Democratizing Energy Management with LLM-Assisted Optimization Autoformalism

Ming Jin, Bilgehan Sel, Fnu Hardeep, Wotao Yin

***Abstract*— This paper introduces a method for personalizing energy optimization using large language models (LLMs) combined with an optimization solver. This approach, termed *human-guided optimization autoformalism*, translates natural language specifications into optimization problems, enabling LLMs to handle various user-specific energy-related tasks. It allows for nuanced understanding and nonlinear reasoning tailored to individual preferences. The research covers common energy sector tasks like electric vehicle charging, HVAC control, and long-term planning for renewable energy installations. This novel strategy represents a significant advancement in context-based optimization using LLMs, facilitating sustainable energy practices customized to individual needs.**

I. Introduction

Despite computational advances, the complex challenges of meeting energy demands and reducing carbon emissions persist. Optimization techniques, as seen in EV charging [1], energy storage [2], renewable investments [3], smart building operations [4], and demand side management [5], hold great promise. **However, the broader democratization of these tools remains a significant hurdle.** This paper tackles this issue, advocating for the accessibility and practicality of advanced computational methods for all, especially the underserved [6]. The emergence of LLMs offers a breakthrough in overcoming these barriers. We demonstrate that LLMs can bridge the gap between the high costs of traditional optimization and the need for personalized, accessible solutions. Our goal is not industry-grade optimization, but rather to provide users with tools for informed decision-making, potentially involving optimization problem formulation. This approach leverages LLMs to streamline modeling and optimization, enabling interaction through natural language without extensive programming or mathematical optimization knowledge.

Technical challenges and solutions. While LLMs such as OpenAI's ChatGPT offer a natural substrate for conversational AI (a technology that enables machines to understand, interpret, and engage in natural conversations), it is essential to recognize that current LLMs have not excelled in tasks like arithmetic and logical reasoning [7]. Recent works have introduced methods such as chain of thoughts [8] (and variants such as algorithm of thoughts [9]), which prompts a series of intermediate reasoning steps, and autoformalism [10], which automatically translates natural language mathematics to formal specifications and proofs. However, many energy problems, like energy storage control and long-term planning for PV panel installation, demand complex decision-making. These problems differ from arithmetic reasoning, commonsense reasoning, and symbolic reasoning in the following aspects: *Complexity*, as they often involve numerous variables, constraints, and objectives with potentially nonlinear relationships between variables; and *incomplete information*, as energy systems are influenced by various factors, such as user preferences, which may not be provided in the initial problem description.

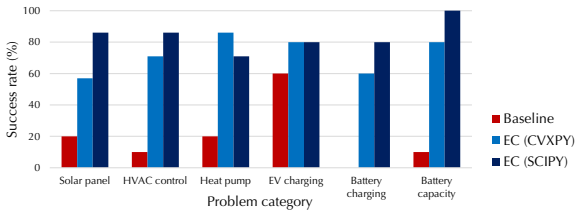

Fig. 1: Comparison of the baseline method (simple prompting with GPT-4) and EC using either CVXPY or SCIPY as the code projection layer across various problem categories. Success rate denotes the percentage of instances in which the provided answer aligns with user expectations, satisfies all given constraints, and is either close to or exactly optimal.

A. Contributions

In this research, we address the above challenges as follows. To manage complexity, we develop a procedure to map a problem description to code using the grammar of packages that support optimization formulation, and to iteratively formulate, debug, and execute the program. We also introduce external tooling to execute the written code and solve the formulated optimization using dedicated algorithms. Furthermore, we implement prompt engineering techniques to enable LLMs to accurately understand and respond to user-specific preferences. To address incomplete information, we leverage the reasoning capabilities of LLMs to identify key parameters and use a question-and-answer format in natural conversations to solicit this information from users. Additionally, we employ LLMs for auto-informalism to explain the solution to users. An overview of the EC framework is depicted in Fig. 2. This approach bears dual benefits: it not only guides the user through complex nonlinear reasoning tasks of energy saving planning but also offers an insightful explanation of the solution.

