LLM-driven Hyperparameter Optimization Framework for Capacity Expansion Planning Models

Andres Lopez[1], Cody J. Newlun[2], Dilip Pandit[1], Tim Wilcox[1], Atri Bera[1], Tu Nguyen[1]
[1]Energy Storage Technology & Systems Department
[2]Electric Power Systems Research Department
Sandia National Laboratories, Albuquerque, NM, USA

***Abstract*—Large Language Model (LLM) optimizers offer a promising alternative to traditional numerical optimization by leveraging semantic reasoning capabilities to handle complex optimization problems that are difficult to express and quantify numerically. While autonomous research agents with LLM optimizers have rapidly emerged in fields such as drug discovery and materials science, their application to power systems and energy storage sectors remains largely unexplored. This paper presents a verification study that integrates an LLM optimizer for configuration-level optimization within capacity expansion planning (CEP) tools rather than direct asset siting or sizing decisions. The proposed approach employs a Creator-Executor iterative framework where the LLM performs semantic enrichment on model execution results and guides configuration changes to optimize system-level objectives such as total cost, emissions, or reliability metrics. By replacing traditional optimization functions with LLM-based reasoning, this work demonstrates the potential for autonomous research agents to effectively guide infrastructure investment decisions in the power systems domain. Results on an open-source CEP tool, QuESt Planning, provide evidence for the efficacy of LLM optimizers in energy storage analytics and capacity expansion planning.**

***Index Terms*—Autonomous agents, capacity expansion planning, energy storage systems, guided optimization, LLM optimizers, semantic enrichment**

I. Introduction

Large Language Model (LLM) optimizers have emerged as a promising alternative that leverages the semantic reasoning capabilities of modern language models to handle optimization problems that are difficult to express numerically. Although autonomous research agents with LLM optimizers have gained rapid traction in fields such as drug discovery, materials science, and computational biology [1]–[3], their application to the power systems and energy storage sectors remains largely unexplored, with existing work focusing primarily on foundational applications. Current approaches in power systems extend context through embeddings, retrieval augmented generation (RAG), or specialized prompting techniques to enhance semantic searching abilities by developing domain-specific models for power systems research [4], [5]. Other works focus on embedding the LLM itself in-the-loop, assisting operators with the decision making process, and allowing simple decisions to be made with some degree of human oversight [6]. This represents a significant opportunity to advance optimization methodologies in the energy domain, where complex planning decisions often involve nuanced trade-offs between cost, reliability, environmental impact, and operational constraints.

Power systems planning, particularly capacity expansion planning (CEP), involves intricate decision-making processes that sometimes benefit from semantic reasoning over purely numerical optimization. The QuESt Planning tool [7] provides an ideal testbed for verifying the efficacy of LLM-based optimization methods in CEP modeling, as it offers a comprehensive framework for optimizing generation, energy storage systems (ESS), and transmission investments. The goal of this research is to verify whether LLM-based optimization can effectively guide capacity expansion decisions by leveraging the tool's existing infrastructure while introducing intelligent optimization capabilities that can reason through complex system interactions and constraints.

The proposed approach employs a Creator-Executor iterative lifecycle framework [1], [8], [9] where the LLM acts as the optimization engine, performing semantic enrichment on model execution results and guiding configuration changes for subsequent iterations. Rather than relying on traditional optimization functions, the LLM processes execution logs through a linear pipeline to make results more interpretable, then uses this enriched understanding to suggest new system configurations [1]. The feedback loop operates as follows: model results are semantically enriched, analyzed for insights, and used to generate suggestions that inform the next iteration's input parameters. This enables the LLM to optimize system-level objectives such as total cost, emissions, or reliability metrics by reasoning through trade-offs in a manner that may be more intuitive than traditional numerical optimization approaches.

 This material is based upon work supported by the U.S. Department of Energy, Office of Electricity (OE), Energy Storage Division.

979-8-3315-8138-1

This work presents an early demonstration of applying LLM-guided optimization in power systems, serving as a verification study to assess whether semantic reasoning can effectively guide infrastructure investment decisions. The integration with the QuESt Planning tool demonstrates how existing CEP frameworks can be enhanced with intelligent optimization capabilities.

