Large language models for power scheduling: A user-centric approach

Thomas Mongaillard, Samson Lasaulce, Othman Hicheur, Chao Zhang, Lina Bariah, Vineeth S. Varma, Hang Zou, Qiyang Zhao, Merouane Debbah

Large Language Models for Power Scheduling: A User-Centric Approach

Thomas Mongaillard*, Samson Lasaulce†*, Othman Hicheur†‡, Chao Zhang†§, Lina Bariah ¶†, Vineeth S. Varma*, Hang Zou ‖, Qiyang Zhao ‖, and Merouane Debbah†‖
* Université de Lorraine, CNRS, CRAN, F-54000 Nancy, France.
† KU 6G Research Center, Khalifa University, Abu Dhabi, UAE.
‡ Ecole Polytechnique, Paris, France. §Central South University, Changsha, China.
¶Open Innovation AI, Abu Dhabi, UAE. ‖TII, Abu Dhabi, UAE

***Abstract*—While traditional optimization and scheduling schemes are designed to meet fixed, predefined system requirements, future systems are moving toward user-driven approaches and personalized services, aiming to achieve high quality-of-experience (QoE) and flexibility. This challenge is particularly pronounced in wireless and digitalized energy networks, where users' requirements have largely not been taken into consideration due to the lack of a common language between users and machines. The emergence of powerful large language models (LLMs) marks a radical departure from traditional system-centric methods into more advanced user-centric approaches by providing a natural communication interface between users and devices. In this paper, for the first time, we introduce a novel architecture for resource scheduling problems by constructing three LLM agents to convert an arbitrary user's voice request (VRQ) into a resource allocation vector. Specifically, we design an LLM intent recognition agent to translate the request into an optimization problem (OP), an LLM OP parameter identification agent, and an LLM OP solving agent. To evaluate system performance, we construct a database (EVRQ) of typical VRQs in the context of electric vehicle (EV) charging. As a proof of concept, we primarily use Llama 3 8B. Through testing with different prompt engineering scenarios, the obtained results demonstrate the efficiency of the proposed architecture. The conducted performance analysis allows key insights to be extracted. For instance, having a larger set of candidate OPs to model the real-world problem might degrade the final performance because of a higher recognition/OP classification noise level. [Paper codes and video[1]]**

***Index Terms*—Large language model, multi-agent, optimization, power scheduling, EV charging, smart grid, resource allocation, user-centric.**

I. Introduction

With the rapid evolution of complex systems across various domains, the need for sophisticated scheduling schemes to efficiently manage the system resources and meet certain requirements has become increasingly critical. While recently we have witnessed noticeable advancements in the development of scheduling algorithms (particularly with leveraging artificial intelligence (AI)-driven methods), it is essential to emphasize that these traditional algorithms are often designed to satisfy predefined constraints that are inherently tied to the specific system they serve. Although they have proven effective in many scenarios, they often fail to perform well when faced with the dynamic and personalized demands of modern users.

In addition to the fact that they might be uncapable of satisfying the level of quality-of-experience (QoE) imposed by services that are increasingly oriented towards satisfying user demands, conventional methods might result in high complexity and not necessarily lend themselves into an optimum solution with respect to energy-efficiency in personalized scenarios. Taking the example of energy management, a key aspect underpinning reliable and sustainable operations in many systems, including wireless communication networks, autonomous vehicular systems, smart grids, etc., humans remain largely out of the loop. For instance, for heating or air conditioning (AC) systems, humans typically just provide the target temperature, and a more or less advanced regulation algorithm does all the rest. This approach operates independently of individual user preferences, or potential changes in energy availability or costs. Consequently, it may not always optimize for energy-efficiency or user comfort, demonstrating a clear limitation.

The reason for this gap between humans and algorithms is twofold. First, most humans are not able to model mathematically a real-world problem and solve it, which is why algorithms are typically tasked with making most decisions. Second, humans cannot communicate easily or not at all with algorithms, machines, or programs. However, the emergence of advanced natural language processing (NLP) tools, such as large language models (LLMs), is completely changing the paradigm. In particular, LLMs enable intuitive and effective human-machine interactions, transforming the operation of complex infrastructures, such as energy and wireless networks, into more responsive and user-centric solutions.

It should be noted that the problem of power scheduling is a key in both wireless networks and in digitalized energy networks. Many wireless resource management problems and home energy management problems can be formulated as power scheduling problems [1]–[5]. The main goal of this paper is to leverage the capabilities of LLMs in order to enable the machine to convert a voice request (VRQ) from a human

The authors acknowledge the KU-TII 6G Chair on Native AI.