We further demonstrate the possibility of our proposed approach in solving a variety of tasks within the energy domain, ranging from EV charging, HVAC and battery control, to long-term planning problems such as cost-benefit evaluation of installing rooftop solar PVs, heat pumps, and battery sizing. In comparison to the simple prompting method, where the task description is directly presented to the LLM,

our proposed EC framework improves the success rate, as illustrated in Fig. 1. Although the optimizations formulated are not exceedingly sophisticated, we find that the LLM effectively addresses the problem to a considerable degree. Throughout the remainder of the paper, unless specified otherwise, we refer to OpenAI's ChatGPT (GPT-4) when mentioning LLM.

B. Related Work

Recent work has explored using LLMs for mathematical autoformalization, converting natural language into formal languages like Isabelle/HOL [11]. Analogously, we use intermediaries like SCIPY and CVXPY to transform intuitive formulations of optimization problems into standard form [12]. Despite few CVXPY examples online, LLMs exhibit surprising adeptness, albeit with occasional syntax errors that we correct via Python debugging. While LLMs struggle with multi-step problems [13], techniques like in-context learning [14] and algorithmic search [9] have shown promise. We present a novel viewpoint using optimization itself as a reasoning tool to enrich existing LLM augmentation techniques [15]. LLMs can also revise responses given feedback, e.g. in question-answering and code debugging [16]. We incorporate revisions based on programming language feedback and user requests, enabling rapid error correction.

Optimizing energy systems requires commonsense reasoning and domain knowledge [17]. Prior work has used ML and expert systems [18], but we harness LLM expertise for optimization through natural language interaction. To our knowledge, we are the first to tackle non-trivial, energy-specific problems with incomplete information this way.

The remaining sections of this paper are structured as follows. Section II describes our proposed framework in detail, including the design principles and the optimization autoformalism approach. Section III presents experimental results that demonstrate the effectiveness of our approach in solving various energy-related tasks. We discuss the potential and future directions of conversational AI for energy sustainability and conclude in Section IV.

II. Energy Concierge Framework

Using an EV charging query as an example, our Energy Concierge operates as follows:

1) User submits natural language request (e.g. optimize my charging schedule). LLM determines if it is an optimization problem.
2) If so, LLM requests input parameters needed to solve it. It may ask clarifying questions (e.g. charging capacity? preferred hours?).
3) LLM formulates Python code with user information, translating the query into an optimization problem via autoformalism.
4) Interface executes code to solve problem, with debugging iterations if needed.
5) LLM explains the optimal solution clearly (charging schedule, cost savings). Enabling informed user decisions.

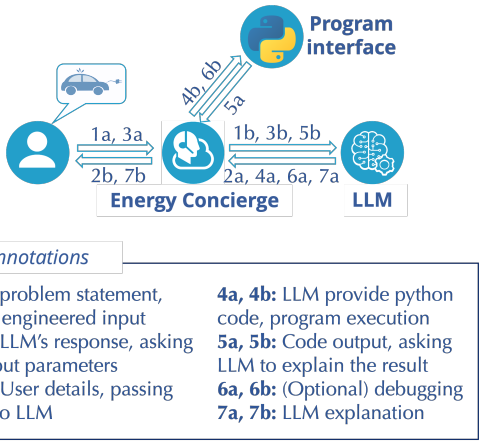

Fig. 2: Energy Concierge framework. The user engages with an LLM through natural language queries and responses. The LLM identifies the necessary input parameters for optimization and generates Python code to address the problem. The program interface then executes the code and relays the solution back to the LLM, which subsequently provides a clear explanation to the user.

Through interactivity and personalization grounded in optimization and autoformalism, this conversational framework empowers users to improve energy efficiency.

A. Optimization autoformalism

A general optimization problem can be written as:

$$\begin{aligned} \text{minimize}_{x\in\mathbb{R}^p} \quad & f(x;\theta) \\ \text{subject to} \quad & g_i(x;\theta) \le 0, \quad i = 1,2,\dots,m \qquad (P(\theta)) \\ & h_j(x;\theta) = 0, \quad j = 1,2,\dots,n, \end{aligned}$$

where we seek to minimize the objective function $f(x;\theta)$ subject to a set of inequality constraints $g_i(x;\theta) \le 0$, and equality constraints $h_j(x;\theta) = 0$. Here, $x \in \mathbb{R}^p$ is the decision variable and θ is the collection of hyperparameters that defines the optimization instance, including objective and constraint functions; in other words, the solution of the optimization $P(\theta)$ can be regarded as a function of the hyperparameters θ [19].

