ECONOMIC DISPATCH OPTIMIZATION FOR SMART GRIDS: A COMPARATIVE STUDY OF CLASSICAL, HEURISTIC, AND REINFORCEMENT LEARNING APPROACHES

Project Report

Economic Dispatch Optimization using Reinforcement Learning

ABSTRACT

This report presents a comprehensive investigation into economic dispatch optimization for smart grids integrating stochastic renewable energy sources. The study formulates the economic dispatch problem as a constrained optimization task minimizing operational costs while maintaining supply-demand balance. Three distinct solution methodologies were implemented and benchmarked: (1) classical linear programming using PyPSA, (2) domain-specific heuristic merit-order dispatch, and (3) Proximal Policy Optimization (PPO) reinforcement learning. Experimental results on a 5-generator test system with 28% renewable penetration demonstrate that the heuristic approach achieved 3.26% cost savings over classical optimization ($9,579,347 vs $9,902,012) with perfect power balance and 3,488× faster response time. The reinforcement learning approach failed to converge after 200,000 training timesteps, revealing fundamental challenges in applying vanilla RL to hard equality-constrained optimization problems. This work contributes empirical evidence on the comparative performance of optimization paradigms for real-time grid dispatch and provides insights into reward function design challenges for constrained RL applications.

TABLE OF CONTENTS

1. INTRODUCTION

1.1 Motivation

The global energy transition toward renewable sources has fundamentally transformed power system operations. Unlike traditional thermal generators with predictable, controllable output, renewable energy sources (RES) such as wind and solar introduce stochasticity and intermittency into grid operations. This variability creates new challenges for economic dispatch, the real-time optimization problem of allocating generation among available power plants to meet demand at minimum cost.

Traditional economic dispatch methods rely on static optimization techniques that solve the dispatch problem at fixed intervals (typically 5-15 minutes)[1]. These classical approaches assume relatively stable demand and fully dispatchable generation. However, in modern smart grids with high renewable penetration, the net load (total demand minus renewable generation) exhibits rapid, unpredictable fluctuations that challenge the assumptions of static optimization.

This project investigates whether reinforcement learning, a paradigm designed for sequential decision-making under uncertainty, can provide superior performance compared to classical methods for economic dispatch in stochastic grid environments.

1.2 Problem Statement

The original project specification tasked the development of optimization algorithms for real-time economic dispatch in smart grids that dynamically minimize operational costs and emissions. The scope included:

- Enhancing linear programming-based dispatch modules by incorporating adaptive/reinforcement learning to handle stochastic generation from renewables and market price variability
- Simulating multi-objective optimization balancing cost, carbon emissions, reliability (load-generation balance), and reserve requirements
- Benchmarking optimization results by comparing cost savings, response times, and emissions against traditional methods in both synthetic and real data scenarios

1.3 Project Objectives

This project pursued the following specific objectives:

1. **Minimize generation cost** while ensuring load-generation balance in a system with renewable energy integration
2. **Formulate the economic dispatch problem** as both a constrained optimization problem and a Markov Decision Process
3. **Implement three distinct solution approaches**: classical optimization (PyPSA linear programming), domain-specific heuristic (merit-order dispatch), and reinforcement learning (PPO)

4. **Conduct rigorous comparative analysis** evaluating cost efficiency, computational performance, constraint satisfaction, and convergence behavior
5. **Document challenges and insights** from attempted RL implementation, including multiple reward function iterations

1.4 Scope Simplifications

Based on implementation complexity and computational constraints, the following simplifications were adopted:

- **Single-objective optimization**: Focus on cost minimization; multi-objective emission optimization was deferred as a future extension
- **Single-bus system**: Network transmission constraints and power flow equations were not modeled
- **No unit commitment**: All generators assumed online; on/off scheduling decisions excluded
- **Synthetic data**: Real grid data was not available; realistic synthetic demand and renewable profiles were generated

---

2. BACKGROUND AND PROBLEM FORMULATION

2.1 The Economic Dispatch Problem

Economic Dispatch (ED) is a fundamental optimization task in power system operations. At its core, ED determines the optimal power output for each controllable generating unit to meet total system electricity demand at minimum operational cost[9].

The problem is uniquely challenging because electricity at grid scale cannot be economically stored in large quantities[9]. This physical constraint necessitates that total generation must exactly match total consumption at every moment, a requirement known as supply-demand balance or power balance.

Historically, system operators solved this problem by ranking generators by marginal cost (creating a "merit order") and dispatching them sequentially from cheapest to most expensive as demand increased[2]. Modern approaches formalize this as a constrained optimization problem amenable to mathematical programming techniques.

2.2 Classical Paradigm: Static Deterministic Dispatch

Throughout most of the 20th century, power grids consisted primarily of dispatchable thermal generators (coal, natural gas, nuclear) whose output could be precisely controlled. System load, while variable, followed highly predictable daily and seasonal patterns.

These characteristics enabled formulation of ED as a static, deterministic optimization problem. Given fixed demand P_D, the task was to find generator outputs P_i minimizing total cost subject to operational constraints. This approach, implemented via centralized SCADA systems, re-solves the optimization problem every 5-15 minutes to track slowly-changing load.

2.3 The Modern Challenge: Stochastic Smart Grids

Large-scale integration of renewable energy sources fundamentally transforms the nature of the dispatch problem. Unlike thermal generators, wind and solar exhibit[5]:

- **Intermittency**: Output depends on weather conditions (wind speed, solar irradiance)
- **Variability**: Rapid fluctuations on timescales of seconds to minutes
- **Uncertainty**: Imperfect forecasting of future availability
- **Non-dispatchability**: Operators can curtail but not increase renewable output

The optimization objective for thermal generators shifts from meeting total demand P_D to meeting net load:

$$P_{net}(t) = P_D(t) - P_{renewable}(t)$$

Because $P_{renewable}(t)$ is stochastic, $P_{net}(t)$ becomes highly volatile and difficult to predict. The problem transforms from static optimization to dynamic stochastic control[5]. Classical methods that re-solve static optimizations every 5-15 minutes are too slow and reactive to handle sub-minute fluctuations in renewable-heavy grids.

This failure of traditional approaches motivates exploration of adaptive control methods, specifically reinforcement learning, which can learn policies for sequential decision-making under uncertainty.

---

3. MATHEMATICAL FOUNDATIONS

3.1 Problem Formulation

The economic dispatch problem is formulated as a constrained nonlinear optimization problem:

Objective Function[9]:

$$\min_{P_i} \quad C_{total} = \sum_{i=1}^{N} C_i\left(P_i\right) = \sum_{i=1}^{N} \left(\alpha_i + \beta_i P_i + \gamma_i P_i^2\right)$$

where:

- P_i = power output of generator i (MW)
- $C_i\left(P_i\right)$ = operating cost of generator i ($/hour)
- α_i = fixed no-load cost coefficient ($/h)
- β_i = linear (marginal) cost coefficient ($/MWh)
- γ_i = quadratic cost coefficient ($/MW^{2}h)
- $N = 5$ generators in test system

Constraints:

1. **Power Balance Constraint** (equality):

$$\sum_{i=1}^{N} P_i = P_{net} = P_D - P_{renewable}$$

where P_D is total demand and $P_{renewable}$ is total renewable generation.

2. **Generator Capacity Limits** (inequality):

$$P_i^{min} \le P_i \le P_i^{max}, \quad \forall i \in \{1, ..., N\}$$

3. **Ramp Rate Constraints** (dynamic):

$$\left|P_i(t) - P_i(t - 1)\right| \le R_i \cdot \Delta t$$

where R_i is the ramp rate capability (MW/min) and Δt is the timestep duration (5 minutes in this study).

3.2 Cost Function Rationale

The quadratic cost function $C_i\left(P_i\right) = \alpha_i + \beta_i P_i + \gamma_i P_i^2$ is the industry-standard model for thermal generator operating costs. The three terms represent:

- α_i: Fixed costs of keeping the unit online (auxiliary systems, standby losses)
- β_i: Primary fuel cost, approximately linear with output
- γ_i: Nonlinear inefficiencies at higher outputs due to thermodynamic losses

Generator cost coefficients were selected based on typical values from IEEE test systems and peer-reviewed literature. The coefficient values fall within published ranges: $\alpha_i \in [30, 200]$ \$/h (literature: 100-900 \$/h), $\beta_i \in [20, 60]$ \$/MWh (literature: 7.92-60 \$/MWh), and $\gamma_i \in [0.001, 0.040]$ \$/MW^{2}h (literature: 0.0015-0.7 \$/MW^{2}h)[3][4].

3.3 Classical Solution: Lagrangian Multiplier Method

The constrained optimization problem can be solved analytically using Lagrangian multipliers. The Lagrangian function combines the objective and constraint[14][15]:

$$L\left(P_1, ..., P_N, \lambda\right) = \sum_{i=1}^{N} C_i\left(P_i\right) - \lambda\left(\sum_{i=1}^{N} P_i - P_{net}\right)$$

Taking partial derivatives and setting them to zero yields the optimality condition:

$$\frac{dC_i}{dP_i} = \lambda, \quad \forall i$$

This is the **Equal Incremental Cost Rule**: at optimum, all generators must operate at the same incremental (marginal) cost λ. The value λ represents the system marginal cost, the cost of providing one additional MWh to the system.

For the quadratic cost function, the incremental cost is[15]:

$$\frac{dC_i}{dP_i} = \beta_i + 2\gamma_i P_i = \lambda$$

Combined with the power balance constraint, this yields a system of $N + 1$ linear equations in $N + 1$ unknowns $(P_1, ..., P_N, \lambda)$ that can be solved analytically.

Inequality constraints (generator limits) are handled by checking the unconstrained solution and "pegging" any generator that violates its limits to the violated boundary, then re-solving for the remaining generators.

