-stage stochastic programming. *Mathematical Programming*, pages 1–34, 2022.

[7] Felipe Nazare and Alexandre Street. Solving multistage stochastic linear programming via regularized linear decision rules: An application to hydrothermal dispatch planning. *European Journal of Operational Research*, 309(1):345–358, 2023.

[8] Andrew W Rosemberg, Alexandre Street, Joaquim Dias Garcia, Davi M Valladão, Thuener Silva, and Oscar Dowson. Assessing the cost of network simplifications in long-term hydrothermal dispatch planning models. *IEEE Transactions on Sustainable Energy*, 13(1):196–206, 2021.

[9] Csaba Szepesvári. *Algorithms for reinforcement learning*. Springer nature, 2022.

[10] Volodymyr Mnih. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

[11] John Schulman. Trust region policy optimization. *arXiv preprint arXiv:1502.05477*, 2015.

[12] Gabriel Dulac-Arnold, Nir Levine, Daniel J Mankowitz, Jerry Li, Cosmin Paduraru, Sven Gowal, and Todd Hester. Challenges of real-world reinforcement learning: definitions, benchmarks and analysis. *Machine Learning*, 110(9):2419–2468, 2021.

[13] Joshua Achiam, David Held, Aviv Tamar, and Pieter Abbeel. Constrained policy optimization. In *International conference on machine learning*, pages 22–31. PMLR, 2017.

[14] Akshay Agrawal, Brandon Amos, Shane Barratt, Stephen Boyd, Steven Diamond, and J Zico Kolter. Differentiable convex optimization layers. *Advances in neural information processing systems*, 32, 2019.

[15] MEP Maceiral, DDJ Penna, AL Diniz, RJ Pinto, ACG Melo, CV Vasconcellos, and CB Cruz. Twenty years of application of stochastic dual dynamic programming in official and agent studies in brazil-main features and improvements on the newave model. In *2018 power systems computation conference (PSCC)*, pages 1–7. IEEE, 2018.

[16] PACE. *Partnership for an Advanced Computing Environment (PACE)*, 2017. URL `http://www.pace.gatech.edu`.

[17] Kevin P Murphy. A survey of pomdp solution techniques. *environment*, 2(10), 2000.

[18] Mike Innes, Alan Edelman, Keno Fischer, Chris Rackauckas, Elliot Saba, Viral B Shah, and Will Tebbutt. A differentiable programming system to bridge machine learning and scientific computing. *arXiv preprint arXiv:1907.07587*, 2019.

[19] J Carpentier. Contribution to the economic dispatch problem. *Bulletin de la Societe Francoise des Electriciens*, 3(8):431–447, 1962.

[20] Daniel K Molzahn, Ian A Hiskens, et al. A survey of relaxations and approximations of the power flow equations. *Foundations and Trends® in Electric Energy Systems*, 4(1-2):1–221, 2019.

[21] Qiang Liu, Mengyu Chu, and Nils Thuerey. Config: Towards conflict-free training of physics informed neural networks. *arXiv preprint arXiv:2408.11104*, 2024.

[22] Chengbin Xuan, Feng Zhang, Faliang Yin, and Hak-Keung Lam. Constrained proximal policy optimization. *arXiv preprint arXiv:2305.14216*, 2023.

[23] Michael Innes, Elliot Saba, Keno Fischer, Dhairya Gandhi, Marco Concetto Rudilosso, Neethu Mariya Joy, Tejan Karmali, Avik Pal, and Viral Shah. Fashionable modelling with flux. *CoRR*, abs/1811.01457, 2018. URL `https://arxiv.org/abs/1811.01457`.

[24] Sungho Shin, François Pacaud, and Mihai Anitescu. Accelerating optimal power flow with GPUs: SIMD abstraction of nonlinear programs and condensed-space interior-point methods. *arXiv preprint arXiv:2307.16830*, 2023.

[25] Gurobi Optimization, LLC. Gurobi Optimizer Reference Manual, 2023. URL `https://www.gurobi.com`.

[26] MOSEK ApS. *The MOSEK optimization toolbox for Julia manual. Version 10.1.*, 2024. URL `https://github.com/MOSEK/Mosek.jl`.

A Background

This paper considers SDMU problems that can be modeled as Multi-stage Stochastic Programs (MSPs). The modeling with MSPs helps highlighting how to use Lagrangian duality to derive exact and informative gradients for parametric policies. Representing the uncertainty explicit makes it easier to understand how TS-DDR avoids the implicit function differentiation needed in related approaches. MSPs can be represented by mathematical programs of the form:

$$\min_{\substack{(\mathbf{y}_1,\mathbf{x}_1)\\ \text{s.t.}}} \quad f(\mathbf{x}_1,\mathbf{y}_1) \quad + \mathbb{E}_1\Big[\min_{\substack{(\mathbf{y}_2,\mathbf{x}_2)\\ \text{s.t.}}} \quad f(\mathbf{x}_2,\mathbf{y}_2) \quad \cdots \quad + \mathbb{E}_t\Big[\min_{\substack{(\mathbf{y}_t,\mathbf{x}_t)\\ \text{s.t.}}} \quad f(\mathbf{x}_t,\mathbf{y}_t) \quad + \mathbb{E}_{t+1}[\cdots]\Big]\Big]. \tag{1}$$
$$(\mathbf{y}_1,\mathbf{x}_1)\in\mathcal{X}_1(\mathbf{x}_0) \qquad (\mathbf{y}_2,\mathbf{x}_2)\in\mathcal{X}_2(\mathbf{x}_1,w_2) \qquad (\mathbf{y}_t,\mathbf{x}_t)\in\mathcal{X}_t(\mathbf{x}_{t-1},w_t)$$

which minimizes a first stage cost function $f(\mathbf{x}_1,\mathbf{y}_1)$ and the expected value of future costs over possible values of the exogenous stochastic variable $\{w_t\}_{t=2}^T \in \Omega$. Here, $\mathbf{x}_0$ is the initial system state and the control decisions $\mathbf{y}_t$ are obtained at every period t under a feasible region defined by the incoming state $\mathbf{x}_{t-1}$ and the realized uncertainty w_t. $\mathbf{E}_t$ represents the expected value over future uncertainties $\{w_\tau\}_{\tau=t}^T$. This optimization program assumes that the system is entirely defined by the incoming state, a common modeling choice in many frameworks (e.g., MDPs [17]). This is without loss of generality, since any information can be appended in the state. The system constraints can be generally posed as:

$$\mathcal{X}_t(\mathbf{x}_{t-1},w_t) = \begin{cases} \mathcal{T}(\mathbf{x}_{t-1},w_t,\mathbf{y}_t) = \mathbf{x}_t \\ h(\mathbf{x}_t,\mathbf{y}_t) \geq 0 \end{cases} \tag{2}$$

where the outgoing state of the system $\mathbf{x}_t$ is a transformation based on the incoming state, the realized uncertainty, and the control variables. The Markov Decision Process (MDPs) framework refers to $\mathcal{T}$ as the "transition kernel" of the system. State and control variables are restricted further by additional constraints captured by $h(\mathbf{x}_t,\mathbf{y}_t) \geq 0$. This paper considers policies that map the past information into decisions. In period t, an optimal policy is given by the solution of the dynamic equations [1]:

$$\begin{aligned} V_t(\mathbf{x}_{t-1},w_t) = \min_{\mathbf{x}_t,\mathbf{y}_t} \quad & f(\mathbf{x}_t,\mathbf{y}_t) + \mathbf{E}[V_{t+1}(\mathbf{x}_t,w_{t+1})] \\ \text{s.t.} \quad & \mathbf{x}_t = \mathcal{T}(\mathbf{x}_{t-1},w_t,\mathbf{y}_t) \\ & h(\mathbf{x}_t,\mathbf{y}_t) \geq 0. \end{aligned} \tag{3}$$