Optimization techniques can be used to solve energy-related problems [1]–[5]. Automating the formulation and solution of optimization problems is essential due to the technical skills gap, challenges in manually incorporating user preferences and constraints, and the inefficiencies in manual modifications based on user feedback. The human-guided autoformalism proposed in this study automatically translates natural language task specifications to optimization instances. However, directlying implementing this approach may face issues such as ambiguity, incompleteness, and incorporating user-specific preferences. The subsequent subsections will delineate strategies to address them.

1) Optimization formulation: The optimization process begins with identifying the objective function, decision variables, and constraints from the user's task description [T]. We tested two approaches:

- Approach 1: Directly ask the LLM to identify the optimization components in [T].
- Approach 2: First prompt the LLM to identify the 5 most important parameters in [T], then use these parameters to formulate the optimization instance.

Approach 2 outperforms Approach 1 because LLMs like GPT-4 are trained on diverse data, not just optimization tasks. Asking for the top 5 parameters simplifies the task and provides context, helping LLMs generate more precise optimization formulations [8], [9]. Directly requesting optimization formulations (Approach 1) can introduce ambiguityDirectly requesting optimization formulations can introduce ambiguity; initial parameter identification (Approach 2) reduces this uncertainty.

After identifying the essential parameters, the natural language query is transformed into a computational instance using the prompt: "Write a Python code using [lib] to solve this optimization problem," where [lib] is either CVXPY or SCIPY. We found that SCIPY yields a higher success rate (71% to 100%) compared to CVXPY (51% to 80%) (see Fig. 1). This difference is likely due to SCIPY's broader acceptance and longer existence in the Python community, resulting in more SCIPY-related content available online for the model to leverage during training.[1]

2) Solving an optimization and debugging: To address LLMs' limitations in nonlinear reasoning [7], we enable the model to interact with an external Python program to solve optimization tasks. We extract the code block from the LLM's output using regular expressions, searching for unique delimiters enclosing the code. This method reliably extracts code without requiring a large labeled dataset, unlike training a machine learning model [20]. During development, we encountered two types of errors in the LLM-generated code: erroneous translation into an optimization problem and syntactic bugs in an otherwise correctly translated problem.[2]

To rectify translation errors, we rely on user interaction and clarification of the formulated optimization. For syntactic errors, we use an automated process: identify the error message, feed it to the LLM to isolate relevant code snippets, generate potential remedies, and assess the proposed solutions in a new iteration (Fig. 7 in Appendix). Taking multiple code samples before debugging often results in fewer LLM queries (Fig. 5). By following this procedure, we successfully resolved most errors encountered during our experiments.

3) Optimization auto-informalism: After discovering the optimal solution, the LLM articulates the results in a comprehensible, natural language format. This system provides detailed explanations of the optimal solution and any constraints or preferences factored in during the optimization process (Steps 7a and 7b in Fig. 2). This auto-*in*formalism approach complements autoformalism by offering intuitive insight into the optimization outcomes.

[1]A search on GitHub (as of 4/25/24) returns 8K repositories for "SCIPY" and only 272 for "CVXPY," supporting our preference for SCIPY.

[2]Errors can also arise due to the limitations of the optimization solvers, but since most energy problems involve linear or quadratic convex optimization, we presume such errors are comparatively infrequent. We do not classify infeasibility as an error.

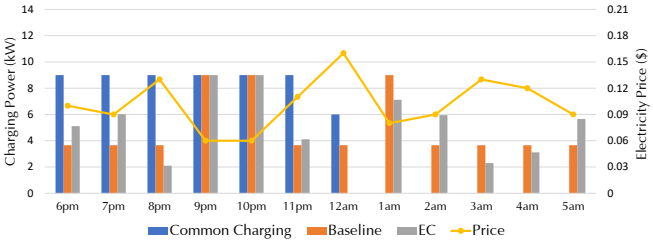

Fig. 3: A comparison of EV charging plans to improve the cost and load on the grid. Observations reveal that EC effectively capitalizes on hours with lower prices in contrast to the baseline.