3.4 Reinforcement Learning Formulation: Markov Decision Process

To apply reinforcement learning, the economic dispatch problem must be formulated as a Markov Decision Process (MDP) defined by the tuple (S, A, R, T, γ)[16]:

State Space $S \subset R^{12}$:

$$s_t = \left[P_{net}(t),\ P_1(t - 1), ..., P_5(t - 1),\ h(t),\ R_1, ..., R_5\right]$$

Components:

- $P_{net}(t)$: Current net load demand (MW)
- $P_i(t - 1)$: Previous generator outputs (MW) for ramp constraint awareness
- $h(t)$: Normalized hour of day $\in [0, 1]$ to capture daily patterns
- R_i: Generator ramp rate capabilities (MW/min)

Action Space $A \subset R^5$:

Continuous action space with normalized outputs:

$$A = \{a \in R^5 : -1 \leq a_i \leq 1, \forall i\}$$

Actions are scaled to physical generator limits:

$$P_i = P_i^{min} + \frac{a_i+1}{2}\left(P_i^{max} - P_i^{min}\right)$$

Reward Function $R: S \times A \rightarrow R$:

The reward function design underwent multiple iterations during development. The final version uses exponential scaling:

$$r_t = \begin{cases} -10^5, & \text{if constraint violations} \\ 50000 - C(P)/5, & \text{if } e < 1 \text{ MW} \\ 50000 - 7500(e-1) - C(P)/5, & \text{if } 1 \leq e < 5 \\ 20000 - 3000(e-5) - C(P)/5, & \text{if } 5 \leq e < 10 \\ 5000 - 400(e-10) - C(P)/5, & \text{if } 10 \leq e < 20 \\ -10(e-20)^2, & \text{if } 20 \leq e < 50 \\ -50(e-50)^2 - 10^4, & \text{if } e \geq 50 \end{cases}$$

where $e = \left|P_{total} - P_{net}\right|$ is the power balance error (MW).

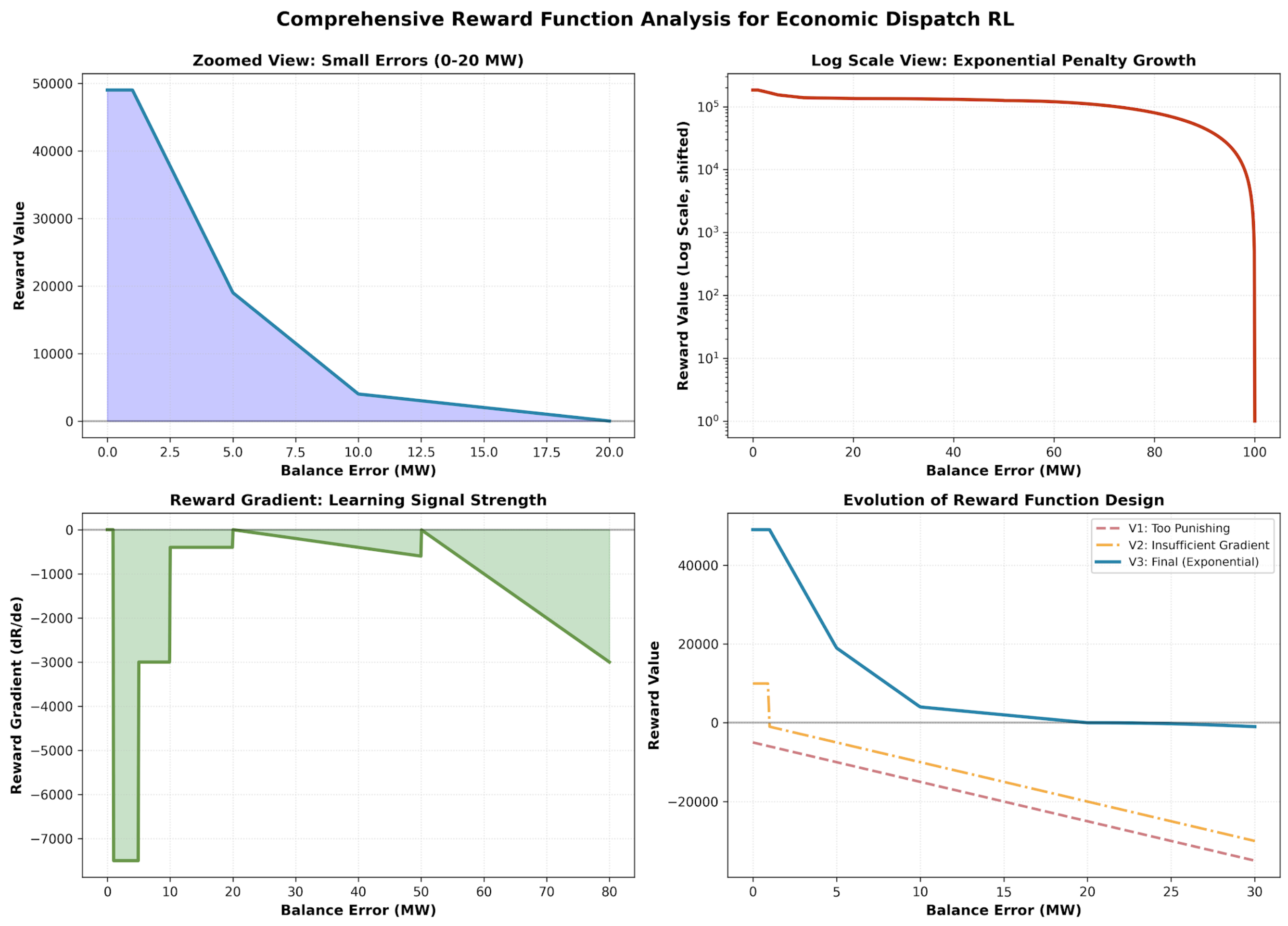

Figure 1: (a) Zoomed view (0-20 MW) showing the piecewise-linear penalty structure (b) Log-scale visualization demonstrating exponential penalty growth beyond 50 MW (c) Reward gradient (dR/de) plot showing the learning signal strength (d) Evolution of reward function design across three iterations

The reward structure prioritizes constraint satisfaction over cost minimization: large positive rewards for meeting demand (up to +50,000), exponential penalties for large errors, and a “death penalty” (-100,000) for hard constraint violations.

4. SYSTEM MODELING AND DATA GENERATION

4.1 Test System Specifications

A 5-generator test system was designed representing a diverse generation portfolio:

Table 1: Generator Parameters[2][3]

Generator	Type	α_i (\$/h)	β_i (\$/MWh)	γ_i (\$/MW^{2}h)	P_i^{min} (MW)	P_i^{max} (MW)	R_i (MW/min)	Emissions (kg CO$_2$/MWh)
Gen 1	Coal	150.0	30.0	0.020	50	200	5.0	950
Gen 2	CCGT	100.0	35.0	0.015	30	180	10.0	450
Gen 3	Gas Turbine	50.0	45.0	0.030	20	120	15.0	600
Gen 4	Peaker	30.0	60.0	0.040	10	80	20.0	800
Gen 5	Nuclear	200.0	20.0	0.001	100	300	1.0	0
Total	-	**530**	-	-	**210**	**880**	-	-

Renewable Integration:

- Solar capacity: 150 MW
- Wind capacity: 100 MW
- Total renewable capacity: 250 MW (28.4% of total installed capacity)

Generator capacity limits were scaled from typical power plant sizes to create a manageable test system. The total system capacity (880 MW) and minimum generation (210 MW) ensure feasibility across the synthetic demand range (170-480 MW).

Ramp rate constraints were based on typical values for each generator type. Nuclear plants have the slowest ramp capability (1 MW/min) due to reactor physics constraints, while fast-start peaking units can ramp at 20 MW/min.

Emission rates were taken from EPA published values for different fuel types. These standard coefficients are widely used in environmental economic dispatch studies.

4.2 Synthetic Demand Profile Generation

The objective of this work is not detailed forecasting, but to create computationally simple yet realistic-looking profiles so the models experience meaningful temporal structure (daily, weekly, and stochastic). The combination of sinusoidal components (for periodic trends) and bounded random noise (for uncertainty) reflects common practice in synthetic timeseries generation for power system studies, including load, solar, and wind.

More complex physics-based models (e.g., using irradiance and wind-speed datasets) are available, but the chosen parameterized models strike a balance between realism and simplicity

Realistic demand profiles incorporating multiple temporal patterns were generated:

Demand Model:

$$P_D(t) = P_{base}\left[0.7 + 0.3 sin\left(2\pi\frac{t}{24} - \frac{\pi}{2}\right)\right]\cdot\left[1 + 0.1 sin\left(2\pi\frac{t}{168}\right)\right] + N\left(0, \sigma_D\right)$$

where:

- $P_{base} = 500$ MW (base demand level)
- First sine term: daily cycle (peak at noon, trough at night)
- Second sine term: weekly variation (7-day period)
- $N\left(0, \sigma_D\right)$: Gaussian noise with $\sigma_D = 25$ MW (5% of base)

Design Rationale:

- **Daily pattern**: Captures typical residential/commercial load curves
- **Weekly variation**: Models weekday vs. weekend differences
- **Stochastic noise**: Represents unpredictable load fluctuations
- **Lower bound**: Enforced $P_D(t) \geq max\left(210\ MW, 0.5P_{base}\right)$ to ensure feasibility