[1]https://github.com/thomasmong/llm-power-scheduling

user into a power scheduling vector. For example, for the case study under consideration in this paper (namely electric vehicle -EV- charging), such a request can be: "Charge my EV for tomorrow at 6 a.m. while managing its battery lifetime". For a cell phone, it might be "Adapt your transmit power to minimize electromagnetic exposure while guaranteeing my SMS messages always go through". *The novel approach to power scheduling introduced and developed in this paper is to exploit the knowledge the LLM has acquired during pre-training and auxiliary instructions to both model and solve the problem at hand.* It is important to highlight that, although we develop this approach for the particular case of the power scheduling problem, such an approach can be generalized to help humans solve a wide range of real-world problems by leveraging mathematical modeling, reasoning, and solving capabilities.

Nowadays, there are several services, e.g., Amazon Alexa, Google Smart Home, and Siri-based Apple Smart Home, with advanced interfaces that allow humans to "talk" to devices such as electrical appliances. However, in all these solutions, the employed deep learning algorithms only act as mere classifiers, i.e., they classify the user's request into a given control action (e.g., switch a given electric appliance on). Therefore, the implemented deep learning schemes do not try to model mathematically or interpret the physical problem at hand, and they thus they do not attempt to solve it by exploiting the interpretation, reasoning, and planning capabilities that LLMs-at least partially-have (see, e.g., [6] [7]). Additionally, existing deep learning solutions are trained for particular tasks, and hence, they do not generalize to a wider range of tasks, as would be encountered when treating VRQs made by a human user. Therefore, the standpoint of this paper is original in the sense that the proposed architecture aims at exploiting LLMs to mathematically model the physical problem at hand and solve it. To our knowledge, the closest literature to this approach is given by the literature of math word problems (see e.g., [8]–[11]). In this literature, the real-world problem is assumed to be perfectly specified by (textual) natural language. The proposed framework is novel in the following ways. Unlike the aforementioned literature, the current framework does not rely on the assumption that user requests are structured in a specific format; instead, they can be arbitrary. This framework is specifically tailored to address power scheduling tasks, which has not been tackled in the literature in the context of LLMs, yet. For the first time, the proposed approach exploits the capabilities of LLMs to perform both, modeling and solving power scheduling problems. The aim is not to provide a power scheduling scheme that outperforms existing ones in terms of predefined performance metric, but rather to revolutionize how optimization problems (OPs) are formulated and solved for the sake of generating the recommended power scheduling vector.

To achieve our goal, we propose an LLM multi-agent architecture which allows a human VRQ to be converted to an OP whose solution is a power scheduling vector (Vec). The approach of multiple LLM agents has been developed recently (see e.g., [12]–[15]) and consists in specializing an LLM for a given range of tasks (experts) and associating the LLM agents for a global task. The main contributions of this paper are summarized as follows: • We develop a new **methodology** (based on LLM) to design a scheduling scheme for power management systems, in which the system requirements are acquired from the user through VRQs (Sec. II); • We propose a novel **architecture** (Sec. III) comprising the design of three LLM agents, with the aim to perform user-driven power scheduling, i.e., an intent recognition agents (to identify the best formulation of the mathematical problem given the VRQ from the user), a parameter identification agent (to determine the required parameters for the OP from the VRQ as well as the physical system), and an OP solving agent (to solve the formulated problem through a bank of solving functions in which the LLM agent assists the solver in the initialization phase through particular prompts); • The proposed architecture is partially implemented mainly for Llama3 and evaluated through a thorough **performance analysis** (Sec. IV); • As effective accuracy measure, we construct a **database** of possible user VRQs related to EV charging (EVRQ), with the corresponding request complexity, ground-truth OP/OCP classes, and optimal solutions. This database enables us to evaluate the performance of our framework by comparing the generated results against these benchmarks, using intent recognition accuracy (IRA) and final utility (Sec. IV).

II. Problem Statement

In this work, we develop an LLM-based converter which takes as input a voice/text request from the user (see examples of use cases in Fig. 1) and transforms it into a power consumption scheduling vector (Vec)

$$x = (x_1, x_2, ..., x_T) \tag{1}$$

with $\forall t \in \{1, ..., T\}, 0 \leq x_{\min} \leq x_t \leq x_{\max}$, T being the number of time-slots over which the power is scheduled.