III. Experiments and Case Studies

This section examines the potential of LLM-based autoformalism in real-time decision-making (Sec. III-A) and sustainable long-term planning (Sec. III-B). A general analysis of the proposed methods is provided in Sec. III-C. The [21, appendix] offers example interactions.

A. Real-time decision making

1) Smart EV charging: The core issue in smart EV charging is to optimize EV charging patterns to balance power grid loads, mitigate energy costs, and fulfill user preferences [1]. EC can correspond user-provided information to optimization parameters, such as the maximum charging rate. It grasps the user's charging availability and translates this information into decision variables and constraints, leading to an optimized schedule (see Fig. 3).

2) HVAC control: HVAC control is a pivotal issue that has been extensively studied [4]. However, the algorithms that have been developed are often more advanced than the current control panels in most buildings, leading to a disconnect between sophisticated methods and user comprehension. The heart of the issue is to discover a setpoint that ensures a comfortable indoor climate while minimizing energy consumption. EC provides clear and actionable advice like "Set your thermostat to the optimal temperature (75°F in this case) during hot, humid days..." with a clear rationale. It also offers customized suggestions based on user conditions, along with valuable energy efficiency tips such as "monitor your energy consumption", and "adapt to changing conditions". Nevertheless, the cost of simplicity is the inability to perform sophisticated controls like pre-heating/cooling, which entails heating the room in anticipation of occupancy and can only be accounted for through a multiperiod formulation involving intricate room thermal dynamics [4].

3) Battery charging control: This problem involves optimizing the charging and discharging cycles of a home battery system, considering factors such as electricity pricing, solar generation, and household demand, as seen in competitions like the CityLearn Challenge [22]. As discussed in [23], selecting the appropriate optimization problem for a given context is crucial. Unlike [23], where the context is derived

from a reward signal, our context is provided by the user in natural language. The EC interprets this context accurately to establish the right optimization problem.

EC successfully formulates an accurate objective that computes the weighted sum of total electricity cost, considering the electricity price. It also correctly formulates the lower and upper limits for the charging and discharging variables, the temporal interdependence of the state of charge and the charging rate, and the maximum capacity constraint. The energy balance constraint identified demonstrates LLM's understanding of world modeling, including this constraint *without* specific user instruction. EC's explanation further shows an understanding of physical constraints and its overall objective. However, EC implicitly assumes that excess energy produced each hour can be sold back to the grid at the same rate as it was purchased, which is not typically the case. Future work can explore methods to encourage EC to be more explicit about assumptions when constructing optimization instances.

B. Long-term planning for sustainability

1) Cost-benefit analysis of installing rooftop solar PVs: To perform a cost-benefit analysis of solar PV, one must estimate the costs and benefits over the system's lifespan and compare them to a relevant alternative system [24].

EC shows proficiency in understanding physical constraints, such as the stipulation that the panel area shouldn't surpass the roof area, and proposes that the installed area be adequate to supply the electricity demand. However, it operates under a few assumptions, such as the user planning to source all electricity demand from PV to fully leverage the budget, and that the PV's efficiency stands at 0.12. While these assumptions are generally sensible, stating them in the explanations would make the model's workings more transparent. Additionally, the model seems to underutilize the user's provided information, such as the building's location, which could potentially inform the required area-to-power conversion efficiency more accurately. This is understandable given the model's lack of access to an external database, and future work that enhances this capability could be a worthwhile pursuit.

2) Cost-benefit analysis of installing a heat pump: Key considerations in the cost-benefit analysis of a heat pump include the initial cost of installation, which can vary based on the type, size, and complexity of the system, as well as the availability of incentives and rebates [25].

Similar to the PV installation scenario in Sec. III-B.1, EC proficiently identifies pivotal relationships, including the annual operating cost of a central AC and a heat pump. EC's approach focuses on optimizing annual savings, leaving the actual estimation of the payback period to the user. For instance, EC calculates that the annual savings with a heat pump equals $550, then elaborates on how to use this data for investment decisions, suggesting "If the purchase and installation of the heat pump cost $5,000, the payback period would equate to around 9.1 years ($5,000 / $550)."