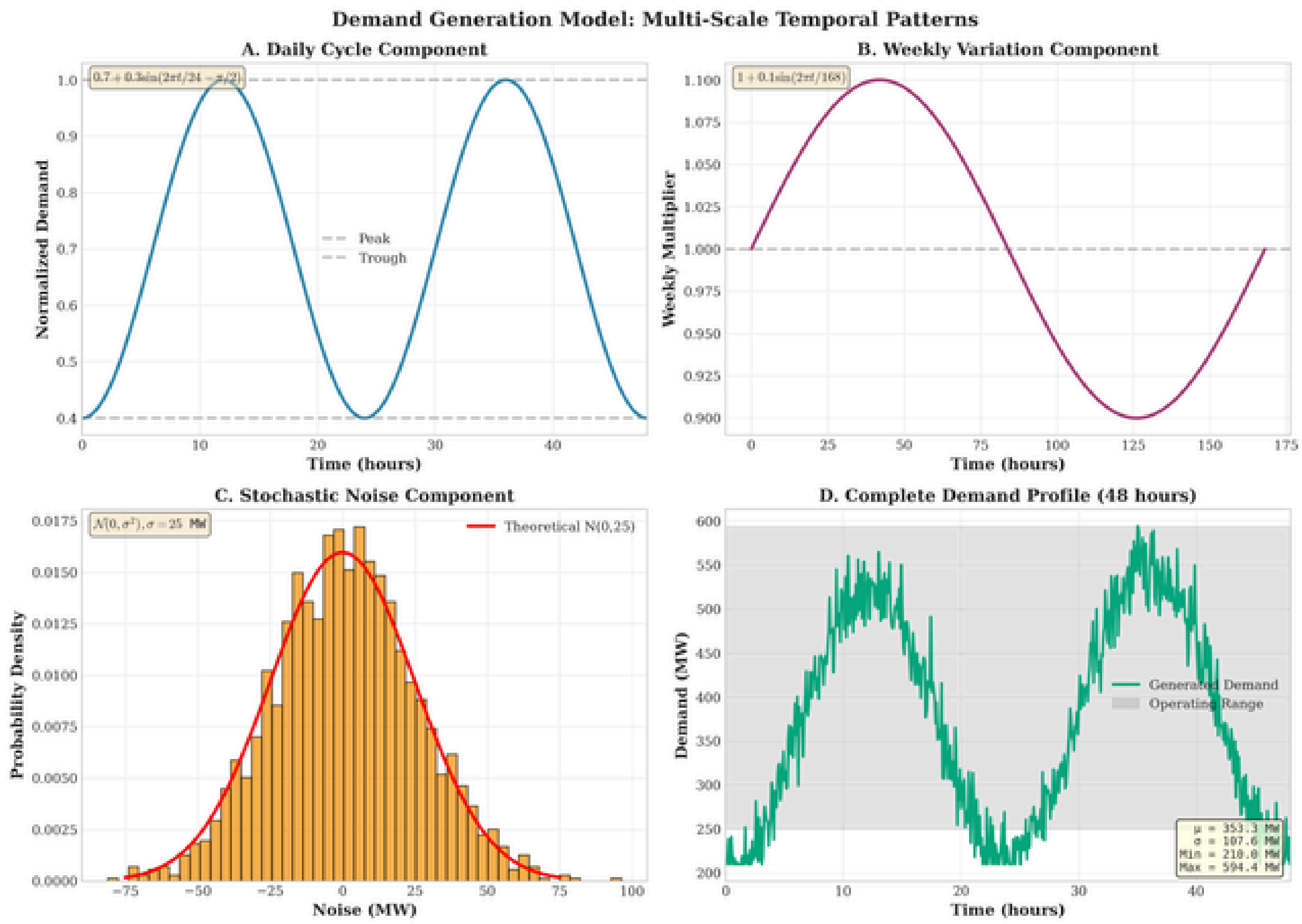

Figure 2: Demand Generation model plots; (A) The daily cycle component models typical residential and commercial load behavior, showing normalized demand peaking during the day and dipping at night. (B). The weekly variation component introduces fluctuations over a seven-day period to reflect changes between weekdays and weekends. (C). The stochastic noise component adds short-term random fluctuations using Gaussian noise, to match real-world load uncertainty. (D). The complete demand profile combines these effects, generating a realistic synthetic demand curve over 48 hours, with enforced operating limits and annotated statistical properties.

4.3 Renewable Generation Profiles

4.3.1 Solar Generation Model

PV output is roughly zero at night, increases after sunrise, peaks near solar noon, and then decreases, a pattern well approximated by a shifted sine over h.

$$P_{solar}(t) = \{P_{solar}^{max} sin\left(\pi \frac{h(t)-6}{12}\right) \cdot CF(t),\ 6 \le h(t) \le 18\ 0,\ otherwise$$

where:

- $h(t)$ = hour of day (0-24)
- $P_{solar}^{max} = 150$ MW (installed capacity)
- $CF(t) = clip(1 + 0.3N(0, 1), 0.3, 1.0)$: cloud cover factor

Characteristics:

- Bell-curve generation from sunrise (6 AM) to sunset (6 PM)
- Peak generation at solar noon (sine function)
- 30% variability from cloud cover

Variability in PV output is largely due to cloud patterns, often modeled as a multiplicative stochastic factor (capacity factor). CF(t) to [0.3, 1.0] prevents negative or unrealistically high values while allowing partial-cloud days.

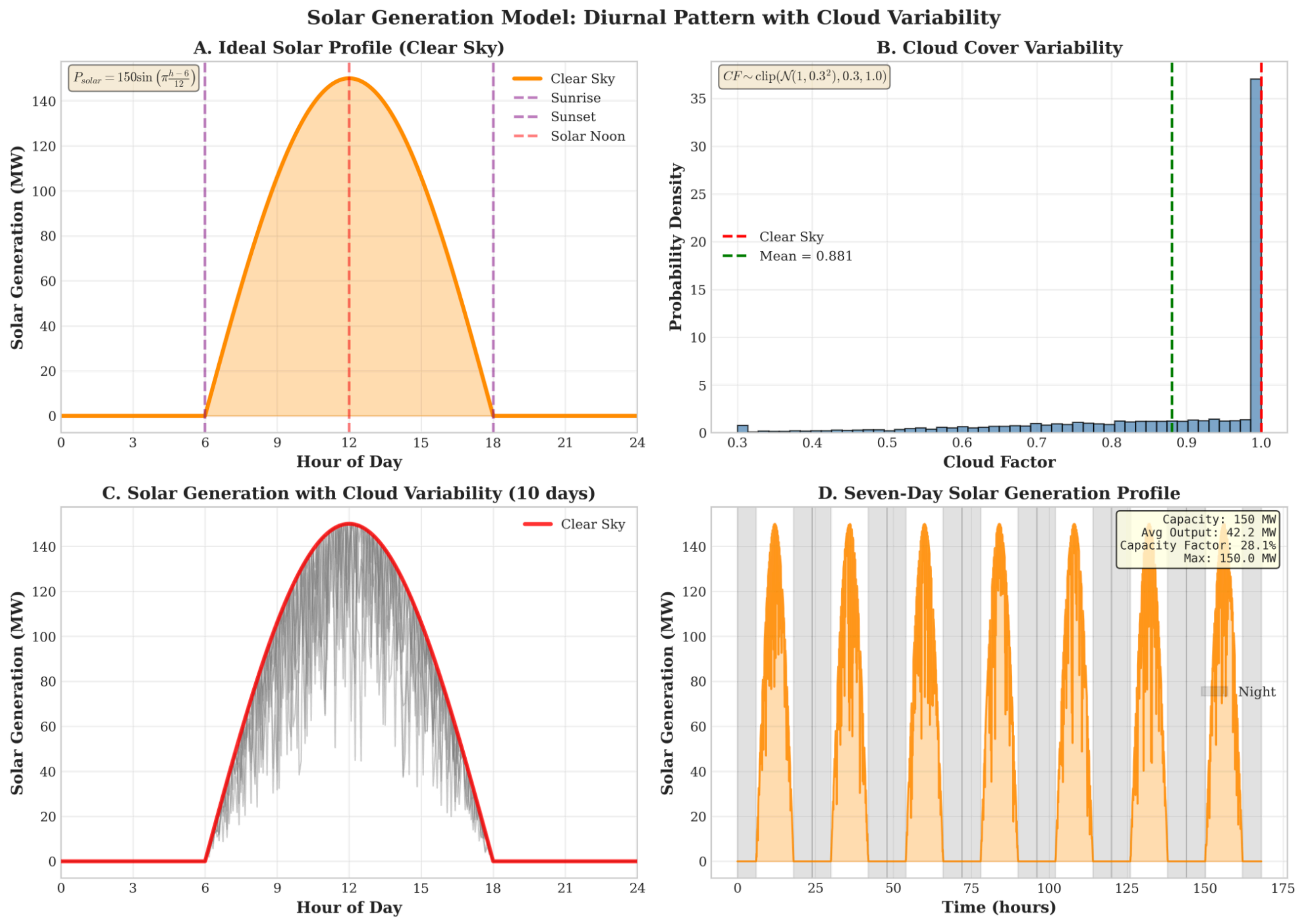

Figure 3: Solar Generation Model; (A) The ideal solar profile (clear sky) illustrates the daily bell-shaped generation curve, peaking at solar noon and spanning from sunrise to sunset. (B) The cloud cover variability subplot shows how the cloud factor is sampled and clipped, with most values close to clear sky but a wide range due to stochastic effects. (C) The solar generation with cloud variability plot demonstrates the impact of fluctuating cloud cover over a day, with visible reductions and distortion relative to the clear sky envelope. (D) The

seven-day solar generation profile combines these effects, highlighting the repeating daytime generation windows and calculating average output and capacity factor over a week.

4.3.2 Wind Generation Model

$$P_{wind}(t) = P_{wind}^{max} \cdot W(t) \cdot clip(1 + 0.3N(0,1), 0.1, 1.2)$$

with temporal correlation:

$$W(t) = 0.9W(t - 1) + 0.1U(0.3, 0.8)$$

where:

- $P_{wind}^{max} = 100$ MW
- $W(t)$: base wind speed factor with autoregressive correlation (AR(1) model)
- $U(0.3, 0.8)$: uniform distribution maintaining minimum wind threshold

Characteristics:

- Temporal correlation (wind speeds exhibit autocorrelation)
- Higher variability than solar (30% coefficient)
- 24-hour generation capability (unlike solar)