II. Methodology: Bilevel CEP with LLM-as-Optimizer

This section outlines our optimization framework that combines a classical bilevel formulation for CEP with an *LLM-as-Optimizer* that oversees the optimization process. Unlike approaches that use an LLM only to tune hyperparameters or provide initialization for a numerical solver, our LLM acts as the optimization engine's overseer: it interprets execution logs, semantically enriches results, and issues explicit configuration updates for the next iteration. The goal is to support system-level evaluations with meaningful, interpretable logs for both domain experts and LLMs.

A. Capacity Expansion Planning

Capacity expansion planning (CEP) models are widely used to support long-term grid investment decisions. CEP is typically formulated as a linear or mixed-integer optimization problem that minimizes net present investment and operational costs over multi-decade planning horizons. These models incorporate constraints spanning generation, storage, reserves, and network operations, and can become highly complex when incorporating nodal resolution, hourly time scales, multiple investment candidates, and uncertain parameters such as capital costs, fuel prices, load forecasts, and weather conditions.

In this paper, the QuESt Planning tool [7], [10], developed by Sandia National Laboratories, is leveraged and tested with the proposed LLM framework. QuESt Planning features flexible modeling such that the modeler can select spatial-temporal scopes and a suite of candidate investment options to investigate modeling tradeoffs and perform robust scenario-based planning. Additionally, QuESt Planning models a broad range of ESS technologies and allows users to define ESS efficiencies, operational strategies, and asset lifetimes. Although QuESt Planning enables robust grid planning, the models can become complex, with tightly coupled constraints that govern critical outcomes. Interpreting and managing the intricate dependencies among the model parameters when building scenarios requires a careful understanding of the mathematical relationships embedded in the simulation. Consequently, these configuration spaces may be difficult to systematically explore using manual scenario construction. In contrast, the proposed LLM framework can effectively manage these relationships through semantic search and execution-log enrichment, dynamically capturing contextual knowledge with each instantiation. These characteristics motivate investigation of LLM-guided configuration strategies to improve planning workflow interpretability and exploration efficiency.

B. Problem Setting and Notation

Let $\theta \in \Theta$ denote the vector of CEP configuration parameters (e.g., technology candidates, bounds, temporal scope choices, or scenario toggles). For any configuration θ, the CEP model produces execution artifacts:

$$f(\theta),\ g(\theta),\ \mathcal{L}(\theta)$$

where f captures scalar performance metrics of interest (e.g., total system cost, or runtime), g summarizes structured results (e.g., capacity build, siting, dispatch statistics presented in graph or table format), and $\mathcal{L}$ is a rich execution log suitable for semantic analysis and auditing.

a) Lower-level CEP (Executor).: Given θ, the Executor solves the CEP optimization (e.g., a linear program) and returns $f, g, \mathcal{L}$:

$$\text{Executor}(\theta) \ \Rightarrow\ \big(f(\theta),\, g(\theta),\, \mathcal{L}(\theta)\big). \tag{1}$$

b) Upper-level Objective (Planner).: The planning objective is to select a configuration θ that optimizes a system-level objective Ω:

$$\min_{\theta\in\Theta}\ \Omega\big(f(\theta), g(\theta)\big) \quad \text{s.t. } \theta \in \mathcal{F}, \tag{2}$$

where $\mathcal{F}$ encodes feasibility rules (policy, engineering constraints, or model validity).

C. LLM-as-Optimizer Overseer

We introduce an LLM-guided operator $\mathcal{O}_\phi$ that consumes the execution artifacts and proposes a new configuration:

$$\theta_{k+1} = \mathcal{O}_\phi\big(\theta_k,\, f(\theta_k),\, g(\theta_k),\, \mathcal{L}(\theta_k)\big). \tag{3}$$

The operator $\mathcal{O}_\phi$ embodies the *semantic enrichment* pipeline: it transforms $\mathcal{L}(\theta_k)$ into interpretable insights and actionable suggestions that directly update CEP model inputs. The update policy follows a hierarchical decision structure that prioritizes feasibility constraint satisfaction, preservation of high-investment planning years identified across execution traces, and reduction of temporal resolution subject to maintaining those constraints. Runtime reduction is not a primary objective, evaluated only after feasibility and investment-anchor preservation criteria are satisfied, reflecting planning workflows where solution validity takes precedence over computational efficiency.