Since the state transition $\mathbf{x}_{t-1} \to \mathbf{x}_t$ is deterministic once the uncertainty is observed, the optimal policy can also be represented as a function of just the realized uncertainties $\{w_j\}_{j=2..t}$ and the initial state $\mathbf{x}_0$:

$$\mathbf{x}_t^* = \pi_t^*(\mathbf{x}_{t-1}^*,w_t) = \pi_t^*(\mathbf{x}_{t-1}^*(\mathbf{x}_{t-2}^*,w_{t-1}),w_t) = \ldots = \pi_t^*(\{w_j\}_{j=2..t},\mathbf{x}_0) \tag{4}$$

A.1 Value Iteration

Value iteration approaches, such as SDDP and Q-learning, try to to learn the value of an action in a particular state to inform decisions. While Q-learning is a model-free algorithm that estimates the state value by extensive evaluations, SDDP approximates the expected future cost $\mathbf{E}[V_{t+1}(\mathbf{x}_t,w_{t+1})]$ by a piece-wise convex function calculating exact derivatives that reduce the number of required evaluations. More precisely, SDDP constructs the optimal policy by computing an outer approximation $\mathcal{V}_{t+1}(\mathbf{x}_t) \approx \mathbf{E}[V_{t+1}(\mathbf{x}_t,w_{t+1})]$ and solving the resulting problem:

$$\begin{aligned} \pi_t^*(\{w_j\}_{j=2..t},\mathbf{x}_0) \in \arg\min_{\mathbf{x}_t,\mathbf{y}_t} \quad & f(\mathbf{x}_t,\mathbf{y}_t) + \mathcal{V}_{t+1}(\mathbf{x}_t) \\ \text{s.t.} \quad & \mathbf{x}_t = \mathcal{T}(\mathbf{x}_{t-1},w_t,\mathbf{y}_t) \\ & h(\mathbf{x}_t,\mathbf{y}_t) \geq 0 \end{aligned} \tag{5}$$

SDDP leverages the convexity of the problem and iteratively refines the value function $\mathcal{V}_{t+1}(\mathbf{x}_t)$ until reaching optimality. Note that SDDP computes optimal decisions for the first stage but its underlying "policy" requires solving of optimization problems for all stages.

A.2 Policy Iteration and Two-stage LDR

Policy Iteration, on the other hand, focuses on iteratively improving the policy itself. It consists of two alternating steps: The **Policy Evaluation** step that calculates the cost of the current policy, and the **Policy Improvement** step which updates the policy in order to improve value function obtained in the policy evaluation step.

The policy evaluation can be computed as follows:

$$\min_{\substack{\mathbf{y}_1 \\ \text{s.t.}(\mathbf{y}_1,\mathbf{x}_1)\in\mathcal{X}_1(\mathbf{x}_0) \\ \mathbf{x}_1=\pi_1(\mathbf{x}_0)}} \Big(f(\mathbf{x}_1,\mathbf{y}_1) \;+\; \mathbb{E}\Big[\min_{\substack{\mathbf{y}_2 \\ \text{s.t.}(\mathbf{y}_2,\mathbf{x}_2)\in\mathcal{X}_2(\mathbf{x}_1,w_2) \\ \mathbf{x}_2=\pi_2(w_2,\mathbf{x}_0)}} \Big(f(\mathbf{x}_2,\mathbf{y}_2) \;+\; \mathbb{E}\Big[\cdots \min_{\substack{\mathbf{y}_t \\ \text{s.t.}(\mathbf{y}_t,\mathbf{x}_t)\in\mathcal{X}_t(\mathbf{x}_{t-1},w_t) \\ \mathbf{x}_t=\pi_t(\{w_j\}_{j=2..t},\mathbf{x}_0)}} \Big(f(\mathbf{x}_t,\mathbf{y}_t) + \; \mathbb{E}[\cdots] \Big) \cdots \Big] \Big) \Big] \Big) \tag{6}$$

In this problem, the optimization in stage t only decides over y_t since the state comes directly from the policy: $\mathbf{x}_t = \pi_t(\{w_j\}_{j=2..t}, \mathbf{x}_0)$. Moreover, since the only term in the objective of stage t that depends on y_t is the immediate cost $f(x_t, y_t)$, the problem becomes independent of the subsequent stages:

$$\begin{aligned} \arg\min_{\mathbf{y}_t} \; & f(\mathbf{x}_t,\mathbf{y}_t) + \overbrace{\mathbf{E}_{t+1}[\cdots]}^{\text{Independent of } y_t} \\ & \text{s.t. } (\mathbf{y}_t,\mathbf{x}_t) \in \mathcal{X}_t(\mathbf{x}_{t-1},w_t) \\ & \quad \mathbf{x}_t = \pi_t(\{w_j\}_{j=2..t},\mathbf{x}_0) \end{aligned} \quad = \quad \begin{aligned} \arg\min_{\mathbf{y}_t} \; & f(\mathbf{x}_t,\mathbf{y}_t) \\ & \text{s.t. } (\mathbf{y}_t,\mathbf{x}_t) \in \mathcal{X}_t(\mathbf{x}_{t-1},w_t) \\ & \quad \mathbf{x}_t = \pi_t(\{w_j\}_{j=2..t},\mathbf{x}_0) \end{aligned} \tag{7}$$

Therefore, the optimal cost of stage t can be decomposed, and the min and $\mathbf{E}_{t+1}$ operators can be permuted:

$$\begin{aligned} \min_{\mathbf{y}_t} \; & f(\mathbf{x}_t,\mathbf{y}_t) + \mathbf{E}_{t+1}[\cdots] \\ & \text{s.t. } (\mathbf{y}_t,\mathbf{x}_t) \in \mathcal{X}_t(\mathbf{x}_{t-1},w_t) \\ & \quad \mathbf{x}_t = \pi_t(\{w_j\}_{j=2..t},\mathbf{x}_0) \end{aligned} \quad = \quad \begin{aligned} \min_{\mathbf{y}_t} \; & f(\mathbf{x}_t,\mathbf{y}_t) + \mathbf{E}_{t+1}[\cdots] \\ & \text{s.t. } (\mathbf{y}_t,\mathbf{x}_t) \in \mathcal{X}_t(\mathbf{x}_{t-1},w_t) \\ & \quad \mathbf{x}_t = \pi_t(\{w_j\}_{j=2..t},\mathbf{x}_0) \end{aligned} \tag{8}$$

$$= \quad \mathbf{E}_{t+1}\left[\begin{aligned} \min_{\mathbf{y}_t} \; & f(\mathbf{x}_t,\mathbf{y}_t) \\ & \text{s.t. } (\mathbf{y}_t,\mathbf{x}_t) \in \mathcal{X}_t(\mathbf{x}_{t-1},w_t) \\ & \quad \mathbf{x}_t = \pi_t(\{w_j\}_{j=2..t},\mathbf{x}_0) \end{aligned} \right] + \mathbf{E}_{t+1}[\cdots] \tag{9}$$

$$= \quad \mathbf{E}_{t+1}\left[\begin{aligned} \min_{\mathbf{y}_t} \; & f(\mathbf{x}_t,\mathbf{y}_t) \quad + \quad \cdots \quad + \\ & \text{s.t. } (\mathbf{y}_t,\mathbf{x}_t) \in \mathcal{X}_t(\mathbf{x}_{t-1},w_t) \\ & \quad \mathbf{x}_t = \pi_t(\{w_j\}_{j=2..t},\mathbf{x}_0) \end{aligned} \right] \tag{10}$$