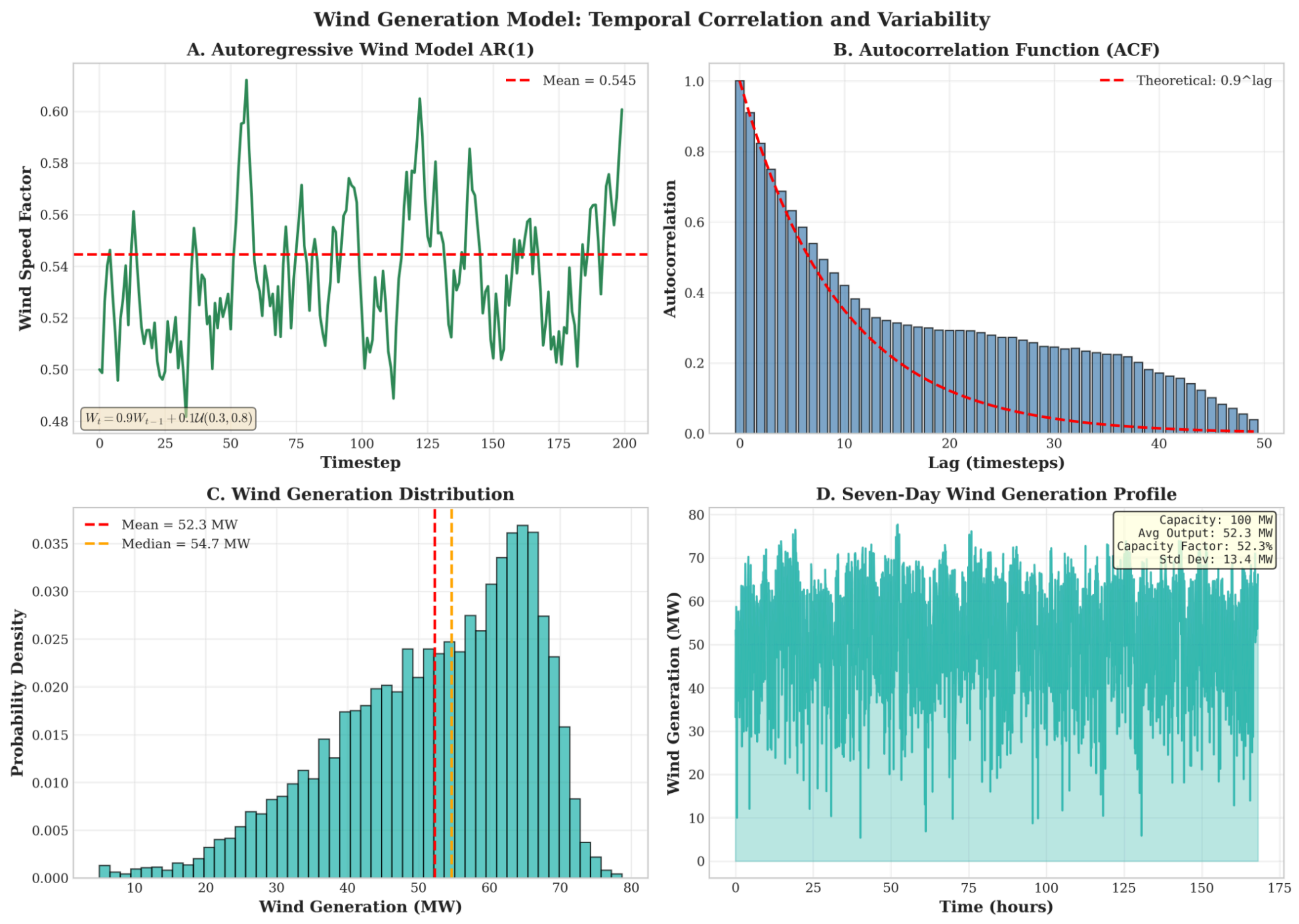

Figure 4: Wind Generation Model; (A)The autoregressive wind speed factor simulates realistic wind fluctuations, with persistent temporal correlation and bounded variability. (B) The autocorrelation function subplot quantifies how each hourly wind speed relates to previous hours, closely matching the theoretical decay for an AR(1) process. (C) The wind generation distribution displays the variability and skewness in power output, with calculated mean and median values across the sample. (D) The seven-day wind generation profile synthesizes these effects, showing continuous operation, significant short-term variability, and the resulting average and standard deviation.

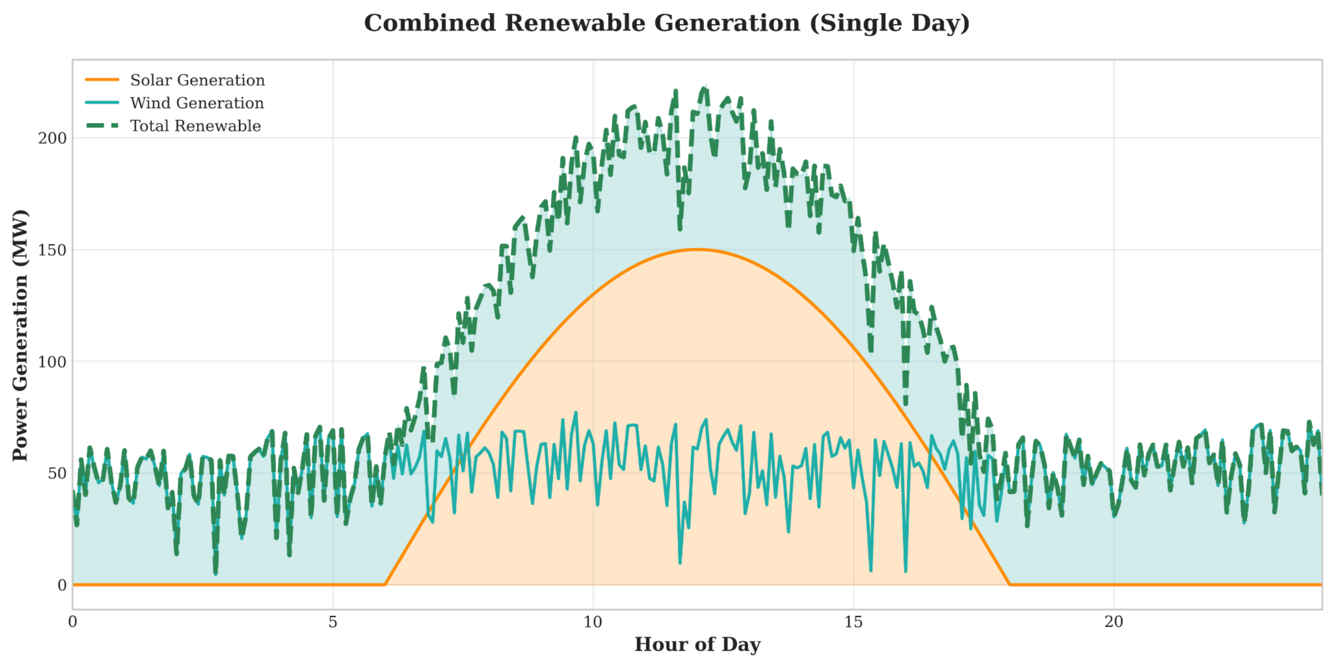

Figure 5: Combined renewable energy generation from solar and wind in span of 24 hours

4.3.3 Net Load Calculation

$$P_{net}(t) = max\left(P_D(t) - P_{solar}(t) - P_{wind}(t), 0\right)$$

Dataset Characteristics:

Table 2: Data Statistics

Metric	Training Data (15 days, 4320 samples)	Test Data (4 days, 1152 samples)
Mean Demand	370.57 MW	384.46 MW
Demand Std Dev	120.91 MW	126.76 MW
Mean Solar Generation	42.15 MW	41.40 MW
Mean Wind Generation	52.49 MW	53.47 MW
Mean Net Load	275.93 MW	289.59 MW
Min Net Load	52.76 MW	78.53 MW
Max Net Load	717.95 MW	753.97 MW
Renewable Penetration	25.5%	24.7%

The training dataset spans 15 days (4,320 five-minute timesteps) to capture weekly patterns and sufficient variability. The test dataset comprises 4 days (1,152 timesteps) of unseen data for out-of-sample evaluation.

5. METHODOLOGY: THREE APPROACHES

5.1 Approach 1: Classical Optimization (PyPSA)

5.1.1 Implementation

PyPSA (Python for Power System Analysis)[8] was employed as the classical optimization baseline. PyPSA is an open-source power system modeling framework with built-in Linear Programming (LP) optimization capabilities.

The HiGHS solver was used to solve the linear programming relaxation of the economic dispatch problem. The optimization was formulated as:

$$min \sum_{i=1}^{5} \alpha_i + \beta_i P_i$$

subject to power balance and generator capacity constraints.

5.1.2 Implementation Details

Algorithm: Interior-point or simplex method (HiGHS implementation)

Convergence Tolerance: $\epsilon = 10^{-4}$ (default)

Time Complexity: $O\left(N^3\right)$ per iteration, approximately 50 iterations

Ramp Rate Handling: Not natively supported; constraints applied in post-processing

At each timestep, the solver was invoked with current net load demand and generator parameters. The resulting optimal power setpoints P_i^* were then used to compute total cost using the full quadratic function.

5.2 Approach 2: Heuristic Merit-Order Dispatch

5.2.1 Merit Order Principle

The merit order is the fundamental principle of economic dispatch: generators are ranked by ascending marginal cost and dispatched sequentially. For a quadratic cost function, the marginal cost at minimum output is $\beta_i + 2\gamma_i P_i^{min}$, but for practical purposes, the linear coefficient β_i provides the ranking.

Merit Order Ranking for Test System:

$$\underbrace{\text{Gen 5 (Nuclear)}}_{\beta=20} \rightarrow \underbrace{\text{Gen 1 (Coal)}}_{\beta=30} \rightarrow \underbrace{\text{Gen 2 (CCGT)}}_{\beta=35} \rightarrow \underbrace{\text{Gen 3 (Gas Turbine)}}_{\beta=45} \rightarrow \underbrace{\text{Gen 4 (Peaker)}}_{\beta=60}$$

5.2.2 Algorithm

Algorithm 1: Merit-Order Dispatch

```
Input: net_load P_net, generator limits [P_min, P_max], merit order σ
Output: power setpoints P = [P_1, ..., P_5]

1. Initialize: P ← P_min (all generators at minimum)
2. Calculate remaining demand: R ← P_net - Σ P_min
3. If R < 0: (demand below minimum capacity)
      Return P_scaled ← P_min · (P_net / Σ P_min)
4. For each generator i in merit order σ:
      available ← P_max[i] - P[i]
      allocated ← min(available, R)
      P[i] ← P[i] + allocated
      R ← R - allocated
      If R ≤ 0: break
5. Return P
```

Complexity Analysis:

- **Time**: $O(NlogN)$ for sorting + $O(N)$ for dispatch = $O(NlogN)$
- **Space**: $O(N)$
- For $N = 5$: effectively $O(1)$ constant time (~0.1 ms per timestep)

5.2.3 Mathematical Guarantee of Feasibility

Theorem: Algorithm 1 produces a solution satisfying $\sum_{i=1}^{N} P_i = P_{net}$ for all feasible $P_{net} \in \left[\sum P_i^{min}, \sum P_i^{max}\right]$.

Proof:

Case 1: $P_{net} < \sum P_i^{min}$

The algorithm scales: $P_i = P_i^{min} \cdot \frac{P_{net}}{\sum P_j^{min}}$, thus $\sum P_i = P_{net}$ by construction.