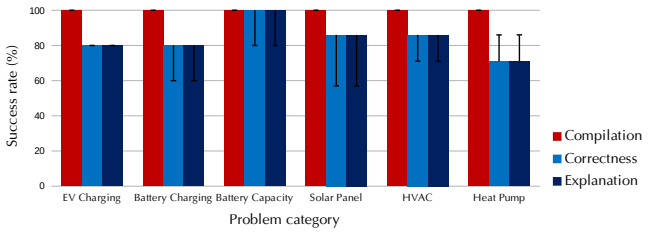

Fig. 4: The efficiency of EC when integrated with SCIPY is gauged by the compilation success of the Python code it generates, the solution's precision, and the lucidity of the explanation presented to the user. The black lines highlight the performance shifts observed when employing CVXPY.

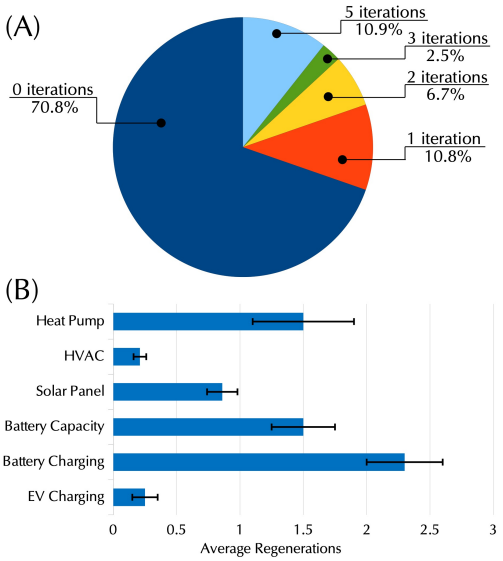

Fig. 5: (A) A comparative analysis of the number of iterations needed to achieve executable code across all problem categories (note that none of the experimental instances use 4 iterations). (B) A comparison of the average number of code generations needed for EC to produce executable code, limited to a maximum of 5. The black bars represent the variation in iterations across problem categories over 20 random runs.

EC also suggests users compare alternatives, plan long-term strategies, evaluate environmental impact, and seek incentives or rebates. This shows that EC can provide not only a decision but also the context and information necessary to empower users to make more cost-effective and environmentally conscious choices.

C. General findings and analysis

To elucidate the types of errors EC commits, we display the success rates concerning error-free code production, logical correctness of the code, and the ability to articulate the solution derived from the optimization code to the end user in Fig. 4. Our framework enables EC to achieve a 100% compilation rate. When EC generates an error-free optimiza-

tion code, it consistently explains the results to the end user. SCIPY outperforms CVXPY in overall performance, aligning with our observation that SCIPY has broader online recognition and use, where most LLMs receive training.

Fig. 5 presents the average number of code regenerations needed and the average debugging iterations required to obtain syntax error-free optimization code. Although each subproblem within the optimization problem category exhibits different success rates, a relatively low number of code regenerations are typically needed. When the LLM consistently commits errors, incorporating debugging information in our framework is essential for achieving optimization solutions. Reducing the number of generated code samples from $s = 5$ to $s = 1$ slightly increases the required debugging iterations across all examples, suggesting that debugging is a more efficient error-handling strategy compared to simple regeneration.

Estimating the Probability of One-Round Autoformalism Success. The aforementioned results allow us to infer the probability of generating valid code successfully, denoted as p. We postulate that the event of successfully generating valid code is independent and identically distributed for the same category of problems, following a truncated geometric distribution with a cap at 5 trials. Based on this supposition, we deduce that $\sum_{k=1}^{5} k(1-p)^{k-1}p + 6\sum_{k\geq 6}(1-p)^{k-1}p = z$, where z signifies the number of generations needed to achieve success (with generations capped at 5, as shown in Fig. 5), and p is the probability of success to be estimated. Substituting the values for z from Fig. 5 into the equation and solving for p, we obtain the one-round autoformalism success rates $p = 0.8, 0.25, 0.38, 0.53, 0.83, 0.38$ for the problems of EV Charging, Battery Charging, Battery Capacity, Solar Panel, HVAC Control, and Heat Pump Investment, respectively.