To accommodate the user's demands, the "VRQ2Vec" converter has to initially recognize the user's intent in an accurate way, find the optimum formulation for the corresponding OP, and then solve the latter to generate the recommended power vector that satisfies the requirements. To achieve this goal, we propose a multi-agent architecture [12], as demonstrated in Fig. 2. For the problem formulation, we assume a list of most common OPs; for simplicity, we will refer to optimal control problems (OCPs) as OPs as well. The first stage of the VRQ2Vec framework is an **intent recognition agent (Agent 1)**, which is tasked to identify from the list the most suitable problem that can ideally model the user's intent. The second stage (Parser) is a problem **parameter identification agent (Agent 2)** which role consists in extracting the parameter information of the selected OP. The third stage corresponds to the **OP solver agent (Agent 3)**. Note that LLM capabilities are exploited in this study in three ways. First, we exploit their ability to describe a real world problem as a mathematical problem (which is imposed here to be an OP). Second, we exploit the LLM for OP parameter identification purposes.

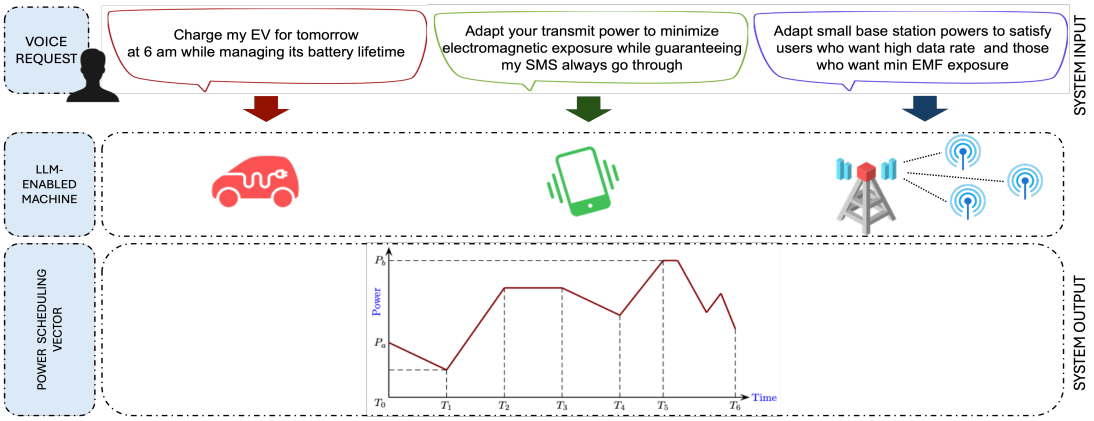

Fig. 1: Use-Cases of the Proposed Intelligent Power Scheduling System

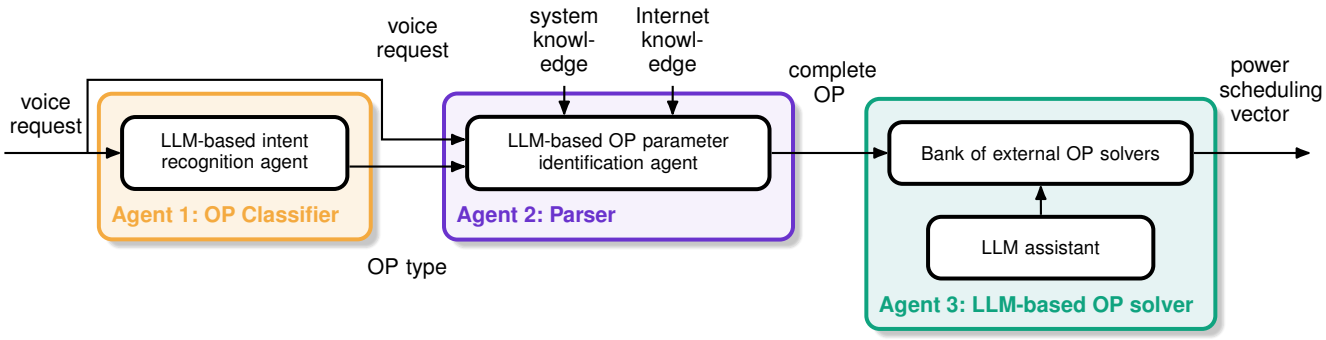

Fig. 2: Proposed multi-agent architecture for a voice request to power scheduling vector converter (VRQ2Vec)

Third, we partially exploit their ability to assist standard OP solvers by allowing the LLM to share their textual knowledge to help better initialize the solver. The performance of the three agents and stages of the proposed VRQ2Vec framework will be assessed through two performance metrics. **Intent Recognition Accuracy (IRA):** assuming the existence of perfect human labeling of every VRQ of a given database into an OP type within a list of OPs, the IRA corresponds to the empirical percentage of OPs properly classified by the LLM-based classifier (Agent 1). **Average relative optimality loss (AROL)**: knowing that misclassification can occur, the AROL measures how suboptimal is the power vector proposed by the chain Agent 1 $\rightarrow$ Agent 2 $\rightarrow$ Agent 3 in average.