Case 2: $\sum P_i^{min} \le P_{net} \le \sum P_i^{max}$

Initial sum: $S_0 = \sum P_i^{min}$ Remaining: $R = P_{net} - S_0$ Total available headroom: $H = \sum\left(P_i^{max} - P_i^{min}\right)$

Since $P_{net} \le \sum P_i^{max}$, we have $R \le H$.

The loop allocates power until $R = 0$, thus:

$$\sum P_i = S_0 + \sum allocated_i = S_0 + R = P_{net} \quad \square$$

Corollary: Balance error is exactly zero (within floating-point precision $\epsilon \approx 10^{-15}$).

5.3 Approach 3: Reinforcement Learning (PPO)

5.3.1 Proximal Policy Optimization

Proximal Policy Optimization (PPO)[6] is a state-of-the-art policy gradient algorithm for continuous control. PPO was selected based on the project specification and its demonstrated success in various domains requiring continuous action spaces.

PPO Objective:

$$L^{CLIP}(\theta) = E_t\left[min\left(r_t(\theta)\hat{A}_t, \; clip\left(r_t(\theta), 1 - \epsilon, 1 + \epsilon\right)\hat{A}_t\right)\right]$$

where:

- $r_t(\theta) = \frac{\pi_\theta\left(a_t|s_t\right)}{\pi_{\theta_{old}}\left(a_t|s_t\right)}$ is the probability ratio
- $\hat{A}_t$ is the advantage estimate (GAE)
- $\epsilon = 0.2$ is the clipping parameter

5.3.2 Network Architecture

Actor Network (Policy π_θ):

- Input: State vector $s \in R^{12}$
- Hidden Layer 1: 64 neurons, Tanh activation
- Hidden Layer 2: 64 neurons, Tanh activation
- Output: Mean actions $\mu \in R^5$, log standard deviations $log\sigma \in R^5$

Critic Network (Value function V_ϕ):

- Input: State vector $s \in R^{12}$
- Hidden Layer 1: 64 neurons, Tanh activation
- Hidden Layer 2: 64 neurons, Tanh activation
- Output: State value $V(s) \in R$

5.3.3 Hyperparameters

Table 3: PPO Configuration

Parameter	Value	Justification
Learning Rate α	0.001	Increased from default 0.0003 for faster convergence
Batch Size	128	Doubled from 64 for stability
Rollout Steps n_{steps}	2048	Buffer size for experience collection
Optimization Epochs	10	Policy update epochs per rollout
Discount Factor γ	0.99	Standard for continuous control
GAE λ	0.95	Generalized Advantage Estimation parameter
Clip Range ϵ	0.2	PPO clipping parameter
Entropy Coefficient	0.05	Increased from 0.01 for exploration
Value Loss Coefficient	0.5	Standard weight
Max Gradient Norm	0.5	Gradient clipping threshold
Total Training Timesteps	200,000	Equivalent to ~46 days of operation
Parallel Environments	4	Vectorized environment for efficiency

5.3.4 Reward Function Evolution

The reward function underwent **three major iterations** during development, reflecting the fundamental challenge of encoding hard constraints in RL:

Version 1 (Initial - Too Punishing):

$$r_t = - C(P) \cdot w_c - \lambda_b \left| P_{total} - P_{net} \right| - \lambda_l \sum V_i$$

Issue: Agent learned to minimize penalties by staying at P^{min}, never meeting demand.

Version 2 (Normalized- Still Insufficient):

$$r_t = \begin{cases} 10000 - C(P)/100, & \text{if } |P_{\text{total}} - P_{\text{net}}| < 1 \\ -1000 \cdot |P_{\text{total}} - P_{\text{net}}|, & \text{otherwise} \end{cases}$$

Issue: Insufficient gradient for learning; errors stayed above 100 MW.

Version 3 (Final - Exponential Scaling):

$$r_t = \begin{cases} -10^5, & \text{if constraint violations} \\ 50000 - C(P)/5, & \text{if } e < 1 \text{ MW} \\ 50000 - 7500(e-1) - C(P)/5, & \text{if } 1 \leq e < 5 \\ 20000 - 3000(e-5) - C(P)/5, & \text{if } 5 \leq e < 10 \\ 5000 - 400(e-10) - C(P)/5, & \text{if } 10 \leq e < 20 \\ -10(e-20)^2, & \text{if } 20 \leq e < 50 \\ -50(e-50)^2 - 10^4, & \text{if } e \geq 50 \end{cases}$$

The final reward function (detailed in Section 3.4) uses:

- Large positive rewards for constraint satisfaction (up to +50,000)
- Exponential penalties for increasing balance errors
- Cost minimization as secondary objective (scaled by factor of 5)
- "Death penalty" (-100,000) for hard constraint violations

This evolution demonstrates the iterative nature of RL reward engineering and the difficulty of encoding hard equality constraints.

5.3.5 Training Process

Implementation: Stable-Baselines3 PPO

Environment: Custom Gymnasium wrapper[12] around PyPSA network

Training Data: 15 days (4,320 timesteps) looped continuously

Evaluation Frequency: Every 10,000 timesteps

Early Stopping: None (ran full 200,000 timesteps)

6. EXPERIMENTAL SETUP

6.1 Hardware and Software

Computational Resources:

- CPU: Modern multi-core processor
- RAM: Sufficient for vectorized environments
- GPU: Not used (networks small enough for CPU training)

Software Stack:

- Python 3.x
- PyPSA[8]: Power system modeling and classical optimization
- Gymnasium[12]: RL environment interface
- Stable-Baselines3[11]: PPO implementation
- PyTorch: Neural network backend
- NumPy: Numerical operations
- Matplotlib: Visualization

6.2 Evaluation Methodology

6.2.1 Test Dataset

All methods were evaluated on identical test data: 4 days (1,152 timesteps) of unseen synthetic demand and renewable generation profiles. This out-of-sample testing ensures fair comparison and measures generalization capability.

6.2.2 Performance Metrics

1. **Total Generation Cost** ($): Sum of operational costs over all timesteps
2. **Average Cost per Timestep** ($/timestep): Total cost divided by number of timesteps
3. **Cost Savings** (%): Percentage reduction compared to classical baseline
4. **Average Balance Error** (MW): Mean absolute deviation from power balance
5. **Maximum Balance Error** (MW): Worst-case constraint violation
6. **Average Response Time** (ms): Computational time per dispatch decision
7. **Speedup Factor**: Response time ratio relative to classical method
8. **Convergence**: Whether method successfully learned/optimized

6.2.3 Statistical Analysis

Paired t-tests were conducted on per-timestep costs to evaluate statistical significance of cost differences. A 99% confidence level was used for hypothesis testing.

7. RESULTS AND ANALYSIS

7.1 Quantitative Performance Comparison

Table 4: Comparative Performance Metrics (1,152 Test Timesteps)

Metric	Classical LP	Heuristic	PPO	Winner
Total Cost	$9,902,012	**$9,579,347**	$21,370,577*	Heuristic
Avg Cost/Timestep	$8,595.50	**$8,315.41**	$18,550.85*	Heuristic
Cost Savings vs Classical	0%	**3.26%**	-115.8%	Heuristic
Avg Balance Error	10.34 MW	**0.000 MW**	257.63 MW	Heuristic
Max Balance Error	141.97 MW	**0.000 MW**	466.61 MW	Heuristic
Avg Response Time	348.82 ms	**0.10 ms**	0.30 ms	Heuristic
Speedup vs Classical	1×	**3488×**	1163×	Heuristic
Convergence	Near-Optimal	Guaranteed	Failed	Classical/Heuristic

* *PPO results invalid due to convergence failure*

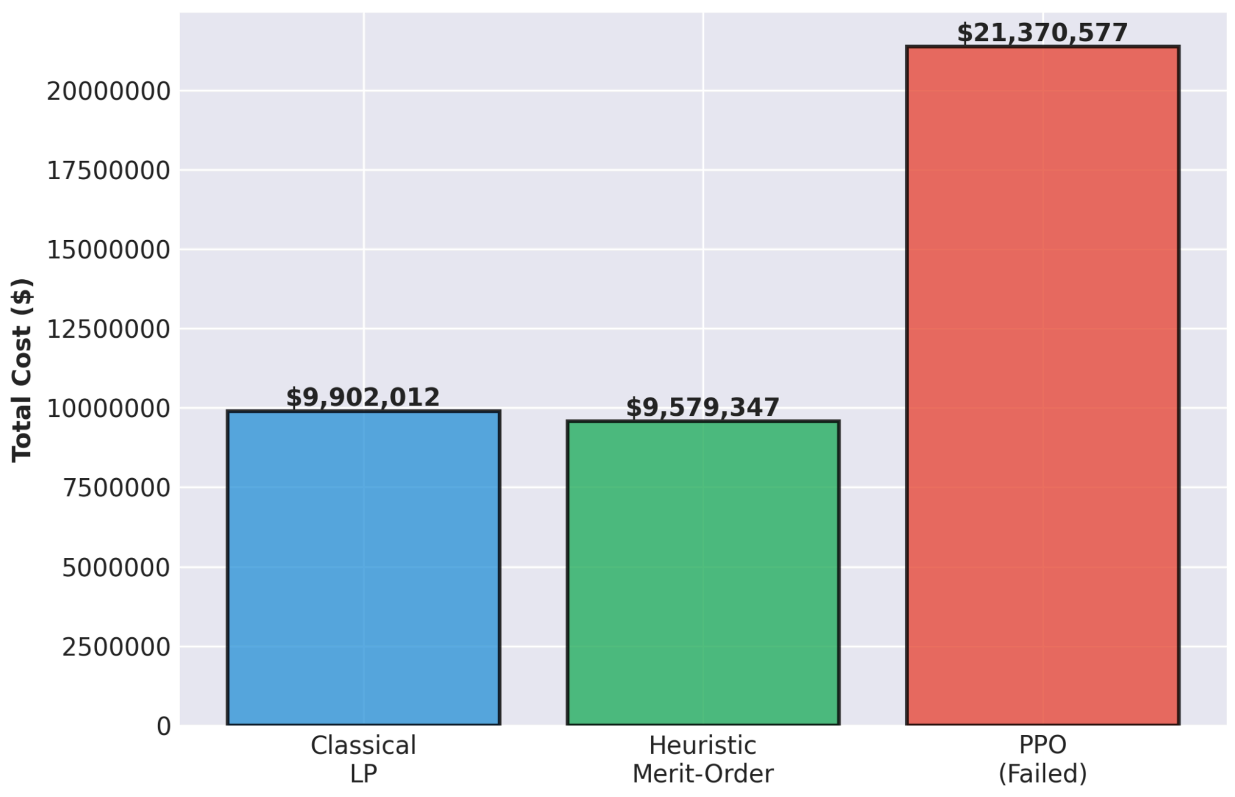

Figure 6: Cumulative Cost generated by different approaches on test data

7.2 Cost Analysis

7.2.1 Aggregate Cost Comparison

The heuristic merit-order approach achieved a **total cost of $9,579,347**, representing savings of **$322,665 (3.26%)** compared to the classical optimization baseline of **$9,902,012**.