D. Iterative Procedure and Convergence

The end-to-end process alternates between execution and LLM-guided updates:

$$\textbf{(E)}\ (f_k, g_k, \mathcal{L}_k) \leftarrow \text{Executor}(\theta_k), \tag{4}$$

$$\textbf{(O)}\ \theta_{k+1} \leftarrow \mathcal{O}_\phi(\theta_k, f_k, g_k, \mathcal{L}_k). \tag{5}$$

We declare convergence when either (i) successive proposals stabilize, $\|\theta_{k+1} - \theta_k\|_2 \le \varepsilon$, or (ii) the marginal improvement in the planning objective is small, $\big|\Omega(f_{k+1}, g_{k+1}) - \Omega(f_k, g_k)\big| \le \delta$, or (iii) a maximum iteration budget is reached.

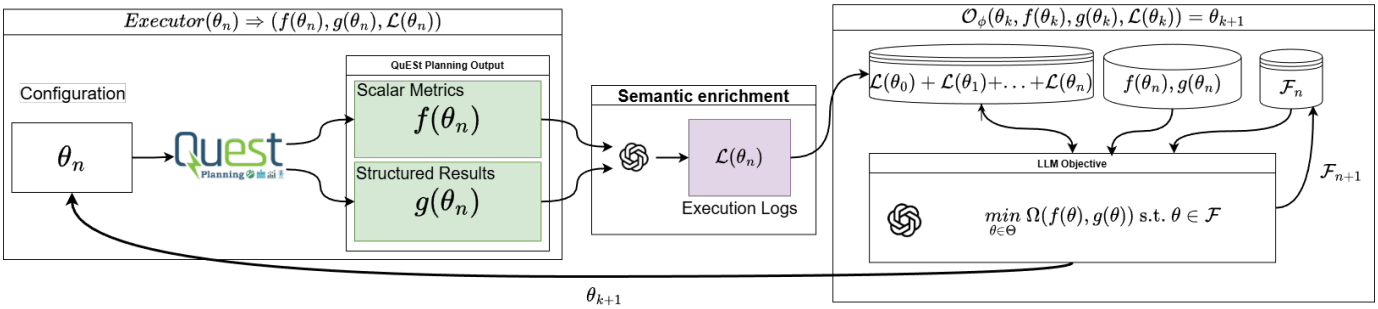

Fig. 1: Proposed LLM optimizer architecture for CEP modeling.

E. Relation to Bilevel Optimization

Equations (1)–(2) define a bilevel perspective: the upper level chooses θ while the lower level solves CEP to optimality. Our contribution differs from prior LLM-for-optimization work that uses LLMs to initialize or tune solvers; instead, $\mathcal{O}_\phi$ replaces the traditional analytic update step with a reasoning-driven, log-grounded policy that operates over system semantics and constraints.

F. Implementation in QuESt Planning

The integration points are: (1) configuration generator (θ assembler), (2) Executor run and result capture, (3) semantic enrichment and suggestion synthesis, and (4) guarded application of updates with feasibility checks for $\mathcal{F}$.

III. LLM Optimizer

The proposed optimization framework integrates a large language model (LLM) with the QuESt Planning execution engine to support semantically informed scenario design and analysis. We employ OpenAI's GPT-5 model as the core reasoning agent and utilize OpenAI's vector store services for semantic retrieval over model documentation, scenario configurations, and enriched execution logs. QuESt Planning operates as the underlying computational engine, executing capacity expansion simulations and producing structured outputs that are conveyed to the LLM through a data-processing pipeline. This pipeline is constructed primarily from OpenAI's standard tooling for document ingestion, vectorization, and function-call orchestration, enabling the LLM to invoke planning functions, interpret simulation traces, and iteratively refine scenario assumptions. The chosen design provides a modular, non-intrusive interface that maintains the clarity of the underlying simulation model while enabling advanced LLM-driven reasoning.

A. System Setup and Data Preparation

The system setup configures the optimizer–executor framework and structures data for LLM consumption. Data is structured around function calls, execution traces, and scenario definitions, with CEP model objectives encoded as structured metadata. Vectorized collections describing CEP functions, model constraints, and relationships are decomposed into text segments. Each scenario configuration is serialized into a structured representation including assumptions, parameter values, and references to CEP outputs. These representations populate a vector database indexed by semantic embeddings and tagged with metadata such as scenario identifier and iteration number, ensuring the LLM has access to a stable, searchable corpus.