III. Multi-Agent Design of the Voice Request to Power Vector Converter

A. Design of the LLM-based Intent Recognition Agent

The role of the LLM-based intent recognition agent (**Agent 1**) is to associate a mathematical problem with a given VRQ from the user. Therefore, it has the role of modeling a speech or text description of a physical problem into equations. Due to the inherent limitations of existing LLMs in describing physical problems as mathematical abstractions, including the latest ChatGPT 4o, we resort to a particular structure of **Agent 1**, in which this agent is designed to function as a classifier of pre-selected OPs. For simplicity and motivated by the literature of power scheduling problems, we define the following set of possible OPs as described in detail in Table I. For instance, the VRQ "You have 24h to charge my EV at 80% while minimizing the cost of charging" can be modeled by a linear program (LP) in which: the vector $(c_1, ..., c_T)$ (see Table I) represents the prices of electricity at time-slots $1, ..., T$; choosing $b = -0.8$, $A = (-1, ..., -1)$ translates that the battery state of charge should be at least 80%. Note that here we assume 6 OPs but more OPs might be assumed without changing the proposed methodology. However, having more OPs does not necessarily imply having a better performance, showing the importance of both choosing the OPs and the number of OPs. Simulations will support this assertion. The intuition behind the existence of an optimal number of OPs to be used within a given set of OPs for classification is similar to the problem of having less robust digital modulation constellations for large constellations. Here, since the LLM recognition capabilities are not perfect, this introduces "noise" whose impact might be higher when the list of possible OPs gets larger.

Agent 1 therefore uses its language processing abilities to classify the VRQ into an OP (or OCP) type. Note that this classification might be performed by a supervised classical neural network (e.g., an MLP) but this will come at the cost of losing the generalizability capability of LLMs. Rather, we consider an LLM that uses a well-designed context as hard

prompting, augmented with function calling abilities. It is designed for the problem of scheduling power for charging an EV but could easily be adapted for the wireless example mentioned in the preceding section. The main LLM we exploit in the performance evaluation part is Llama3 because it is partly open-sourced and because it can be run locally with relatively affordable computational power. To sum up, the design of **Agent 1** therefore comprises selecting an LLM model (Llama 3 in this paper) and utilizing hard-prompting, with a carefully curated list of OPs (Table I) and a well-structured context (Fig. 8). This setup enables the agent to efficiently achieve the task of classifying the VRQs. To evaluate the performance of **Agent 1** in terms of IRA, we have constructed a database of VRQs (EVRQ); more details will be provided in Sec. IV. Note that the LLM's domain knowledge in power scheduling can be further improved through fine-tuning with a specialized dataset, but this is left as an extension of this paper.

B. Design of the LLM-based Parameter Identification Agent

The classifying agent, **Agent 1**, identifies a relevant OP type that mathematically describes the request made by the user. In order for **Agent 3** to be able to numerically solve the corresponding OP, it is necessary first to determine the parameters of the OP. This is where **Agent 2** plays a role: the OP parameter identifier. One can distinguish between three types of parameters. Type 1: parameters that can be extracted from the VRQ (e.g., the time at which the battery should be recharged to a certain level). Type 2: parameters that are pertinent to the physical system (e.g., the maximum power $x_{\max}$). Type 3: common knowledge parameters that can be sourced from the Internet (e.g., a typical value for the ambient temperature). As far as the design of **Agent 2** is concerned, the parameters of Types 2 and 3 are considered as inputs to the agent, whereas Type 1 parameters require the LLM-based agent to exploit its language processing skills to extract the relevant parameters from the VRQ.

To access the parameters determined by the parser, **Agent 2** requires the capability of function calling, i.e., the ability of passing some parameters to another program by using a particular syntax in order to call a real code function. For each function that needs to be called, we provide a description of the required parameters to the agent. For the example of EV charging, the parser must behave as follows: initially, it has to extract the time parameters from the user request and then call a function that will initialize those parameters. A time parameter is either an initial time instant, a final time instant, or a duration that can be explicit (8 a.m.) or implicit (tomorrow morning). It is necessary to extract those parameters first, since most of the other parameters are vectors or matrices, whose size depends on the duration of the scheduling. Then, the parser has to call a solving function that depends on the OP type selected, for which the arguments are the parameters of the OP type. In the case of EV charging, to access external parameters, we allow the agent to create the parameters by using attributes and methods from a smart meter. The smart meter is the interface between the environment (or context) and the parser agent. All these parameters form the complete OP that can be passed to the third agent.