Trivially applying the same logic for all stages and combining expectations, the policy evaluation problem becomes:

$$\mathbf{E}_1\left[\mathbf{E}_2\left[\begin{aligned} & \cdots \min_{\mathbf{y}} \; \sum_t f(\mathbf{x}_t,\mathbf{y}_t) \cdots \\ & \quad \text{s.t. } (\mathbf{y}_t,\mathbf{x}_t) \in \mathcal{X}_t(\mathbf{x}_{t-1},w_t) \; \forall t \\ & \quad\quad \mathbf{x}_t = \pi_t(\{w_j\}_{j=2..t},\mathbf{x}_0) \qquad \forall t \end{aligned} \right] \right] = \mathbf{E}\left[\begin{aligned} & \mathcal{Q}(w) = \min_{\mathbf{y}} \; \sum_t f(\mathbf{x}_t,\mathbf{y}_t) \\ & \quad \text{s.t. } (\mathbf{y}_t,\mathbf{x}_t) \in \mathcal{X}_t(\mathbf{x}_{t-1},w_t) \; \forall t \\ & \quad\quad \mathbf{x}_t = \pi_t(\{w_j\}_{j=2..t},\mathbf{x}_0) \qquad \forall t \end{aligned} \right]$$

where $\mathcal{Q}(w)$ is the Policy evaluation for a single sample $\{w_t\}_{t=2}^T \in \Omega$ - a deterministic (large-scale) single-stage problem. Thus, the policy iteration for a policy $\pi_t(\{w_j\}_{j=2..t}, \mathbf{x}_0; \theta_t)$ parametrized by θ in original MSP problem (1) is trying to solve the following **Two-Stage** equivalent problem:

$$\min_{\theta} \quad \mathbf{E}[\mathcal{Q}(w;\theta)] \tag{11}$$

In [6], authors propose to learn linear policies using a slightly modified version of the the above policy iteration method, calling it **Two-Stage Linear Decision Rules (LDR)**. In their framework, the policy evaluation for a single sample is defined by:

$$\begin{aligned}
\min_{\mathbf{x},\mathbf{y},\delta} \quad & \sum_t f(\mathbf{x}_t, \mathbf{y}_t) + C_t^{\delta}||\delta_t|| && (12)\\
\text{s.t.} \quad & \mathbf{x}_t = \mathcal{T}(\mathbf{x}_{t-1}, w_t, \mathbf{y}_t) && \forall t\\
& \mathbf{x}_t + \delta_t = \sum_{j=2}^{t} \theta_{t,j} w_j + \theta_{t,1}\mathbf{x}_0 \ : \lambda_t && \forall t\\
& h(\mathbf{x}_t, \mathbf{y}_t) \geq 0 && \forall t
\end{aligned}$$

Now, the policy, $\sum_{j=2}^{t} \theta_{t,j} w_j + \theta_{t,1}\mathbf{x}_0$, defines the target state to attain at every period t, while the cost is a result of the optimal control variable $\mathbf{y}$ and the slack variable δ that represents target infeasibility. Adding the slack variable δ is a common practice in stochastic programming to guarantee control feasibility for any θ, ensuring *relatively complete recourse* [1]. λ_t is the dual of the target constraint, i.e., the derivative of the objective function with respect to the target defined by the policy. If the second stage (12) is convex, the problem may be solved by benders decomposition (or any similar method). Since the expectation is often computed through sampling, this approach typically suffers from over-fitting; which can be mitigated using regularization [7].

B Policy Types and Further Training details of TS-DDR

Time-Specific Policies A time-specific policy π for TS-DDR can then be expressed as

$$\pi(\{w_j\}_{j=2..T}, \mathbf{x}_0; \theta) = \begin{bmatrix} \pi_t(w_1, \mathbf{x}_0; \theta_1) \\ \vdots \\ \pi_t(\{w_j\}_{j=2..T}, \mathbf{x}_0; \theta_T) \end{bmatrix} \tag{13}$$

and obtained using an automatic differentiation framework (e.g., Zygote.jl [18]) and a compatible optimizer (e.g., Gradient Descent, Adam, AdaGrad) to update the policy parameters.

Time-Invariant Policies An additional contribution of this paper with respect to TS-LDR is the implementation of Time-Invariant Policies for TS-DDR. Indeed, TS-LDR learns separate policies $\pi_1, \ldots, \pi_T$ for each stage; these policies have the signature $\pi_t(\{w_j\}_{j=2..t}, \mathbf{x}_0; \theta)$, where $\{w_j\}_{j=2..t}$ represents the entire sequence of past uncertainties and $\mathbf{x}_0$ is the initial state. This means each stage has a different number of inputs since the history grows larger. TS-DDR uses $(\hat{\mathbf{x}}_t, \ell_t) \leftarrow \pi(w_t^s, \ell_{t-1}; \theta_i)$, a single time-invariant policy inspired by recurrent networks. The policy evaluation at each stage t shares a hidden (latent) state ℓ_t with the subsequent stage $t+1$. Trough training, ℓ_t carries the necessary information about $\{w_j\}_{j=2..t}$ and $\mathbf{x}_0$. Since the policy uses the same parameters at every stage, it generalizes beyond the finite training horizon, which is a distinct advantage. Indeed, it requires less computational effort to train and fewer computational resources at execution time as discussed in the next section.

The Training Algorithm The training algorithm for TS-DDR is shown in Algorithm 1. Lines 6-13 are the forward pass: they compute the target states (lines 6-12) before solving a second stage subproblem through optimization. Lines 14-15 and line 17 are the backward pass. It is interesting to highlight the similarities and differences between RL and TS-DDR. Both share a forward phase evaluating various scenarios with decisions over time and an update of the policy parameters through gradient computations. However, TS-DDR uses dedicated optimization technology to minimize costs and ensure feasibility in its forward phase, as well as exact gradients obtained through the nature of the optimizations and duality theory. *This integration of RL and multi-stage stochastic optimization is a key contribution of this paper for solving MSPs.*