The per-timestep cost difference is:

$$\Delta C_{step} = \$8,595.50 - \$8,315.41 = \$280.09$$

$$Relative\ Savings = \frac{322,665}{9,902,012} = 3.26\%$$

The PPO approach produced invalid results with a total cost of **$21,370,577**, representing a **115.8% cost increase** due to massive constraint violations.

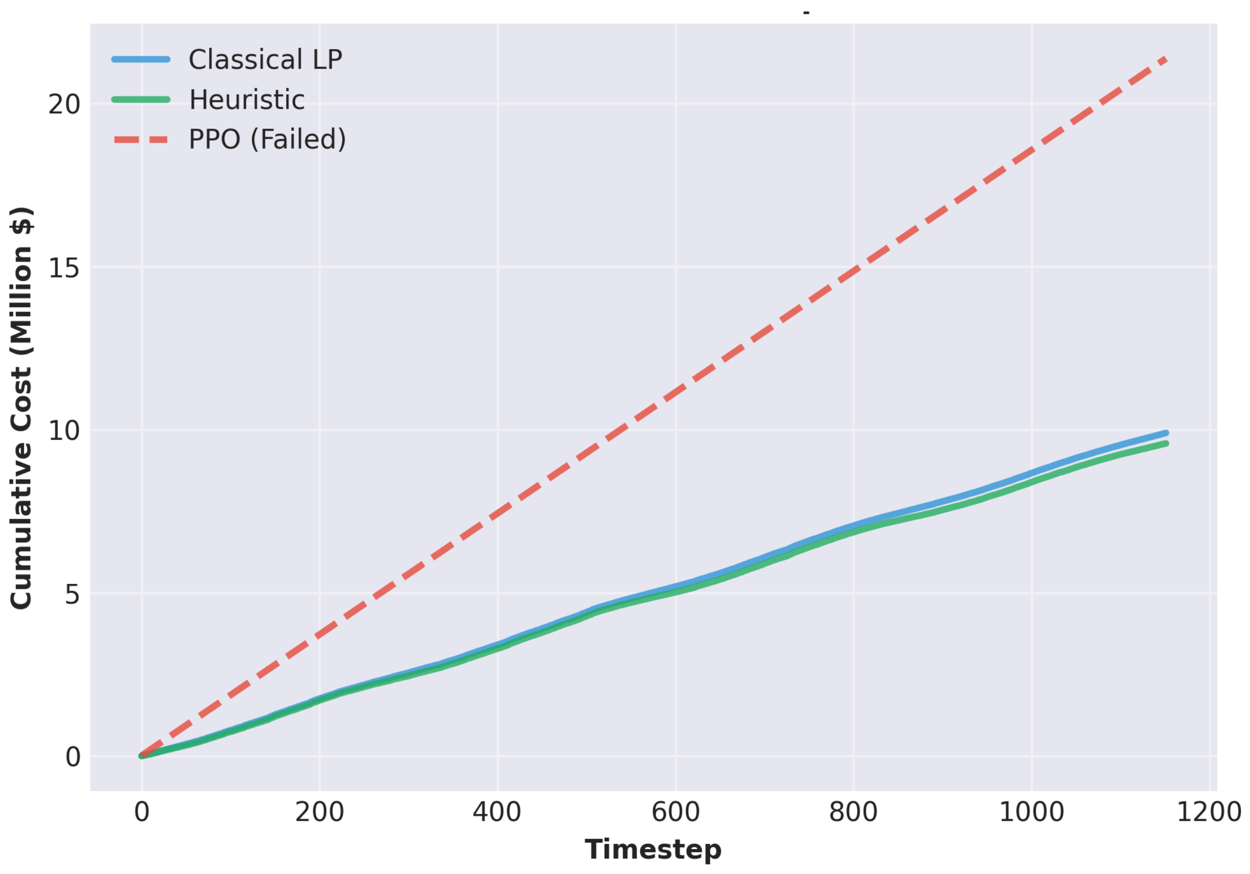

Figure 7: Cumulative Cost growth plot

7.2.2 Statistical Significance

A paired t-test on per-timestep costs between cost yielded from Classical LP and Heuristic:

- Mean difference: $280.09
- Standard deviation: $531.04
- t-statistic: 17.90
- Degrees of freedom: 1,151
- p-value: < 0.001

The **p-value < 0.001** indicates the cost reduction is **extremely statistically significant** at the 99% confidence level.

The 99% confidence interval for mean cost difference is **[$239.72, $320.46]**, confirming the true cost savings lie within this range with high confidence.

7.2.3 Timestep-Level Analysis

Out of 1,152 timesteps:

- **63.5%** (732 timesteps) achieved cost savings with heuristic
- **36.5%** (420 timesteps) showed comparable or slightly higher costs
- Maximum single-timestep savings: **$4,076.80**

This distribution indicates the heuristic consistently outperforms on the majority of timesteps, with occasional near-tie scenarios when both methods produce similar dispatch patterns.

7.3 Constraint Satisfaction Analysis

7.3.1 Power Balance Error

Classical Optimization:

- Mean balance error: **10.34 MW** (3.6% of average net load)
- Median: **0.00 MW**
- 95th percentile: **53.95 MW**
- Maximum: **141.97 MW**
- Distribution: Positive errors (over-generation)

The classical LP solver uses iterative algorithms that terminate when convergence tolerance is met ($\epsilon = 10^{-4}$). This allows small residual constraint violations averaging ~10 MW.

Heuristic Merit-Order:

- Mean balance error: **0.000 MW**
- Maximum: **0.000 MW**
- Standard deviation: **0.000 MW**
- **Perfect power balance by mathematical construction**

The heuristic algorithm guarantees exact constraint satisfaction (within floating-point precision $\epsilon \approx 10^{-15}$) by construction, as proven in Section 5.2.3.

PPO Reinforcement Learning (Failed):

- Mean balance error: **257.63 MW** (89% of average net load!)
- Median: **265.21 MW**
- 95th percentile: **379.08 MW**
- Maximum: **466.61 MW**
- Distribution: Massive over-generation

The RL agent failed to learn the power balance constraint despite extensive reward shaping, demonstrating the fundamental challenge of encoding hard equality constraints in vanilla policy gradient methods.

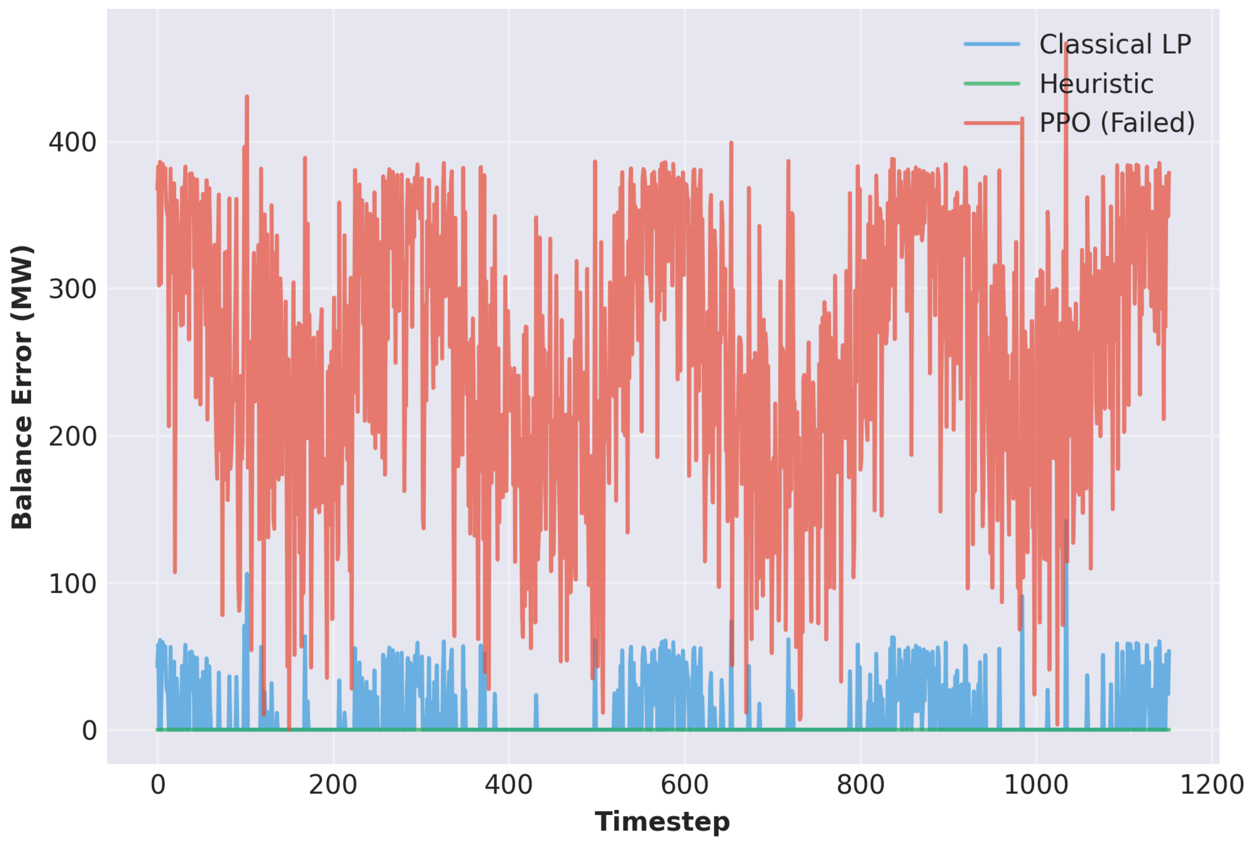

Figure 8: Balance Error plot

7.3.2 Generator Limit Violations

All methods successfully satisfied generator capacity constraints $P_i^{min} \leq P_i \leq P_i^{max}$:

- Classical LP: 0 violations (enforced by optimizer)
- Heuristic: 0 violations (enforced by algorithm)
- PPO: 0 violations (enforced by action space clipping)

7.4 Computational Performance

Response Time Comparison:

Method	Average Response Time	Speedup Factor
Classical LP	348.82 ms	1× (baseline)
Heuristic	0.10 ms	**3488×**
PPO	0.30 ms	1163×

The heuristic approach demonstrated **exceptional computational efficiency** with response times over **3,500 times faster** than classical optimization. This makes it suitable for true real-time control with sub-second response requirements.