C. Design of the LLM-based OP Solving Agent

A possible approach to solve the OP which describes the power scheduling problem associated with the VRQ is to use a purely LLM-based agent, that is, to ask an LLM such as GPT 4o or Llama3 to solve the OP. This approach is adopted e.g., by OPRO [11] (OPtimization by PROmpting). Assuming that the optimization task can be described in natural language, OPRO proposes a prompt-based framework to leverage LLMs as numerical optimizers. In each optimization step, the LLM generates new solutions from the prompt that contains previously generated solutions with their values, then the new solutions are evaluated and added to the prompt for the next optimization step. Although promising, these approaches are still limited due to the fact that LLMs were not originally designed to solve mathematical equations[2], and may suffer from accuracy issues, which can be problematic when dealing with stringent physical or quality of service (QoS) constraints. Therefore, instead of trying to design LLM-based optimizers which compete with existing numerical solvers, we rather pursue a **coupling** approach in which existing solvers are assisted by an LLM. This allows the exploitation of both the determinism/guarantees offered by existing solvers and the creative problem-solving capabilities of LLMs. **Agent 3** is chosen to be composed of a bank of 6 solvers based on `scipy.optimize`, `cvxpy` and `control` Python libraries: solve_LP (`scipy.optimize.linprog`); solve_QP (`cvxpy.minimize`); solve_CP (`scipy.optimize.minimize`); solve_MM (`scipy.optimize.minimize`); solve_LMT (`scipy.optimize.milp`); solve_LQR (`control.lqr`). The initialization of these solvers are performed by asking an LLM to make the best choice to its knowledge. The values of the parameters which cannot be extracted from the VRQ and the system knowledge are found by using an LLM.

IV. Performance Evaluation

In this section, we evaluate the performance of the proposed LLM-based approach in EV charging applications. Given the wide diversity of OPs encountered in this domain, we divide them into 6 categories, each with a given type of performance metric that users might consider when initiating a charging session. The main objectives of each category can be listed as follows: Charging cost (CC): Reducing the charging cost; Charging time (CT): Minimizing the time to charge the EV to a target level; Environmental impact (EI): Maximizing the use of renewable energy; Power peak (PP): Minimizing the power peak on the electrical installation; Power variations (PV): Minimizing fluctuations in the power supply to the EV charger; Grid Damage (GD): Limiting the potential damage to the distribution grid installation. Each of these performance

[2]As per its version of May 31st 2024, Llama 3 8B cannot reliably solve w/o human assistance first-order equations such as $ax + b = 0$ [21].

metrics is linked to the most suitable OP class that can model the VRQ. To simplify our approach, we assume there is only one OP within each category, as follows:

Performance metric	CC	CT	PP	PV	GD
OP class	LP	LMT	MM	QP	CP

We create knowledge files for each OP class that contains both the description of the problem associated with the performance metrics from the perspective of EV charging, and the generic mathematical description of the OP class independent of EV charging. This comprehensive categorization ensures that we can better instruct LLM agents with prompts for power scheduling in EV charging scenarios.

A. Database and evaluation metric

To evaluate our framework, we generate a variety of requests that capture multiple types of decisions relevant to different EV charging problems. Specifically, our dataset EVRQ includes scenarios such as minimizing charging costs, reducing charging time, maximizing the use of renewable energy, limiting power peaks, reducing power variations, and minimizing grid damage. Additionally, the requests are designed to be either explicit or implicit. Explicit requests are requests that explicitly mention to optimize a particular performance metric, for example: *Charge my EV while minimizing the electricity cost*. However, implicit requests such as *I want my EV to juice up but only when it's financially wise*, specify the performance metric to be assessed with a paraphrase. In total, 800 requests have been generated to compute the IRA, that correspond to 160 requests for each performance metric. To evaluate the performance of our approach, we provide labels for each request, including the ground-truth OP/OCPs class and the optimal solution for the corresponding problem. By comparing these labels with the generated results, the IRA, as defined in Section II, serves as an effective measure of accuracy. In addition, to evaluate the system performance degradation, we consider the average relative optimality loss (AROL), consisting in weighting the optimality loss due to the misclassification by the probability of misclassification, as follows:

$$\mathrm{AROL}_i = \sum_{J \neq I} p_{i \to J} \frac{1}{N} \sum_{n=1}^{N} \frac{f_i\left(x_j^{(n)}\right) - f_i\left(x_i^{(n)}\right)}{f_i\left(x_i^{(n)}\right)} \qquad (2)$$