Algorithm 1 TS-DDR Policy Estimation

procedure TS-DDR($\pi(\cdot;\theta)$, M, S, T, η)
  Input: $\pi(\cdot;\theta)$ ▷ initial policy parameters
  $\theta \leftarrow \theta_0$ ▷ set initial parameters
  for $i \leftarrow 1$ **to** M **do** ▷ outer training loop
    $\mathbf{x}_0 \leftarrow$ initial state conditions
    $\left\{\{w_t^s\}_{t=1}^T\right\}_{s=1}^S \leftarrow$ sample the stochastic process S times
    for $s \leftarrow 1$ **to** S **do** ▷ Monte-Carlo rollouts
      $\ell_0 \leftarrow$ initial latent state
      $(\hat{\mathbf{x}}_1, \ell_1) \leftarrow \pi(\mathbf{x}_0, \ell_0; \theta)$ ▷ first-stage target
      for $t \leftarrow 2$ **to** T **do**
        $(\hat{\mathbf{x}}_t, \ell_t) \leftarrow \pi(w_t^s, \ell_{t-1}; \theta)$ ▷ remaining targets
      end for
      loss $\leftarrow \mathcal{Q}\big(w^s; \hat{\mathbf{x}}\big)$ ▷ implementation loss
      $\lambda^s \leftarrow \nabla_{\hat{\mathbf{x}}} \mathcal{Q}\big(w^s, \hat{\mathbf{x}}\big)$ ▷ duals
      $\nabla_\theta \mathcal{Q}\big(w^s; \theta\big) \leftarrow \lambda^s \odot \nabla_\theta \pi$ ▷ loss gradient
    end for
    $\theta \leftarrow \theta + \eta \dfrac{1}{S} \displaystyle\sum_{s=1}^{S} \nabla_\theta \mathcal{Q}\big(w^s; \theta\big)$ ▷ parameter update
  end for
  return θ
end procedure

C Decision Making process

To make first-stage decisions, SDDP (a time-specific method) iterates over scenarios to approximate future outcomes. It considers future decisions just enough to provide some guarantees regarding future costs. This process does not ensure that future policies have converged to their optimal state. Moreover, future periods are likely to encounter states not simulated during training. As a result, decision makers must rerun the SDDP procedure over time, even before the initial training horizon ends, which is highly time-consuming. Time-specific models, such as Two-Stage LDR, may encounter similar issues since they do not account for all possible scenarios. However, this is effectively mitigated by training under appropriate model assumptions and using regularization, as demonstrated in [7]. Time-specific models also require retraining at the end of the considered time horizon. TS-GDR utilizes regularization via SGD and employs a typical "training-validation-test" split process to check for convergence. Additionally, with time-invariant policies, TS-DDR avoids the finite horizon limitation, further reducing the need for retraining.

D Detailed Long-Term Hydrothermal Dispatching Case and Additional Results

Problem Description: Energy storage is a cornerstone in the quest for sustainable energy solutions, offering a critical avenue for emission reduction and grid optimization. Their dispatchable nature mitigates the intermittency of renewable energy sources like wind and solar, thus enhancing grid stability and enabling greater reliance on clean energy. Among the different types of storage, hydro reservoirs are prominent large-scale energy storage systems. These reservoirs are capable of efficiently storing and releasing vast amounts of energy. Their mechanism, that involves the storing of water during periods of low demand and the subsequent releases to generate electricity during peak hours, provides a critical means of balancing supply and demand.

The optimization of sequential decision strategies (policies) to manage energy storage systems and minimize energy dispatch costs is referred to as the LTHD problem. The LTHD is a complicated MSP problem as it needs to take into account the complex power flow equations that represent the physics of electricity transmission. The non-convex nature of AC power flow inhibits the use of classical optimization methods such as SDDP. Moreover, other control and ML methods (such as RL) have

difficulties scaling with decision and scenario dimensions. It is an important case study to evaluate the benefits of any novel method. The second-stage policy implementation is described in Appendix D.1.

D.1 LTHD Policy Implementation Problem

$$Q(w;\hat{\mathbf{x}}) = \min_{\mathbf{x},\mathbf{y},\delta} \sum_{t} \sum_{i\in\mathcal{I}} C_{it}\, p^{g}_{it} \;+\; C^{\delta}_{t}\, \|\delta_t\| \tag{14a}$$

$$\text{s.t.} \quad \mathbf{x}_{jt} + u_{jt} + s_{jt} \;=\; \mathbf{x}_{j,t-1} + A_{j,t}(\omega_t) + \sum_{k\in\mathcal{H}^U_j} u_{kt} \;+\; \sum_{k\in\mathcal{H}^S_j} s_{kt}, \quad \forall\, t,\ j\in\mathcal{H}, \tag{14b}$$

$$u_{jt} = \Phi_j\, p^{g}_{jt}, \quad \forall\, t,\ j\in\mathcal{H}, \tag{14c}$$

$$\mathbf{x}_t + \delta_t \;=\; \hat{\mathbf{x}}_t \quad : \lambda_t, \quad \forall\, t, \tag{14d}$$

$$p^{g}_{t} \;\in\; \text{PF}_t, \quad \forall\, t. \tag{14e}$$

The second-stage policy implementation for the LTHD problem is described in Equation 14. The objective (14a) is to minimize the total cost of generation dispatch while meeting electricity demand and adhering to physical and engineering constraints. State variables $\mathbf{x}$ represent the volumes of the hydro reservoirs, while control variables u denote the outflows from these reservoirs, and $\mathbf{p}$, the energy production variables for all generators (not only hydro). δ are the deviation variables ensuring that the state meets the target. The model considers uncertain scenarios through ω_t, which is accounted in the stochastic inflow of water to the reservoirs, $A_{j,t}(\omega_t)$. s_{jt} represents the spillage, which is the excess water that cannot be stored and must be released.

The key constraints of the model include: (14b) which ensures the balance of water volumes in the reservoirs by accounting for inflows ($A_{j,t}(\omega_t)$), outflows (u_{jt}), spillage (s_{jt}), and the flow (u_{kt}, s_{kt}) from upstream reservoirs ($\mathcal{H}^U, \mathcal{H}^S$). (14c) relates the outflows to energy production via the production factor Φ_j. (14d) ensures that the state and deviation δ_t match the target $\hat{\mathbf{x}}_t$, with λ_t representing the dual variables for these constraints. Finally, (14e) ensures that the energy dispatch respects power flow equations and generator limits defined by the set PF_t for each stage t.

The most accurate formulation of the physical constraints, PF_t, were formalized in [19] to describe AC power-flow (AC-PF). Annex G provides a general overview of these constraints. The AC-PF model is a non-convex non-linear problem (NLP), not suitable for the classical SDDP algorithm. Thus, as in many applications, convex approximations and relaxations can be used to meet the SDDP convexity assumption [20]. Specifically, planning agents utilize simplified models to compute cost-to-go functions and couple them to optimization problems that guarantee a feasible operative decision. Annex I details the coupling of planning cost-to-go functions to find implementable and how to simulate similar polices efficiently.

Experimental Setting: *The results are reported for linear, conic, and nonlinear nonconvex problems to consider fundamental different problem structures.* These are obtained from the DCLL approximation of power systems (linear), the Second-Order Cone (SOC) relaxation (conic), and the AC-PF (non-convex). See Appendix H and G.1 for the descriptions of linear and conic cases. Appendix J provides the computational resources and the hyperparameters used.

The evaluation compares the time-invariant TS-DDR, SDDP, and TS-LDR. For the linear and conic case, SDDP is optimal for stage-independent uncertainty, which allows for estimating how accurate TS-DDR is. The evaluation also compares the inconsistent policies trained under SOC and DCLL with TS-DDR. The evaluation also compares TS-DDR with TS-LDR [6]. However, instead of using explicit regularization to avoid over-fitting, as in [7], the implementation uses SGD.