Estimation of the Probability of Debugging Success. We posit that the event of successful debugging, given the presence of an erroneous code, is independent and identically distributed across all problem classes. Using Fig. 5, we first normalize the frequency of the required number of debugging iterations by the frequency of generating an erroneous code in the first run (i.e., $1 - 0.7 = 0.3$). This reveals that the frequencies of observing debugging iterations 1, 2, 3, 4, and ≥ 5 are represented by y_k, where $k \in \{1, 2, ..., 5\}$. Let q denote the probability of successfully debugging the code. From this, we can construct a system of polynomial equations in the form of $(1-q)^{k-1}q = y_k$ for $k \in \{1, 2, 3, 4\}$, and $1 - \sum_{k=1}^{4}(1-q)^{k-1}q = y_5$ given that debugging is capped at 5 iterations. Using a line search between 0 and 1 in increments of 0.01, we calculate the values of $\hat{y}_1$ to $\hat{y}_5$ for each q using the specified equations, and determine the mean squared error (MSE) between the vectors $\hat{y}$ and y. Our optimal estimate for the probability is $q = 0.26$.

Optimality Gap and Improvement Over Baseline. The optimality gap for a feasible solution is calculated as the ratio $v/v^* - 1$, where v denotes the objective value of the candidate solution and v^* signifies the optimal value of the corresponding minimization problem. In Fig. 4, we classify test cases as correct only if they equate to the globally optimal solution value, i.e., a 0% optimality gap. Additional results presented in Fig. 6 demonstrate the average optimality gap when incorrect logic within the code results in the omission of key parameters (such as the efficiency of a battery). Our framework was evaluated using 20 examples for each energy optimization problem, including EV charging. As depicted, the instances display an optimality gap generally within the 20% range, with the majority under 10%.

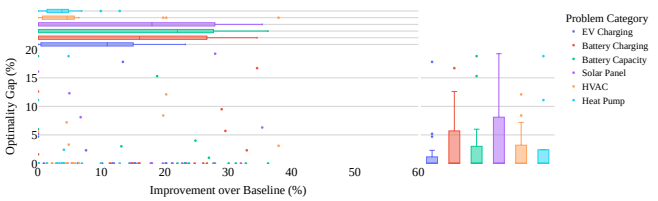

Fig. 6: We assess the average optimality gap by comparing the exact solution of the optimization problem with the solution determined by EC, alongside evaluating the enhancement over the baseline in terms of optimality.

For the baseline method, the model was simply prompted with our question along with necessary parameters to solve the optimization problem. We noticed that even when the problem is clearly an optimization issue, LLMs may opt to generate responses by attempting to apply logic towards reaching an answer. As seen in Fig. 6, this approach does not yield favorable results in comparison to the EC framework. This can be attributed to the fact that many energy-related optimization problems of importance do not yield closed-form solutions, making the correct utilization of convex program solvers critical.

We further report on the improvement over the baseline, measured by $v_b/v - 1$, where v_b represents the objective value of the baseline solution. As is evident, the improvement can amount to as much as 60%, with most instances falling within the 30% range. Some of the problems yielding the greatest improvements include Solar Panel, Heat Pump, Battery Charging, and EV Charging, which encompass both real-time decisions and long-term investments.

IV. Discussions and Conclusion

Conversational AI for sustainability has long seemed aspirational, but LLMs offer new potential. By providing an intuitive interface, our framework helps individuals engage more consciously about energy use. Greater awareness drives sustainable behaviors like responsible consumption, efficiency, and solar adoption [26], [27].

Households average 3.5 kWh peak electricity use, while EV chargers approach 10 kWh. EV charging at peak triples load, straining infrastructure. Utilities offer off-peak rates, but financial incentives alone don't fully shift behaviors—an "efficiency gap" phenomenon [26], [27]. Our framework simplifies this via conversation. With 2022's 3 million EVs, adopting off-peak charging could save $876 million annually (at $0.06 vs $0.14 per kWh rates) [28], [29]. With 2035's

projected 73 million EVs [30], savings could reach $21.3 billion. Beyond consumer savings, off-peak charging lessens grid stress, stabilizes prices, and reduces needs for new plants. By simplifying optimization, our system enables impactful sustainability actions.