where i, j are indices for performance metrics with their corresponding OP classes I, J, $p_{i \to J}$ represents the probability that a request of type i is classified as J, N is large and represents the number of random abstract requests to evaluate the average loss for misclassified requests from i to J, $x_i^{(n)}$ represents the optimal power vector of the n-th request for performance metric i, $x_j^{(n)}$ represents the obtained power vector of the n-th request misclassified to J and solved as j, f_i is the cost function of performance metric i. Generating a random abstract request simply consists in randomly selecting a starting time and a duration that will be used to solve the predetermined OPs related to the performance metrics. By taking a large N, we estimate the loss induced by selecting a wrong OP class on average. Then, weighting by the misclassification rates from IRA simulations gives the average relative optimality loss for a particular type of performance metric.

B. Prompt engineering

Basic Prompting: Use of basic prompt and simple mathematical description of optimization and control problems for classification. In this scenario, we aim to evaluate the performance of the LLM model using a straightforward system prompt. The agent is tasked with classifying the request into an OP class from a predefined list. This setup provides the LLM with only the basic prompts that include the names and mathematical forms of the OP classes, without additional contextual information or guidance related to EV charging.

Contextualized Prompting: improving classification using basic prompts and contextualization of optimization and control problems in the context of EV charging. To assess the impact of augmenting the LLM with more detailed knowledge, we extend the basic system prompt from Scenario 1 by appending comprehensive knowledge files. These files provide both textual and mathematical descriptions of typical EV charging problems within each OP class. By enriching the LLM's input with this detailed information, we aim to improve its ability to recognize the appropriate OP class and accurately translate the EV charging request into a canonical form suitable for external solvers.

Error-Informed Prompting: improving classification by analyzing the previous errors. In this scenario, we further refine the system prompt to enhance the classifier's performance. This involves incorporating specific remarks to guide the LLM more effectively. These remarks are mainly obtained from analyzing the mistakes made with the two prompting techniques mentioned above, as well as leveraging expertise in EV charging and optimization to identify descriptions with textual ambiguity and sources of optimization confusion. By regulating the structure and content of the prompts, we aim to streamline the LLM's decision-making process, ensuring more accurate and reliable classification and problem formulation.

These scenarios illustrate the progressive enhancements in prompt engineering techniques, demonstrating how additional knowledge and refined guidance can significantly improve the performance of LLMs in classifying and solving complex OPs in EV charging applications, as shown in the following figures.

C. Simulation results

We implemented the different agents in Python by using the *ollama* library and Llama3 8B as the base model. We set the temperature of the model to 0 to eliminate any randomness, ensuring that each request is consistently classified into the same OP class. Due to the computational constraints of using Llama3 8B, we limited the scope of some simulations by reducing the number of OP classes known to Agent 1 to three.

Fig. 3 illustrates the IRA performance with respect to the different proposed prompting techniques. The obtained results

TABLE I: Description of Optimization (and optimal control) Problem classes considered by Agent 1

Linear Programming (LP)	**Quadratic Programming (QP)**	**Mini-Max Class (MM)**
$\underset{x}{\text{minimize}} \quad c^\top x$ s.t. $\quad Ax \leq b$ $A_{\text{eq}}x = b_{\text{eq}}$ $0 \leq x_{\min} \leq x_t \leq x_{\max}$	$\underset{x}{\text{minimize}} \quad \frac{1}{2}x^\top Qx + c^\top x$ s.t. $\quad Ax \leq b$ $A_{\text{eq}}x = b_{\text{eq}}$ $0 \leq x_{\min} \leq x_t \leq x_{\max}$	$\underset{x}{\text{minimize}} \quad \max_i f_i(x)$ s.t. $\quad Ax \leq b$ $A_{\text{eq}}x = b_{\text{eq}}$ $0 \leq x_{\min} \leq x_t \leq x_{\max}$
Convex Programming (CP)	**Linear Minimum-Time (LMT)**	**Linear Quadratic Regulator (LQR)**
$\underset{x}{\text{minimize}} \quad f(x)$ s.t. $\quad g_i(x) \leq 0, \quad i = 1 \dots m$ $A_{\text{eq}}x = b_{\text{eq}}$ $0 \leq x_{\min} \leq x_t \leq x_{\max}$	$\underset{x}{\text{minimize}} \quad \tau$ s.t. $\quad s_{t+1} = As_t + Bx_t$ $s_0 = s_i, s_\tau = s_f$ $0 \leq x_{\min} \leq x_t \leq x_{\max}$ $s_{\min} \leq s_t \leq s_{\max}$	$\underset{x}{\text{minimize}} \quad \sum_{t=0}^{N-1}(s_t^\top Q s_t + r x_t^2) + s_N^\top Q_f s_N$ given s_0 s.t. $\quad 0 \leq x_{\min} \leq x_t \leq x_{\max}$ $s_{t+1} = As_t + Bx_t$
Notations: x: power scheduling vector; x_t, s_t : power and state at time t; other quantities are parameters.		