The 28-bus Case Study from Bolivia: To examine the scalability of TS-DDR under different power flow formulations, the results employ a realistic case study based on the Bolivian power system. Results on a smaller 3-bus system are shown in the Appendix for completeness with prior results. The Bolivian system comprises 28 buses, 26 loads, 34 generators (including 11 hydro units), and 31 branches, with a predominantly radial configuration and only three loops. The planning horizon extends to 96 periods. For each stage, the evaluation uses 165 scenarios from historical data.

Results: Tables 1-3 shows the out-of-sample test results for different policies in the LTHD problem instances. Tables have 5 columns: (1) "Model" signaling the policy type; (2) "Plan" representing the power flow formulation used in training; (3) "Imp Costs" meaning the average grid operational costs under the power flow formulation described in the table caption ($\pm$ the standard deviation); (4) "GAP" making explicit the average percentage difference of the models' *Imp Costs* with the best model ($\pm$ the standard deviation); (5) "Training" presenting the amount of time in training the policy; and (6) "Execution" shows the amount of time need for producing implementable policies. For each experiment ("Case & Formulation"), policies are ordered by their implementation costs.

In the convex cases (Tables 1-2), the SDDP Implementation (Imp) cost serves as the reference point for the calculation of the optimality GAP. However, as previously discussed, when the planning formulation differs from the implementation one, and the resulting SDDP policy is time-inconsistent and thus sub-optimal. The GAP, in Table3, is then calculated with respect to the best performing model.

By not (over) fitting to the finite horizon of the experiments, the TS-DDR policy is sub-optimal, which explains its non-zero optimality GAP. In fact, the TS-DDR polices have higher implementation costs than other benchmarks in both the DCLL and AC formulations for the 3-bus case. Nevertheless, the optimality to the best found policy implementation cost decreases as the problem becomes bigger and the assumptions of the remaining benchmarks are increasingly violated. For the Bolivian case under the AC (non-convex) formulation, the TS-DDR policy is the best performing policy beating all other policies in both terms of training computational resources and implementation costs.

More precisely, on the convex cases, TS-DDR is within 0.63% and 0.49% of optimality and is 4 orders of magnitude faster than SDDP at execution time. Interestingly, even the training time of TS-DDR is an order of magnitude faster than the SDDP execution times, On the non-convex case, TS-DDR outperforms SDDP by at least 0.32% in solution quality, and brings orders of magnitude improvements in solution times again. TS-DDR also produces order of magnitude improvements in solution quality compared to TS-LDR, highlighting the benefits of complex policy representations.

Reinforcement Learning: In order to assess the advantage of using TS-DDR compared to traditional RL methods, the evaluation considers the application of standard algorithms in the RL literature to LTHD. Appendix L provides a description of these algorithms. Figure 2 highlights that some methods showed promising initial results with cumulative rewards increasing over time (i.e., a decrease in operating costs). However, the improvements then level off and the methods stagger at a very low quality solution. Moreover, no RL model-free method was able to find feasible policies - i.e. in which there is no discrepancy between the target state $\hat{\mathbf{x}}_t$ and the achieved state $\mathbf{x}_t$. These results highlight the challenges documented in constrained learning spaces such as those encountered in training Physics-Informed Neural Networks (PINNs). As noted by [21], learning problems involving multiple additive terms with conflicting objectives, like penalties for satisfying various constraints, leads to gradients with conflicting update directions. This suggests that achieving better results with model-free RL for the LTHD may require careful tuning of constraint penalties and a more balanced trade-off with the reward function, as seen in approaches like [22]. Nevertheless, no successful RL approach has been proposed for the general case. This paper proposes TS-DDR as a model-based constrained RL approach that can be used along side other important techniques from the RL literature to overcome other issues such as epistemic uncertainty.

Storage Behavior: Figure 3 shows the expected stored energy (volume) and thermal generators dispatch over time for each analysed policy in the Bolivian grid under the AC power flow formulation. TS-DDR stores less water over time compared to most of the other policies. It is capable of finding a cost effective strategy that avoids more expensive generators. SDDP polices, in contrast, tend to exhibit higher needs for stored energy. This is in line with the findings in [8].

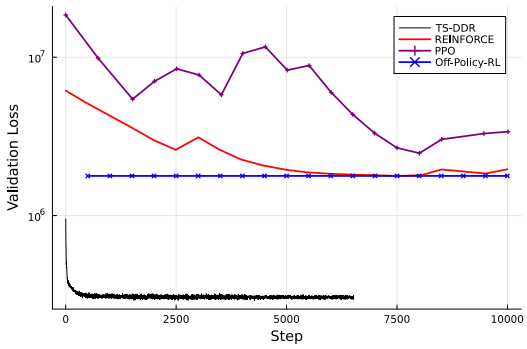

Figure 2: LTHD Training Curves for Model-Free RL Methods and TS-GDR Baseline

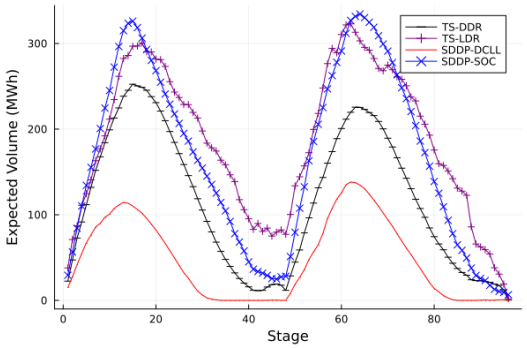

Figure 3: Expected Stored Energy for the AC Formulation.

E The Goddard Rocket Control Problem

To showcase the capability of TS-DDR to learn other complex policies, this section considers the Goddard Rocket control (GRC) problem from "Benchmarking Optimization Software with COPS 3.0". The evaluation considers 1,200 stages and the test case was modified to have a random force, $w_t \in \mathcal{N}(0,1)$, affecting the velocity transition equation, i.e.,

$$\frac{v_t - v_{t-1}}{\Delta t} = \frac{u_{t-1} - D(h_{t-1}, v_{t-1})}{m_{t-1}} - g(h_{t-1}) \overbrace{-w_{t-1}}^{\text{random force}}$$

While SDDP is a stochastic version of the common Model Predictive Control (MPC) approach, it requires convexity and, for non-convex applications, convex relaxations and approximations need be used for estimating the value functions. However, for the GRC problem, no such convex approximation is readily available, therefore the fallback is to solve the deterministic version of the multi-stage problem and update initial conditions based on the realized uncertainty in a rolling horizon fashion. This is suboptimal in general and, for the GRC problem, the gap with TS-DDR is 23%. Table 4 also highlights the significant computational benefits of TS-DDR.

Table 4: Comparison of MPC and TS-DDR for the Goddard Case.

Model	**Training**	**Inference**	**GAP** %
TS-DDR	35 min.	0.091 sec.	–
MPC	None	2 hours	23.42($\pm$12.62)

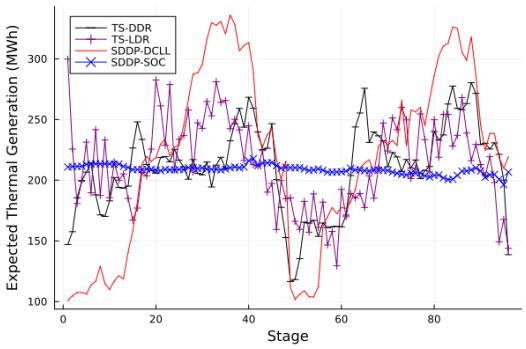

Figure 4: Expected Thermal Dispatch for the AC Formulation.