During initialization, the LLM runtime is instantiated with tool interfaces for callable CEP functions, scenario execution routines, and log-generation mechanisms. The system initializes a knowledge base that contains feasibility constraints $\mathcal{F}$ that encode domain-specific rules from previous optimization sessions and user-specified instructions. These constraints, stored in the vector database and indexed for semantic retrieval, represent learned patterns about parameter combinations that lead to infeasible solutions, numerical instabilities, or policy violations. The executor initializes the computational environment, loads the current model configuration, and registers template function calls, parameter schemas, and expected output formats alongside the constraint repository, establishing a hierarchical information architecture where feasibility constraints take precedence over general domain knowledge.

B. Semantic Enrichment

Semantic enrichment uses LLM responses to convert text, images, or data into interpretable natural-language summaries that can be easily interpreted by both LLMs and humans. The semantic enrichment phase transforms raw QuESt Planning execution logs into LLM-interpretable artifacts. As CEP simulations execute, they generate intermediate variables, and key performance metrics. These logs are parsed into structured JSON objects, which are then expanded with LLM-generated natural-language explanations of the call sequence, constraints, and the relationships among intermediate outputs. During this process, the LLM leverages its initialized knowledge base to contextualize the execution results. Each enriched log is embedded and stored alongside its originating scenario metadata and prompt with explicit annotations. This enrichment process converts raw execution data into interpretable natural-language summaries, enabling both the LLM and human analysts to understand not only the values and images produced by a run but also the structural dependencies and causal relationships that govern how those values arise.

C. Reasoning

With structured documents, enriched logs, and scenario metadata in place, the LLM performs reasoning by combining semantic retrieval with parameter-aware filtering. The reasoning process operates within a hierarchical information architecture that prioritizes feasibility constraints from the vector store above general execution results and historical summaries. Given a user query, the agent first queries the constraint repository to ensure proposed configurations respect previously learned limitations, then retrieves relevant artifacts and synthesizes a response informed by both hard constraints and empirical patterns. This reasoning process allows the LLM to conduct cross-scenario comparisons, surface correlations between system-level decisions, and interpret how variations in assumptions influence CEP's high-level outcomes while maintaining feasibility. The accumulated constraints act as guardrails that prevent the exploration of invalid configurations. Crucially, this shifts the burden from manual inspection of complex planning models to automated inference that benefits from iteratively refined domain knowledge. The entire reasoning process is operationalized through function calls and interfaces that access the LLM during its prompt API calls.

D. Logging and Context Management

During interactive use, all LLM actions are logged with associated execution context, including tool calls, retrieved artifacts, and generated outputs. These logs support retrieval-augmented reasoning by enabling cross-scenario comparisons through indexed execution traces, function descriptions, and scenario metadata. This context management ensures reproducibility, facilitates iterative refinement of planning scenarios, and supports continuous improvement of the semantic retrieval process as new scenarios and model versions are incorporated into the knowledge base. Although, similar approaches are common in LLM agent frameworks [1], [11], it must be noted that as the model scales, the framework will become increasingly limited by the over-saturation of the global context window. In the current framework context is summarized once the window length is exceeded, however repeated uses of context summary will inevitably result in data loss.

IV. Results

A. Temporal Scope Optimization

Our results focus on evaluating how the choice of temporal scope affects both the computational efficiency and fidelity of CEP simulations. Because most system level parameters in QuESt Planning are dictated by immutable physical constraints, the temporal scope becomes one of the meaningful optimizable parameters. However, increasing temporal resolution rapidly inflates computational costs, making execution time a critical component of the objective of $f(\theta)$ shown in Table I.

To support LLM reasoning over these trade-offs, we embedded feasibility rules into the knowledge base that explicitly described the relationship between temporal resolution, execution time, and model quality. Without these constraints, the LLM tends to treat all of the temporal configurations as equivalent, lacking the context needed to distinguish between high-cost, high-fidelity settings and more computationally tractable alternatives. By encoding this information directly into its indexed vector stores, we provide the model with the semantic structure required to query and interpret temporal scope decisions during iterative inference cycles and guide the optimization process effectively. We do not claim that LLMs outperform black-box optimizers on scalar objective search. Instead, this study evaluates their ability to integrate heterogeneous semantic signals including execution logs, plots, feasibility narratives, and contextual planning knowledge.