A key limitation of EC is its reliance on the LLM's ability to accurately formulate optimization problems, ensuring proper alignment of objectives and constraints. Incorrect formulations can lead to ineffective or harmful solutions, despite the perceived reliability of these systems. Auto-informalism (Sec. II-A.3) helps mitigate this risk by scrutinizing problem formulations more carefully and alerting users to potential issues or assumptions. Enhancing auto-informalism's effectiveness is thus critical for reliable solutions. Additional reliability methods like validation routines, robustness testing, and expanded user feedback loops can further strengthen the framework.

While we merely sketch out the potential, the proposed shift towards human-guided optimization autoformalism could democratize access to sophisticated technologies and set the stage for a more equitable and sustainable future.

REFERENCES

[1] S. M. Arif, T. T. Lie, B. C. Seet, S. Ayyadi, and K. Jensen, "Review of electric vehicle technologies, charging methods, standards and optimization techniques," *Electronics*, vol. 10, no. 16, p. 1910, 2021.

[2] R. Khezri, A. Mahmoudi, and H. Aki, "Optimal planning of solar photovoltaic and battery storage systems for grid-connected residential sector: Review, challenges and new perspectives," *Renewable and Sustainable Energy Reviews*, vol. 153, p. 111763, 2022.

[3] Y. Yang, S. Bremner, C. Menictas, and M. Kay, "Battery energy storage system size determination in renewable energy systems: A review," *Renewable and Sustainable Energy Reviews*, vol. 91, pp. 109–125, 2018.

[4] J. Drgoňa, J. Arroyo, I. C. Figueroa, D. Blum, K. Arendt, D. Kim, E. P. Ollé, J. Oravec, M. Wetter, D. L. Vrabie *et al.*, "All you need to know about model predictive control for buildings," *Annual Reviews in Control*, vol. 50, pp. 190–232, 2020.

[5] B. P. Esther and K. S. Kumar, "A survey on residential demand side management architecture, approaches, optimization models and methods," *Renewable and Sustainable Energy Reviews*, vol. 59, pp. 342–351, 2016.

[6] J. Currie, D. I. Wilson, N. Sahinidis, and J. Pinto, "Opti: Lowering the barrier between open source optimizers and the industrial matlab user," *Foundations of computer-aided process operations*, vol. 24, p. 32, 2012.

[7] J. W. Rae, S. Borgeaud, T. Cai, K. Millican, J. Hoffmann, F. Song, J. Aslanides, S. Henderson, R. Ring, S. Young *et al.*, "Scaling language models: Methods, analysis & insights from training gopher," *arXiv preprint arXiv:2112.11446*, 2021.

[8] J. Wei, X. Wang, D. Schuurmans, M. Bosma, F. Xia, E. H. Chi, Q. V. Le, D. Zhou *et al.*, "Chain-of-thought prompting elicits reasoning in large language models," in *Advances in Neural Information Processing Systems*, 2022.

[9] B. Sel, A. Al-Tawaha, v Vanshaj, R. Jia, and M. Jin, "Algorithm of thoughts: Enhancing exploration of ideas in large language models," *ICML*, 2024.

[10] Y. Wu, A. Q. Jiang, W. Li, M. N. Rabe, C. E. Staats, M. Jamnik, and C. Szegedy, "Autoformalization with large language models," in *Advances in Neural Information Processing Systems*, 2022.

[11] Y. Wu, A. Q. Jiang, W. Li, M. Rabe, C. Staats, M. Jamnik, and C. Szegedy, "Autoformalization with large language models," *Advances in Neural Information Processing Systems*, vol. 35, pp. 32 353–32 368, 2022.

[12] A. Agrawal, R. Verschueren, S. Diamond, and S. Boyd, "A rewriting system for convex optimization problems," *Journal of Control and Decision*, vol. 5, no. 1, pp. 42–60, 2018.

[13] J. Austin, A. Odena, M. Nye, M. Bosma, H. Michalewski, D. Dohan, E. Jiang, C. Cai, M. Terry, Q. Le *et al.*, "Program synthesis with large language models," *arXiv preprint arXiv:2108.07732*, 2021.

[14] J. Wei, X. Wang, D. Schuurmans, M. Bosma, F. Xia, E. Chi, Q. V. Le, D. Zhou *et al.*, "Chain-of-thought prompting elicits reasoning in large language models," *Advances in Neural Information Processing Systems*, vol. 35, pp. 24 824–24 837, 2022.