F Limitations

There are a number of limitations in this study, as well as additional opportunities. First, TS-GDR is a general framework and the paper only considers one of its instantiations: it would be interesting to explore other machine learning models. Second, the LTHD problem is particularly difficult for RL due to its complexity and high dimensionality. It would be interesting however to study how TS-GDR behaves on applications where RL has been effective, to gain deeper insights into each approach's strengths and weaknesses. Third, the scalability of TS-DDR on extremely large instances needs to be investigated. This paper focuses on an application for which SDDP was applicable. Evaluating TS-DDR for applications outside the reach of SDDP is an important direction. Fourth, it would be interesting to evaluate TS-DDR on some additional applications where its strengths can be highlighted. There is a scarcity of test cases in this domain and the team aims at addressing this limitation. Figure 4 highlights another source of possible improvement for the two stage policies, TS-DDR and TS-LDR, which exhibit a larger variance in the amount of thermal generation dispatch since, in some applications, control actuators might have frequency restrictions. This could be solved by either modeling these restrictions explicitly or by adding regularization penalties for intermittent behaviour. Appendix J shows the same pictures for both the SOC and DCLL formulations.

G Alternating Current (AC) - Power Flow

This appendix details the formulation of the Alternating Current Power Flow (AC-PF) problem, which is fundamental for analyzing and optimizing electrical power systems. The AC-PF model captures the complex relationships between voltages, power generation, and power consumption across the network, ensuring adherence to physical laws and operational constraints. The following equations (15) provide a comprehensive mathematical representation of these relationships, incorporating key constraints such as Kirchhoff's current law, Ohm's law, thermal limits, and bounds on voltage and power generation. By attending the AC-PF constraints, feasibility of the power generation dispatch to meet demand is ensured while maintaining system stability and efficiency. For clarity, certain complexities such as transformer tap ratios, phase angle difference constraints, and reference voltage constraints are not included in this presentation.

$$
\begin{aligned}
\text{PF} = \Big\{ \mathbf{p} \mid \mathbf{S}^{\text{g}}_i - \mathrm{S}^{\text{d}}_i - (Y^s_i)^\star |\mathbf{V}_i|^2 = \sum_{ij \in \mathcal{E} \cup \mathcal{E}^R} \mathbf{S}^{\text{f}}_{ij} \qquad & \forall i \in \mathcal{N} && \text{(15a)} \\
\mathbf{S}^{\text{f}}_{ij} = (Y_{ij} + Y^c_{ij})^\star |\mathbf{V}_i|^2 - Y^\star_{ij} \mathbf{V}_i \mathbf{V}^\star_j \qquad & \forall ij \in \mathcal{E} && \text{(15b)} \\
\mathbf{S}^{\text{f}}_{ji} = (Y_{ij} + Y^c_{ji})^\star |\mathbf{V}_j|^2 - Y^\star_{ij} \mathbf{V}^\star_i \mathbf{V}_j \qquad & \forall ij \in \mathcal{E} && \text{(15c)} \\
|\mathbf{S}^{\text{f}}_{ij}|, |\mathbf{S}^{\text{f}}_{ji}| \leq \bar{s}_{ij} \qquad & \forall ij \in \mathcal{E} && \text{(15d)} \\
\underline{\mathrm{v}}_i \leq |\mathbf{V}_i| \leq \overline{\mathrm{v}}_i \qquad & \forall i \in \mathcal{N} && \text{(15e)} \\
\underline{\mathrm{p}}_i \leq \mathbf{p}_i \leq \overline{\mathrm{p}}_i \qquad & \forall i \in \mathcal{N} && \text{(15f)} \\
\underline{\mathrm{q}}_i \leq \mathbf{q}_i \leq \overline{\mathrm{q}}_i \qquad & \forall i \in \mathcal{N} \Big\} && \text{(15g)}
\end{aligned}
$$

Equation 15 presents the AC-PF formulation, in complex variables. Constraints (15a) enforce power balance (Kirchhoff's current law) at each bus. Constraints (15b) and (15c) express Ohm's law on forward and reverse power flows, respectively. Constraints (15d) enforce thermal limits on forward and reverse power flows. Finally, constraints (15e)–(15g) enforce minimum and maximum limits on nodal voltage magnitude, active generation, and reactive generation, respectively.

G.1 Second Order Cone (SOC) - Power Flow

This appendix introduces the Second-Order Cone Power Flow (SOC-PF) formulation, a convex relaxation of the AC-PF model. The SOC-PF model simplifies the problem by introducing additional variables and relaxing certain non-convex constraints, resulting in a more tractable problem while maintaining a close approximation to the original AC-PF. The following equations (16) encapsulate the SOC-PF formulation, addressing active and reactive power balance, Ohm's law, thermal limits, voltage bounds, and generation constraints.

$$
\begin{aligned}
\text{PF} = \Big\{ \mathbf{p} \mid \mathbf{p}_i - \mathrm{p}^{\mathrm{d}}_i - g^s_i \mathbf{w}_i = \sum_{ij \in \mathcal{E} \cup \mathcal{E}^R} \mathbf{f}_{ij} \qquad & \forall i \in \mathcal{N} && \text{(16a)}\\
\mathbf{q}_i - \mathrm{q}^{\mathrm{d}}_i + b^s_i \mathbf{w}_i = \sum_{ij \in \mathcal{E} \cup \mathcal{E}^R} \mathbf{f}^{\mathrm{q}}_{ij} \qquad & \forall i \in \mathcal{N} && \text{(16b)}\\
\mathbf{f}_{ij} = \gamma^{p}_{ij} \mathbf{w}_i + \gamma^{p,r}_{ij} \mathbf{w}^{\mathrm{r}}_{ij} + \gamma^{p,i}_{ij} \mathbf{w}^{\mathrm{i}}_{ij} \qquad & \forall ij \in \mathcal{E} && \text{(16c)}\\
\mathbf{f}^{\mathrm{q}}_{ij} = \gamma^{q}_{ij} \mathbf{w}_j + \gamma^{q,r}_{ij} \mathbf{w}^{\mathrm{r}}_{ij} + \gamma^{q,i}_{ij} \mathbf{w}^{\mathrm{i}}_{ij} \qquad & \forall ij \in \mathcal{E} && \text{(16d)}\\
\mathbf{f}_{ji} = \gamma^{p}_{ji} \mathbf{w}_i + \gamma^{p,r}_{ji} \mathbf{w}^{\mathrm{r}}_{ij} + \gamma^{p,i}_{ji} \mathbf{w}^{\mathrm{i}}_{ij} \qquad & \forall ij \in \mathcal{E} && \text{(16e)}\\
\mathbf{f}^{\mathrm{q}}_{ji} = \gamma^{q}_{ji} \mathbf{w}_j + \gamma^{q,r}_{ji} \mathbf{w}^{\mathrm{r}}_{ij} + \gamma^{q,i}_{ji} \mathbf{w}^{\mathrm{i}}_{ij} \qquad & \forall ij \in \mathcal{E} && \text{(16f)}\\
(\mathbf{f}_{ij})^2 + (\mathbf{f}^{\mathrm{q}}_{ij})^2 \leq \bar{s}^2_{ij} \qquad & \forall ij \in \mathcal{E} \cup \mathcal{E}^R && \text{(16g)}\\
(\mathbf{w}^{\mathrm{r}}_{ij})^2 + (\mathbf{w}^{\mathrm{i}}_{ij})^2 \leq \mathbf{w}_i \mathbf{w}_j \qquad & \forall ij \in \mathcal{E} && \text{(16h)}\\
\underline{\mathrm{v}}^2_i \leq \mathbf{w}_i \leq \overline{\mathrm{v}}^2_i \qquad & \forall i \in \mathcal{N} && \text{(16i)}\\
\underline{\mathrm{p}}_i \leq \mathbf{p}_i \leq \overline{\mathrm{p}}_i \qquad & \forall i \in \mathcal{N} && \text{(16j)}\\
\underline{\mathrm{q}}_i \leq \mathbf{q}_i \leq \overline{\mathrm{q}}_i \qquad & \forall i \in \mathcal{N} \Big\} && \text{(16k)}
\end{aligned}
$$