TABLE I: Execution times for six temporal configuration strategies, with LLM knowledge base context

Type	Time (s)	LLM Comparison Note
Single	37.455	Fastest configuration; minimal structural overhead
Original	44.132	Stable baseline; moderate computational load
Stage-Right	81.324	Slower tier; moderate synchronization cost
Stage-Even	92.353	Heavy variant; uneven workload distribution
Stage-Left	102.444	High overhead; substantial coordination penalty
Maximum	165.077	Slowest overall; saturates pipeline and amplifies delays

B. Evaluation

Our experimental evaluation focused on temporal scope optimization using a zonal test system [12] with a planning horizon of 2024-2040. As established in our methodology, the LLM operates as the optimization function handler $\mathcal{O}_\phi$, directly generating configuration updates rather than serving as a parameter tuning mechanism or numerical solver initializer. The LLM generated several intermediate configurations through iterative reasoning outputs, ultimately converging on a two-year staggering approach. Convergence emerged from semantic pattern recognition rather than purely numerical optimization. Across evaluated model executions, the LLM consistently identified 2032 as the dominant investment year, exhibiting the highest investment concentration in all but one scenario. Investment timing is encoded as a salient feature within the knowledge base, and the system identifies years with high investment activity as semantically critical planning anchors. This induced stabilization as the optimizer repeatedly preserved 2032 across successive configurations while satisfying feasibility and temporal coverage constraints. Table II summarizes each iteration's reasoning output and corresponding actions, with results shown in Figure 2.

Human-in-the-loop validation was used to ensure that simplified temporal staggering strategies remained physically feasible. Validation involved reviewing operational feasibility indicators including compliance, dispatch feasibility, and absence of infeasible system states across skipped years. This paper focused on optimizing the simulation years of the CEP model, which plays a role in the model's complexity, solve time, and investment plans that grid planners have to rely on to make informed investments. By optimizing this parameter, investments schedules reflect the accuracy needed to account

TABLE II: LLM reasoning outputs and optimization actions $l \in \mathcal{L}(\theta_{n-1})$ on temporal scope optimization

Figure	LLM Reasoning Output	Action
2(1)	Initial baseline configuration	Establish baseline
2(2)	*Probing minimum temporal granularity to assess sensitivity of results to year selection.*	Set to single year
	Invalid configuration detected and insufficient temporal coverage produces no meaningful output.	Reconsider approach
2(3)	*Exploring upper bound of temporal resolution to establish performance ceiling.*	Maximize granularity
2(4)	*Testing non-uniform spacing hypothesis—staggered years may reveal temporal dependencies in investment patterns.*	Stagger years
	Year 2032 exhibits stability across iterations, suggesting critical planning period.	Identify 2032 pattern
2(5)	*Evenly partitioning temporal span to isolate effect of uniform spacing and verify 2032 consistency hypothesis.*	Evenly space years
	Pattern recognition indicates 2032 remains invariant and this temporal anchor should be preserved in subsequent configurations.	Preserve 2032 anchor
2(6)	*Alternating staggering patterns to explore solution space while maintaining 2032 inclusion based on observed stability.*	Alternate 2- and 4-year staggering
2(6+)	*Converging on two-year staggering with 2032 anchor—optimal balance between coverage and computational efficiency.*	Converge on 2-year staggering

for project lead times and supply chain bottlenecks, especially with ESSs. Additionally, the framework enables efficiency in simulation setup, which can result in lower project study periods and labor costs.

V. Conclusions and Future Work

This work demonstrates that LLM optimizers can support long-term grid planning, enabling utilities to make more informed decisions through semantic enhancement of execution logs and reasoning over system-level interactions.

Several limitations remain: the framework relies on a global context window that can introduce lossy compression as data scales, vector-store retrieval may not always match scenario context, and convergence cannot be guaranteed. Future research should explore advanced action-space search mechanisms and frameworks for identifying relaxable constraints to reduce computational overhead in complex scenarios.

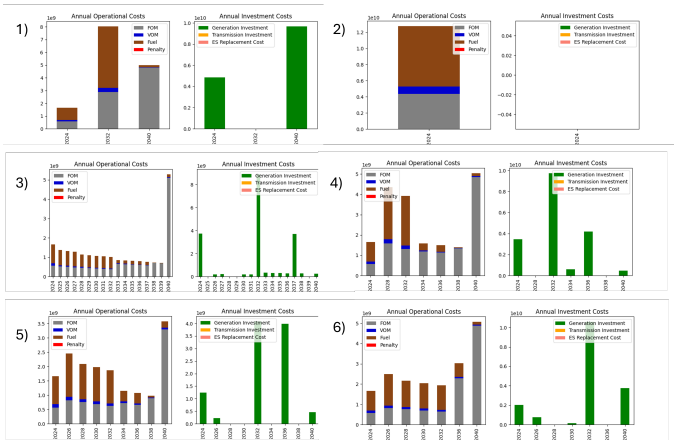

Fig. 2: QuESt Planning - LLM optimization results for each strategy. Annual operations (*right*) and investment costs (*left*).