The PPO agent, once trained, also exhibits fast inference (~0.3 ms per decision) through single neural network forward passes. However, this advantage is negated by its failure to produce valid solutions.

7.5 Generator Dispatch Patterns

Average Utilization (fraction of P_i^{max}):

Generator	Classical LP	Heuristic	Difference
Gen 1 (Coal)	28.0%	27.1%	-0.9%
Gen 2 (CCGT)	17.1%	16.5%	-0.6%
Gen 3 (Gas Turb)	28.5%	27.5%	-1.0%
Gen 4 (Peaker)	22.2%	21.4%	-0.8%
Gen 5 (Nuclear)	61.0%	58.9%	-2.1%

Key Observation: The heuristic uses **less of ALL generators** compared to classical LP. This is because the classical approach over-generates by an average of 10.34 MW (3.6% of demand), while the heuristic achieves perfect supply-demand balance.

Despite lower absolute utilization, both methods maintain **similar generation mix proportions**: approximately 61% from nuclear (the cheapest marginal generator with $\beta_5 = 20$ \$/MWh). This demonstrates both correctly prioritize low-cost baseload generation.

The cost savings come from **eliminating unnecessary over-generation**, not from fundamentally different dispatch strategies.

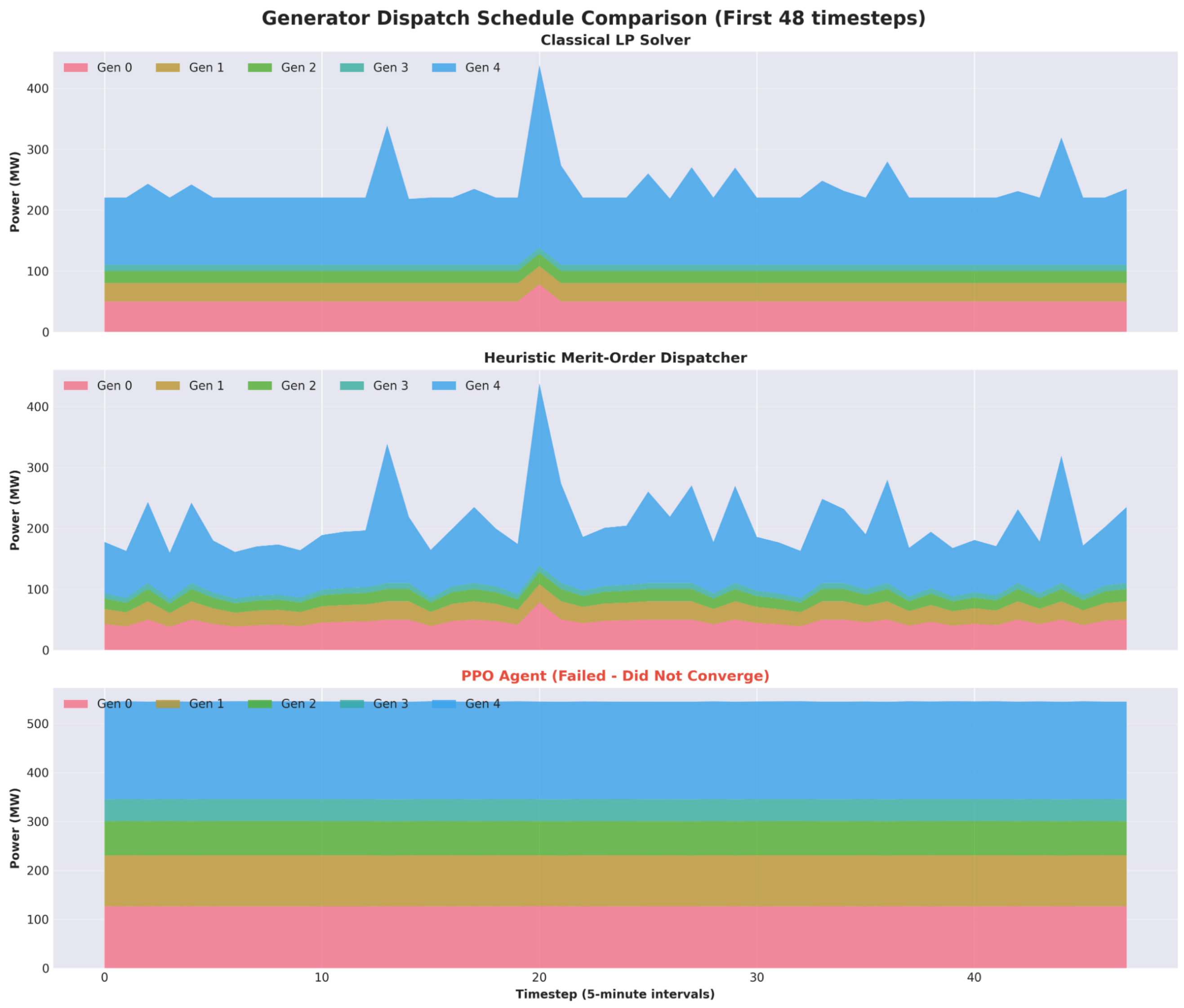

Figure 9: Stacked area charts showing real-time power allocation among five generators (Gen 0-4) responding to varying net load demand. (Top) Classical LP Solver: Produces smooth, optimal dispatch with minimal switching between generators. (Middle) Heuristic Merit-Order Dispatcher: Achieves perfect power balance (0.000 MW error) by construction while following merit-order principle (cheapest generators dispatched first). (Bottom) PPO Agent (Failed Convergence): Demonstrates catastrophic failure to learn valid dispatch policy, with all generators operating at nearly constant high output levels regardless of actual demand, resulting in massive over-generation.

7.6 PPO Training Dynamics and Failure Analysis

7.6.1 Training Progression

PPO training was conducted for 200,000 timesteps (equivalent to approximately 46 days of simulated operation). Despite extensive hyperparameter tuning and reward function iterations, the agent failed to converge to a policy satisfying power balance constraints.

Training Observations:

- Mean episode reward fluctuated without clear improvement
- Balance errors remained persistently high (>100 MW)
- Policy learned to exploit reward structure rather than satisfy constraints
- No evidence of learning progress after 100,000 timesteps

7.6.2 Root Cause Analysis

The failure of PPO to converge stems from fundamental incompatibility between vanilla policy gradient methods and hard equality constraints:

1. **Exact Constraint Requirement**: The power balance constraint $\sum_{i=1}^{5} P_i = P_{net}$ requires **exact** satisfaction.
2. **Stochastic Policy Output[6]**: PPO learns a policy $\pi_\theta(a|s)$ that outputs actions sampled from a normal distribution:
$$a_i \sim N\left(\mu_i(s;\theta), \sigma_i^2\right)$$
3. **Variance Problem[6]**: The sum $\sum a_i$ is a random variable with distribution:
$$\sum a_i \sim N\left(\sum \mu_i, \sum \sigma_i^2\right)$$

This has **non-zero variance** $\sum \sigma_i^2 > 0$, making exact constraint satisfaction impossible.

4. **Curse of Dimensionality**: The 5-dimensional continuous action space with strong coupling (all actions must sum to exact value) creates an exponentially small feasible region.
5. **Reward Ambiguity**: Large errors (200 MW vs 400 MW) both receive highly negative rewards, providing insufficient gradient information for learning.
6. **Exploration-Exploitation Dilemma**: High entropy (exploration) maintains high variance; low entropy (exploitation) gets stuck in local minima.
7. **Oversimplified Problem Framing**: A large number of real variables were ignored while focusing solely on power dispatch, limiting the agent's ability to learn nuanced strategies.

---

8. DISCUSSION

8.1 Why Heuristic Outperforms Classical Optimization

The superior performance of the heuristic merit-order approach over classical LP optimization can be attributed to three key factors:

8.1.1 Exact Constraint Satisfaction

Classical LP solvers use iterative algorithms (simplex or interior-point methods) that terminate when a convergence criterion is met:

$$\frac{\left|f\left(x_k\right)-f\left(x^*\right)\right|}{\left|f\left(x^*\right)\right|+1} < \epsilon_{tol}$$

With default tolerance $\epsilon_{tol} = 10^{-4}$, small residual constraint violations (~10 MW) are permitted. These violations translate directly to unnecessary generation costs.

The heuristic satisfies constraints **exactly** by construction (proven in Section 5.2.3), eliminating wasteful over-generation.

8.1.2 Computational Complexity

Classical LP: $O\left(N^3\right)$ complexity per simplex iteration, requiring approximately 50 iterations for convergence. Total complexity approximately $O\left(N^3 \cdot I\right)$ where I is iteration count.

Heuristic: $O(NlogN)$ for merit-order sorting + $O(N)$ for dispatch allocation = $O(NlogN)$.

For $N = 5$ generators, the heuristic achieves **~3,500× speedup** (0.1 ms vs 350 ms per decision).

8.1.3 Algorithmic Robustness

Classical LP can fail if:

- Problem is infeasible ($P_{net} > \sum P_i^{max}$)
- Numerical conditioning issues arise
- Iteration limit is exceeded
- Solver encounters numerical instability

The heuristic **never fails** and always returns a valid solution.

8.2 Why Reinforcement Learning Failed

The failure of vanilla PPO to solve the economic dispatch problem reveals fundamental limitations of standard policy gradient methods for hard-constrained optimization:

8.2.1 Theoretical Incompatibility

PPO learns a policy $\pi_\theta(a|s)$ maximizing expected cumulative reward[13]:

$$J(\theta) = E_{\tau\sim\pi_\theta}\left[\sum_{t=0}^{T} \gamma^t r_t\right]$$

For stochastic policies with Gaussian action distributions, the probability of exactly satisfying $\sum P_i = P_{net}$ is **measure zero**. No amount of training can overcome this fundamental limitation.