The SOC-PF formulation introduces additional variables:

$$
\begin{aligned}
\mathbf{w}_i &= \mathbf{v}^2_i, && \forall i \in \mathcal{N} && \text{(17)}\\
\mathbf{w}^{\mathrm{r}}_{ij} &= \mathbf{v}_i \mathbf{v}_j \cos(\theta_j - \theta_i), && \forall ij \in \mathcal{E} && \text{(18)}\\
\mathbf{w}^{\mathrm{i}}_{ij} &= \mathbf{v}_i \mathbf{v}_j \sin(\theta_j - \theta_i), && \forall ij \in \mathcal{E} && \text{(19)}
\end{aligned}
$$

The non-convex constraint:

$$
(\mathbf{w}^{\mathrm{r}}_{ij})^2 + (\mathbf{w}^{\mathrm{i}}_{ij})^2 = \mathbf{w}_i \mathbf{w}_j, \qquad \forall ij \in \mathcal{E} \qquad \text{(20)}
$$

is relaxed to:

$$
(\mathbf{w}^{\mathrm{r}}_{ij})^2 + (\mathbf{w}^{\mathrm{i}}_{ij})^2 \leq \mathbf{w}_i \mathbf{w}_j, \qquad \forall ij \in \mathcal{E} \qquad \text{(21)}
$$

Equation 16 presents the SOC-PF formulation using real variables. Constraints (16a) and (16b) enforce Kirchhoff's current law for active and reactive power at each node. Constraints (16c)–(16e) capture Ohm's law on active and reactive power flows. The γ parameters derive from substituting variables $\mathbf{w}, \mathbf{w}^{\mathrm{r}}, \mathbf{w}^{\mathrm{i}}$ in (15b)–(15c). Constraints (16g) enforce thermal limits on power flows. Constraint (16h) is Jabr's inequality. Finally, constraints (16i)–(16k) enforce limits on nodal voltage magnitude, active, and reactive generation.

The SOC-PF formulation is nonlinear and convex, making it more tractable than AC-PF and solvable using polynomial-time interior-point algorithms. As a relaxation of AC-PF, SOC-PF provides valid dual bounds on the optimal value of AC-PF.

H Direct Current with Line Losses (DCLL) - Power Flow

This appendix introduces the Direct Current with Line Losses Power Flow (DCLL-PF) formulation, a quadratic approximation of AC-PF. This approximation assumes all voltage magnitudes are one per-unit, voltage angles are small, losses are quadratically proportional to the flow, and reactive power is ignored. The DCLL-PF improves on the DC-PF by approximating line losses, making it a more accurate linear approximation widely used in electricity markets and planning problems.

$$
\begin{aligned}
\text{PF} = \Big\{ \mathbf{p} \mid \mathbf{p}_i + \sum_{ji \in \mathcal{E}} \mathbf{f}_{ji} - \sum_{ij \in \mathcal{E}} \mathbf{f}_{ij} = \mathrm{p}_i^{\mathrm{d}} \quad & \forall i \in \mathcal{N} && \text{(22a)} \\
\mathbf{f}_{ij} = b_{ij}(\theta_j - \theta_i) \quad & \forall ij \in \mathcal{E} && \text{(22b)} \\
\mathbf{f}_{ij} + \mathbf{f}_{ji} \geq \frac{g_{ij}}{g_{ij}^2 + b_{ij}^2} \mathbf{f}_{ij}^2 \quad & \forall ij \in \mathcal{E} && \text{(22c)} \\
|\mathbf{f}_{ij}| \leq \bar{s}_{ij} \quad & \forall ij \in \mathcal{E} && \text{(22d)} \\
\underline{\mathrm{p}}_i \leq \mathbf{p}_i \leq \overline{\mathrm{p}}_i \quad & \forall i \in \mathcal{N} \Big\} && \text{(22e)}
\end{aligned}
$$

Equation 22 presents the DCLL-PF formulation using a linear programming (LP) approach. Constraints (22a) enforce active power balance at each node. Constraints (22b) approximate Ohm's law using a phase-angle formulation. Constraints (22c) ensure the consideration of line losses. Constraints (22d) enforce thermal constraints on each branch. Constraints (22e) enforce limits on active power generation. Constraints on phase angle differences and slack bus are omitted for readability but are implemented in numerical experiments.

I Implementable Decisions in Stochastic Dual Dynamic Programming (SDDP)

This appendix details how decision-makers use convex models to get implementable decisions for non-convex systems.

In SDDP, planning agents utilize simplified (convex) models to compute cost-to-go functions, which guide decision-making over medium- and long-term horizons. These functions provide insights into the optimal reservoir levels and system operation strategies. However, for operational implementation, it is essential to ensure that the derived decisions align with the intricacies of the real network.

To translate planning decisions into operational actions, Independent System Operators (ISOs) employ a coupling approach. Initially, the SDDP algorithm converges using simplified convex network models. Subsequently, the second-stage cost-to-go function is integrated into more detailed models that offer a realistic portrayal of the system.

A common strategy employed by ISOs involves a rolling-horizon operating scheme. In this approach, decisions obtained from the planning stage, embedded in the cost-to-go function, are periodically updated with real-time data. For instance, in regions like Brazil and Chile, ISOs update the state variables, such as reservoir levels, and re-converge SDDP in subsequent periods.

However, fitting a policy under a simplified convex model but implementing decisions in the detailed non-convex reality produces what is called time-inconsistent (sub-optimal) policies [1]. As shown in [8], time-inconsistency increases power system operational costs, produces higher energy prices and increases emissions. Nevertheless, results from the aforementioned paper indicate that tight convex approximations and relaxations, such as the second-order cone (SOC) relaxation and the Direct-Current with line losses (DCLL) quadratic approximation, can greatly mitigate these negative effects, although still under a significant computational hurdle.

To evaluate the performance of time-inconsistent policies induced by model approximations for long horizons in a tractable manner, a modified version of the SDDP algorithm is used. The modified version performs forward passes using the AC power-flow model and the backward passes, that calculate the future cost functions, using the relaxation - this allows us to fit the future cost functions to the state values that would be visited in the rolling horizon.

J Additional Results 28-Bus Case

This provides additional details and results about the 28-Bus Case.

Policy and optimization details:

- All ML models are implemented in Julia using *Flux.jl* [23].
- Weights Optimizer: $Adam(\eta = 0.001, \beta :: Tuple = (0.9, 0.999), \epsilon = 1.0\ 10^{-8})$
- Latent space size: 64.
- Data split (training batch-size | validation number of samples | test number of samples): $(32|1000|1000)$.
- Experiments are carried out on Intel(R) Xeon(R) Gold 6226 CPU @ 2.70GHz machines with NVIDIA Tesla A100 GPUs on the Phoenix cluster [16].
- Non-Convex solver: *MadNLP.jl* [24].
- Quadratic solver: *Gurobi* [25].
- Conic solver: *Mosek* [26].