References

[1] S. Liu, C. Gao, and Y. Li, "Large Language Model Agent for Hyper-Parameter Optimization," Feb. 2025, arXiv:2402.01881 [cs]. [Online]. Available: http://arxiv.org/abs/2402.01881

[2] Y. Inoue, T. Song, X. Wang, A. Luna, and T. Fu, "DrugAgent: Multi-Agent Large Language Model-Based Reasoning for Drug-Target Interaction Prediction," Apr. 2025, arXiv:2408.13378 [cs]. [Online]. Available: http://arxiv.org/abs/2408.13378

[3] C. Lu, C. Lu, R. T. Lange, J. Foerster, J. Clune, and D. Ha, "The AI Scientist: Towards Fully Automated Open-Ended Scientific Discovery," Sep. 2024, arXiv:2408.06292 [cs]. [Online]. Available: http://arxiv.org/abs/2408.06292

[4] A. Chebbi and B. Kolade, "Towards EnergyGPT: A Large Language Model Specialized for the Energy Sector," Sep. 2025, arXiv:2509.07177 [cs]. [Online]. Available: http://arxiv.org/abs/2509.07177

[5] S. Jin and S. Abhyankar, "ChatGrid: Power Grid Visualization Empowered by a Large Language Model," in *2024 IEEE Workshop on Energy Data Visualization (EnergyVis)*. St. Pete Beach, FL, USA: IEEE, Oct. 2024, pp. 12–17. [Online]. Available: https://ieeexplore.ieee.org/document/10747635/

[6] Y. Cheng, H. Zhao, X. Zhou, J. Zhao, Y. Cao, C. Yang, and X. Cai, "A large language model for advanced power dispatch," *Scientific Reports*, vol. 15, no. 1, p. 8925, Mar. 2025. [Online]. Available: https://www.nature.com/articles/s41598-025-91940-x

[7] Sandia National Laboratories, "QuESt Planning: A long-term power system capacity expansion planning tool," https://github.com/sandialabs/quest_planning, version 1.0.0; latest release October, 2024.

[8] S. Tang, C. Priebe, R. Mahapatra, L. Qin, and H. Esmaeilzadeh, "Compiler Optimization via LLM Reasoning for Efficient Model Serving," Jun. 2025, arXiv:2506.01374 [cs]. [Online]. Available: http://arxiv.org/abs/2506.01374

[9] P. Ma, T.-H. Wang, M. Guo, Z. Sun, J. B. Tenenbaum, D. Rus, C. Gan, and W. Matusik, "LLM and Simulation as Bilevel Optimizers: A New Paradigm to Advance Physical Scientific Discovery," May 2024, arXiv:2405.09783 [cs]. [Online]. Available: http://arxiv.org/abs/2405.09783

[10] C. J. Newlun, A. Bera, D. Pandit, G. Cuello-Pollo, A. Lopez, Y. J. Pomeroy, and T. Nguyen, "QuESt Planning: A Long-term Capacity Expansion Planning Tool for Energy Storage Systems & Modeling Flexibility," in *Proceedings of the 2026 IEEE Electrical Energy Storage Applications and Technologies Conference (EESAT)*, Tucson, AZ, USA, 2026, accepted.

[11] K.. M. Jablonka et al., "14 Examples of How LLMs Can Transform Materials Science and Chemistry: A Reflection on a Large Language Model Hackathon," *Digital Discovery*, vol. 2, no. 5, pp. 1233–1250, 2023, arXiv:2306.06283 [cond-mat]. [Online]. Available: http://arxiv.org/abs/2306.06283

[12] C. Barrows, A. Bloom, A. Ehlen, J. Ikäheimo, J. Jorgenson, D. Krishnamurthy, J. Lau, B. McBennett, M. O'Connell, E. Preston *et al.*, "The IEEE reliability test system: A proposed 2019 update," *IEEE Trans. Power Syst.*, vol. 35, no. 1, pp. 119–127, 2019.