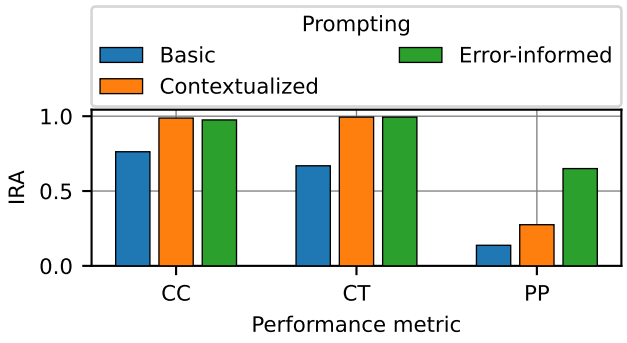

Fig. 3: Influence of context knowledge for Agent 1 in terms of IRA.The evaluation is performed for 3 types of voice requests (CC, CT, and PP). The gains provided by knowledge files are seen to be significant.

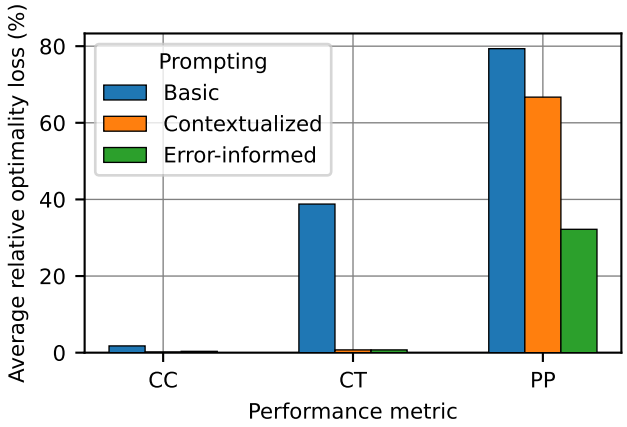

Fig. 4: Influence of context knowledge in terms of optimality loss for the final power scheduling performance metric.

show a clear trend where error-informed prompting achieves the highest IRA across all performance metrics. Meanwhile, by providing the typical EV charging problems specific to each OP class, contextualized prompting shows a moderate improvement over basic prompting, where the latter is application agnostic. To further study the impact of different prompting schemes, in Fig. 4 we demonstrate the average optimality loss using the three different prompting techniques. Confirming the earlier results, error-informed prompting exhibits the lowest average optimality loss, indicating the most accurate problem formulation. These two figures corroborate the advantages of utilizing advanced prompting techniques in the underlying framework to notably enhance the model's ability to classify requests accurately, as well as to introduce improved accuracy performance with respect to the final charging power vector.

Fig. 5 illustrates the IRA performance depending on the number of OP classes provided to the classifier. With only one OP class (LP), the classifier can only handle CC requests, as other requests cannot be resolved using linear programming alone. This explains the absence of the blue bar in the chart for requests outside the CT category, as the IRA is zero when relying only on LP. In addition, the figure highlights that for certain requests, such as those in the PP category, the IRA decreases as the number of OP classes increases. Thus, while a limited number of OP classes restricts the number of treatable requests, adding more OP classes negatively impacts the IRA. It also emphasizes the importance of explicit user requests, as seen with PP requests where providing 3 OPs results in a significant gap in IRA (90% compared to 35% accuracy), between implicit and explicit requests. A similar pattern is observed for GD requests. This indicates that detailed and explicit user requests are essential to achieve higher accuracy in classification. It can be further observed from the figure that classifiers with different sets of OP classes perform very well in CC and CT categories, compared to other ones. This is