[15] G. Mialon, R. Dessì, M. Lomeli, C. Nalmpantis, R. Pasunuru, R. Raileanu, B. Rozière, T. Schick, J. Dwivedi-Yu, A. Celikyilmaz *et al.*, "Augmented language models: a survey," *arXiv preprint arXiv:2302.07842*, 2023.

[16] T. X. Olausson, J. P. Inala, C. Wang, J. Gao, and A. Solar-Lezama, "Demystifying gpt self-repair for code generation," *arXiv preprint arXiv:2306.09896*, 2023.

[17] A. B. Saka, L. O. Oyedele, L. A. Akanbi, S. A. Ganiyu, D. W. Chan, and S. A. Bello, "Conversational artificial intelligence in the aec industry: A review of present status, challenges and opportunities," *Advanced Engineering Informatics*, vol. 55, p. 101869, 2023.

[18] R. Panchalingam and K. C. Chan, "A state-of-the-art review on artificial intelligence for smart buildings," *Intelligent Buildings International*, vol. 13, no. 4, pp. 203–226, 2021.

[19] M. Jin, V. Khattar, H. Kaushik, B. Sel, and R. Jia, "On solution functions of optimization: Universal approximation and covering number bounds," *AAAI*, 2023.

[20] S. Iyer, I. Konstas, A. Cheung, and L. Zettlemoyer, "Summarizing source code using a neural attention model," in *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2016, pp. 2073–2083.

[21] M. Jin, B. Sel, H. FNU, and W. Yin, "Democratizing energy management with llm-assisted optimization autoformalism," 2024, full Version: http://www.jinming.tech/papers/ChatEnergy23-smartgridcomm.pdf.

[22] J. R. Vázquez-Canteli, J. Kämpf, G. Henze, and Z. Nagy, "Citylearn v1.0: An openai gym environment for demand response with deep reinforcement learning," in *Proceedings of the 6th ACM International Conference on Systems for Energy-Efficient Buildings, Cities, and Transportation*, ser. BuildSys '19. New York, NY, USA: Association for Computing Machinery, 2019, p. 356–357. [Online]. Available: https://doi.org/10.1145/3360322.3360998

[23] V. Khattar and M. Jin, "Winning the citylearn challenge: Adaptive optimization with evolutionary search under trajectory-based guidance," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 37, no. 12, 2023, pp. 14 286–14 294.

[24] M. Thebault and L. Gaillard, "Optimization of the integration of photovoltaic systems on buildings for self-consumption–case study in france," *City and Environment Interactions*, vol. 10, p. 100057, 2021.

[25] F. Belaïd, Z. Ranjbar, and C. Massié, "Exploring the cost-effectiveness of energy efficiency implementation measures in the residential sector," *Energy Policy*, vol. 150, p. 112122, 2021.

[26] O. I. Asensio and M. A. Delmas, "Nonprice incentives and energy conservation," *Proceedings of the National Academy of Sciences*, vol. 112, no. 6, pp. E510–E515, 2015.

[27] T. D. Gerarden, R. G. Newell, and R. N. Stavins, "Assessing the energy-efficiency gap," *Journal of economic literature*, vol. 55, no. 4, pp. 1486–1525, 2017.

[28] APPALACHIAN POWER, "Virginia s.c.c. tariff no. 26 appalachian power company," 2021. [Online]. Available: https://www.appalachianpower.com/lib/docs/ratesandtariffs/Virginia/Tariff26-MASTER-Standard-June1-2023RPS-RAC.pdf

[29] IEA, "Trends in electric light-duty vehicles," 2023. [Online]. Available: https://www.iea.org/reports/global-ev-outlook-2023/trends-in-electric-light-duty-vehicles

[30] NREL, "National economic value assessment of plug-in electric vehicles," 2016. [Online]. Available: https://www.nrel.gov/docs/fy17osti/66980.pdf

[31] H. C. Hesse, R. Martins, P. Musilek, M. Naumann, C. N. Truong, and A. Jossen, "Economic optimization of component sizing for residential battery storage systems," *Energies*, vol. 10, no. 7, p. 835, 2017.