For a broader view of the policy simulation under different models of reality, Figures J and J show the expected stored energy (volume) and thermal generators dispatch over time for each analysed policy in the Bolivian grid under the SOC and DCLL power flow formulation in the bellow figures.

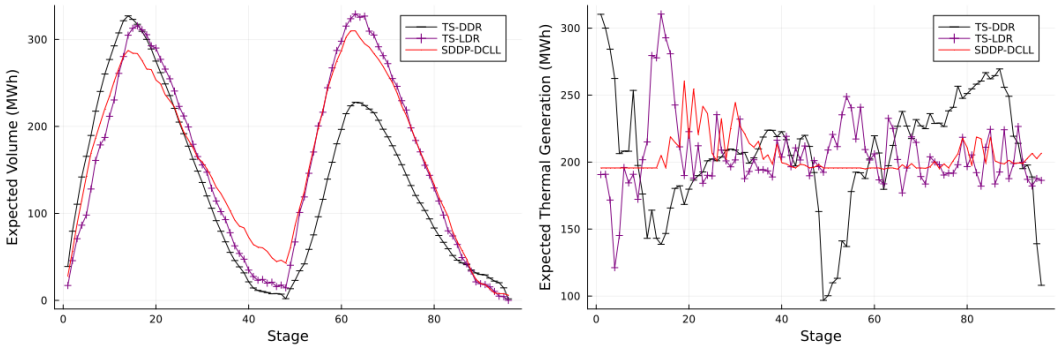

Figure 5: Expected stored energy and thermal dispatch over time for the DCLL formulation.

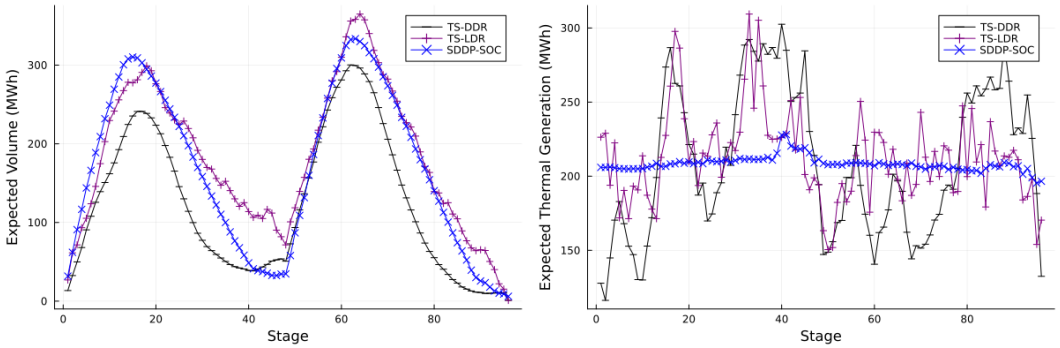

Figure 6: Expected stored energy and thermal dispatch over time for the SOC formulation.

K Results on a Small Test Case

This provides details and results for an additional small test case.

Policy and optimization details:

- All ML models are implemented in Julia using *Flux.jl* [23].
- Weights Optimizer: $Adam(\eta = 0.001, \beta :: Tuple = (0.9, 0.999), \epsilon = 1.0\ 10^{-8})$
- Latent space size: 16.
- Data split (training batch-size | validation number of samples | test number of samples): $(32|1000|1000)$.
- Experiments are carried out on Intel(R) Xeon(R) Gold 6226 CPU @ 2.70GHz machines with NVIDIA Tesla A100 GPUs on the Phoenix cluster [16].
- Non-Convex solver: *MadNLP.jl* [24].
- Quadratic solver: *Gurobi* [25].
- Conic solver: *Mosek* [26].

This case study utilizes a three-bus system to demonstrate the impacts of underlying policies. The system forms a loop by connecting all three buses, making it an illustrative example to observe the influence of Kirchhoff's Voltage Law (KVL) constraints on the quality of the approximations studied. The setup includes a hydroelectric unit at bus 1, a thermo-electric unit at bus 2, and the most costly thermo-electric unit along with the demand at bus 3. The planning horizon spans 48 periods, with each stage representing 730 hours (equivalent to one month). The case study employs three scenarios per stage (low, medium, and high), which sums to 3^{48} possible scenarios for the entire problem.

Table 5: Comparison of SDDP and ML Decision Rule for Case3 with DCLL Implementation.

Model	Plan	Imp Cost (USD)	GAP (%)	Training (Min)	Execution (Min)
SDDP	DCLL	47703	-	-	3
TS-LDR	DCLL	48937	2.59	5	0.00047
TS-DDR	DCLL	49449	3.66	5	0.00047

Table 6: Comparison of SDDP and ML Decision Rule for Case3 with SOC Implementation.

Model	Plan	Imp Cost (USD)	GAP (%)	Training (Min)	Execution (Min)
SDDP	SOC	49178	-	-	17
TS-DDR	SOC	50819	3.33	30	0.0046
TS-LDR	SOC	50926	3.56	90	0.0046

L RL Algorithms

The model-free RL algorithms explored are diverse in terms of their underlying approaches, which include both value-based and policy-based methods, as well as hybrid actor-critic approaches:

- **REINFORCE**: A foundational policy-gradient algorithm that directly optimizes the policy by maximizing expected returns through Monte Carlo estimates. REINFORCE updates the policy based on complete trajectories, providing unbiased estimates of the gradient, but can suffer from high variance in environments with complex, high-dimensional action spaces like AC-OPF.
- **Proximal Policy Optimization (PPO)**: A popular actor-critic method that restricts policy updates within a "trust region" by clipping the policy gradient, which stabilizes training. PPO's robustness and stability make it particularly suited for the high-dimensional feasible region in AC-OPF, where actions need to satisfy strict constraints.

Table 7: Comparison of SDDP and ML Decision Rule for Case3 with AC Implementation.

Model	Plan	Imp Cost (USD)	GAP (%)	Training (Min)	Execution (Min)
SDDP	DCLL	52330	-	-	5
SDDP	SOC	53705	2.62	-	5
TS-DDR	AC	53829	2.86	40	0.014
TS-LDR	AC	60493	15.59	150	0.014

- **Deep Deterministic Policy Gradient (DDPG)**: An off-policy actor-critic algorithm designed for continuous action spaces. DDPG uses a deterministic policy and leverages experience replay and target networks for stability. It is particularly suitable for continuous control problems like AC-OPF but may require significant tuning to handle the complex constraints.
- **Twin Delayed Deep Deterministic Policy Gradient (TD3)**: A variant of DDPG that reduces overestimation bias by using a pair of Q-networks and delaying policy updates. TD3 enhances stability in continuous control tasks and is expected to provide better convergence in the high-dimensional, constrained AC-OPF setting.
- **Soft Actor-Critic (SAC)**: An off-policy entropy-regularized actor-critic method that balances exploration and exploitation by optimizing for maximum entropy in addition to reward. SAC is robust in environments with complex dynamics, making it a strong candidate for navigating the feasible region defined by AC-OPF's constraints.