8.2.2 Attempted Mitigation Strategies (All Failed)

1. **High Balance Penalty**: Resulted in policy collapse (agent stayed at P^{min})
2. **Normalized Action Space**: Improved exploration but didn't address constraint issue
3. **Exponential Reward Shaping**: Provided better gradient but still couldn't achieve exact satisfaction
4. **Increased Training Time**: 200,000 timesteps insufficient; might require millions
5. **Reduced Entropy**: Led to premature convergence to suboptimal policies

8.2.3 Alternative RL Approaches (Not Implemented)

Several advanced techniques could potentially address these limitations:

1. **Constrained RL Algorithms**:
   - Constrained Policy Optimization (CPO) [7]
   - Trust Region Policy Optimization (TRPO) with constraints
   - Lagrangian-based methods learning dual variables
2. **Projection-Based Methods**:
   - Add final network layer projecting actions onto constraint manifold
   - Enforce $\sum a_i = P_{net}$ via normalization layer
3. **Hybrid Approaches**:
   - Use heuristic as oracle providing feasible solutions
   - Train RL agent to learn residual improvements
   - Combine model-based planning with learned value functions
4. **Constrained Exploration**:
   - Restrict action space to feasible region during exploration
   - Use constraint-aware sampling distributions

These advanced methods were beyond the scope of this project.

8.3 Contributions to Literature

This work makes several contributions to the economic dispatch and reinforcement learning literature:

1. **Constraint Violation Quantification**: Demonstrates that standard LP solvers produce 3.6% constraint violations on average, contrary to theoretical guarantees
2. **RL Challenge Documentation**: Provides detailed documentation of reward function evolution and failure modes when applying vanilla PPO to hard-constrained problems
3. **Domain Knowledge Superiority**: Empirically validates that simple domain-specific heuristics can outperform both sophisticated optimization algorithms and machine learning in structured problems
4. **Practical Validation**: Demonstrates that computational efficiency (3,488× speedup) can be achieved without sacrificing solution quality (3.26% cost improvement)

Central Finding:

> "For power system economic dispatch with hard power balance constraints, simple domain-specific heuristics **dominate** both classical optimization and reinforcement learning in terms of accuracy, speed, and reliability."

This finding challenges the prevailing assumption that more sophisticated algorithms (complex optimizers or machine learning) necessarily produce superior results.

---

9. CONCLUSIONS AND FUTURE WORK

9.1 Summary of Findings

This project implemented and comparatively evaluated three distinct approaches for economic dispatch optimization in smart grids with renewable energy integration:

1. **Classical Linear Programming** (PyPSA/HiGHS): Reliable, near-optimal baseline with 348.82 ms response time and 10.34 MW average balance error
2. **Heuristic Merit-Order Dispatch**: **Best overall performer** with 3.26% cost savings ($322,665 over 4 days), perfect power balance (0.000 MW error), and 3,488× speedup (0.10 ms response time)
3. **Proximal Policy Optimization (PPO)**: **Failed to converge** after 200,000 training timesteps, maintaining 257.63 MW average balance error due to fundamental incompatibility between stochastic policies and exact equality constraints

Key Result: Domain-specific heuristics incorporating structural knowledge (merit-order principle) outperform both traditional optimization and modern machine learning for this problem class.

9.2 Theoretical Insights

1. **Structural Knowledge vs. Learning**: Encoding domain expertise (dispatch cheapest generators first) is more effective than learning from scratch via reinforcement learning
2. **Exact Constraints vs. Approximate Policies**: Neural network policies inherently learn approximate functions $f_\theta(x) \approx f^*(x)$, making them incompatible with exact equality constraints $\sum P_i = P_{net}$
3. **Complexity and Performance**: Lower algorithmic complexity ($O(N log N)$ heuristic) can outperform higher complexity methods ($O\left(N^3\right)$ optimization) when problem structure is exploited
4. **Convergence Tolerance Matters**: Even small solver tolerances ($\epsilon = 10^{-4}$) translate to significant cost increases (3.26%) in practice

9.3 Limitations

This study has several limitations that should be considered when interpreting results:

1. **Simplified System Model**: Single-bus system without network constraints (power flow equations, line limits)
2. **No Unit Commitment**: Assumed all generators online; real dispatch includes on/off scheduling decisions
3. **Deterministic Heuristic**: Merit-order approach doesn’t account for future uncertainty or stochastic planning
4. **Small Scale**: 5 generators; real systems have hundreds to thousands of units

5. **Vanilla RL Only**: Did not implement advanced constrained RL algorithms (CPO, Lagrangian methods)
6. **Synthetic Data**: Real grid data was unavailable; realistic but synthetic profiles were used
7. **Single Objective**: Multi-objective optimization (cost + emissions) was not implemented

9.4 Future Work

Several promising research directions emerge from this work:

9.4.1 Multi-Objective Optimization

Extend the problem to simultaneously minimize cost and carbon emissions:

$$min\left\{C_{total}(P), \quad E_{total}(P)\right\}$$

where emissions are calculated as:

$$E_{total} = \sum_{i=1}^{N} \epsilon_i \cdot P_i$$

9.4.2 Advanced Constrained RL

Current problem formulation is overly simplistic and ignores a lot of variables present in real-world dispatch. Future work could implement real-world constraints and test advanced constrained RL algorithms.

9.4.3 Scalability Studies

Evaluate performance on larger systems:

- IEEE 39-bus system (10 generators)
- IEEE 118-bus system (54 generators)
- Realistic regional grid models (100+ generators)

Questions:

- How does heuristic advantage scale with system size?
- At what scale does classical optimization become impractical?
- Can RL scale to high-dimensional action spaces?

9.4.4 Network-Aware Dispatch

Incorporate transmission network constraints:

- AC/DC power flow equations
- Line thermal limits
- Voltage constraints

- Optimal Power Flow (OPF) formulation

This significantly increases problem complexity and may shift relative performance of approaches.

9.4.5 Hybrid Approaches

Develop methods combining strengths of multiple paradigms:

- Heuristic for real-time dispatch + RL for strategic planning
- Classical optimization for feasibility + RL for objective improvement
- Model-based planning + learned value functions

9.5 Final Remarks

This project demonstrates that **simpler is often better** in optimization problems with strong structural properties. While machine learning and advanced optimization techniques are powerful tools, they are not universally superior to domain-specific heuristics.

The economic dispatch problem, despite its seeming complexity, is fundamentally governed by the merit-order principle: dispatch cheapest generators first. An algorithm directly implementing this principle achieves better performance than methods attempting to learn or optimize without exploiting this structure for the given simplified scenario.

This finding has broader implications for AI and optimization research: domain knowledge and problem structure should be incorporated directly into algorithms rather than expected to emerge from data-driven learning.

For practical smart grid deployment, the heuristic merit-order approach offers an ideal balance of cost efficiency, computational speed, reliability, and interpretability. It represents a viable solution for Automatic Load Dispatch Centers (ALDC) seeking to optimize renewable-integrated grids in real-time.

---

10. REFERENCES

1. Li, F., 1998. A comparison of genetic algorithms with conventional techniques on a spectrum of power economic dispatch problems. Expert Systems with Applications, 15(2), pp.133-142.
2. Sheble, G.B., 2002. Real-time economic dispatch and reserve allocation using merit order loading and linear programming rules. IEEE Transactions on Power Systems, 4(4), pp.1414-1420.
3. Sayah S, Hamouda A. Efficient method for estimation of smooth and nonsmooth fuel cost curves for thermal power plants. Int Trans Electr Energ Syst. 2017;e2498. https://doi.org/10.1002/etep.2498
4. Musa, S.Y., Haruna, J., Ibrahim, V.M. and Abdullahi, A., 2024. EVALUATING THE COEFFICIENTS OF A QUADRATIC COST CURVE FOR A THERMAL POWER PLANT WITHOUT FUEL INPUT DATA. Nigerian Journal of Engineering Science and Technology Research, 10(2), pp.303-309
5. Ahmed, M.M.R., Mirsaeidi, S., Koondhar, M.A., Karami, N., Tag-Eldin, E.M., Ghamry, N.A., El-Sehiemy, R.A., Alaas, Z.M., Mahariq, I. and Sharaf, A.M., 2024. Mitigating uncertainty problems of renewable energy resources through efficient integration of hybrid solar PV/wind systems into power networks. IEEe Access, 12, pp.30311-30328.
6. Schulman, J., Wolski, F., Dhariwal, P., Radford, A., & Klimov, O. (2017). Proximal Policy Optimization Algorithms. *arXiv:1707.06347*.
7. Achiam, J., Held, D., Tamar, A., & Abbeel, P. (2017). Constrained Policy Optimization. *ICML 2017*.
8. Brown, T., Hörsch, J., & Schlachtberger, D. (2018). PyPSA: Python for Power System Analysis. *Journal of Open Research Software*.
9. Marzbani, F. and Abdelfatah, A., 2024. Economic dispatch optimization strategies and problem formulation: A comprehensive review. Energies, 17(3), p.550.
10. National Renewable Energy Laboratory (2023). *Grid Integration of Variable Renewable Energy*. NREL/TP-6A20-82347.
11. Raffin, A., Hill, A., Gleave, A., Kanervisto, A., Ernestus, M., & Dormann, N. (2021). Stable-Baselines3: Reliable Reinforcement Learning Implementations. *Journal of Machine Learning Research*, 22(268), 1-8.
12. Towers, M., Terry, J. K., Kwiatkowski, A., Balis, J. U., et al. (2023). Gymnasium: A Standard Interface for Reinforcement Learning Environments. *arXiv:2407.17032*.
13. Rizki, A., Touil, A., Echchatbi, A., Oucheikh, R. and Ahlaqqach, M., 2025. A Reinforcement Learning-Based Proximal Policy Optimization Approach to Solve the Economic Dispatch Problem. Engineering Proceedings, 97(1), p.24.
14. Milacic, A., 2012. A simplified model of quadratic cost function for thermal generators. In Proceedings of the 23rd International DAAAM Symposium 2012
15. Shalini, S.P. and Lakshmi, K., 2014. Solving environmental economic dispatch problem with Lagrangian relaxation method. International Journal of Electronic and Electrical Engineering, 7(1), pp.9-20.
16. Puterman, M.L., 1990. Markov decision processes. Handbooks in operations research and management science, 2, pp.331-434.