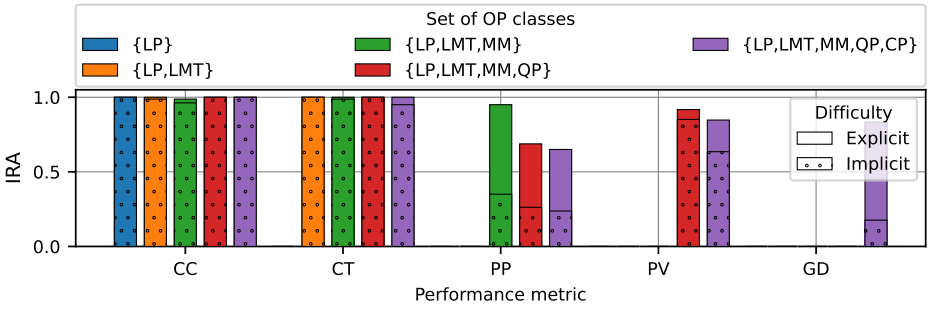

Fig. 5: For 5 different performance metrics (CC,...,GD): influence of the set of selected OPs ($\{\mathrm{LP}, \mathrm{LMT}\}$ means for instance that either Linear Programming or Linear Minimum Time has to be chosen) on its ability to select the most suitable OP class.

motivated by the fact that cost and time are common requests by the users and are relatively easier to be classified, for both explicit and implicit requests.

To further understand other metrics that impact OP classification accuracy, Fig. 6 explores the influence of a set of selected OPs on Agent 1. The IRA is plotted versus different cardinality levels with varying distributions of request categories. The probabilities are defined such that a request belonging to CC, CT, and PP is defined as $\pi, (1-\pi)/2, (1-\pi)/2$, respectively. When requests are uniformly distributed across all categories, using a larger cardinality improves the IRA significantly. This demonstrates that well-designed prompts can handle a broader range of classes effectively, hence, enhancing classification accuracy. When requests are predominantly from one category (with π close to 1), a smaller cardinality can yield better IRA. Additionally, in scenarios where requests are evenly spread, advanced prompts can help the LLM to distinguish between different OPs efficiently, whereas basic prompts are sufficient for an acceptable classification accuracy when requests are very concentrated.

To assess the effect of the selected LLM model in this framework, we tested the error-informed prompting with other LLM models, including GPT-4o, AdvancedGemini (AG), and Llama 3 70B, and examined the classification accuracy on a sample of 30 requests from our database. These samples comprise three distinct sets: 10 requests that were correctly classified by Llama3 8B to verify if other models can perform at least as well as Llama3 8B, 10 requests that were misclassified by Llama3 8B to evaluate if the other models can improve upon Llama3's performance, and 10 EI requests that are not included in the classifier's knowledge base to examine if the selected models can extrapolate and classify requests that are not directly within their known dataset. We compare the performance of Llama3 8B model versus GPT-4o, AdvancedGemini (AG), and Llama 3 70B. The obtained IRA results for different models are outlined in Table. II. The results demonstrate the potential of sophisticated/larger LLMs in improving the OP classification. It further shows that the proposed framework is well-suited for most popular LLMs, emphasizing on the generalizability of this proposed architecture. Also, it suggests that such performance can be further enhanced with better-trained models.

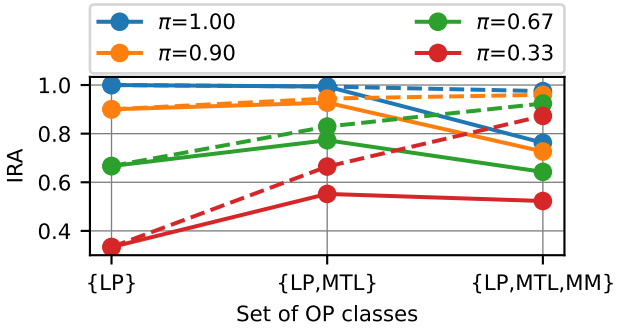

Fig. 6: Impact of the set size of OPs on the classification accuracy of Agent 1. The result is seen to depend on the distribution of the voice request database (which is represented by the probability π of having a request of type CC). Having a larger set of OPs to model the power scheduling problems can negatively affect the recognition accuracy.

Finally, in Fig. 7, we illustrate the patterns of the yielded power vectors for different requests types. For charging cost minimization requests, the charging power is primarily allocated to time slots with lower electricity prices. In contrast, for requests aiming to minimize the charging time, the power vector is more concentrated at the beginning, regardless of other metrics. For power peak minimization requests, the charging power is higher during time slots with lower non-flexible loads to balance the overall electrical load. By adapting to the charging requests, the power vector can be adjusted to meet very diverse requirements effectively.

TABLE II: Influence of the LLM model on accuracy (IRA).

	Llama3 8B	**GPT-4o**	**AG**	**Llama3 70B**
Perfectly Classified	100%	100%	100%	100%
Misclassified	0%	90%	60%	90%
EI	90%	100%	100%	90%

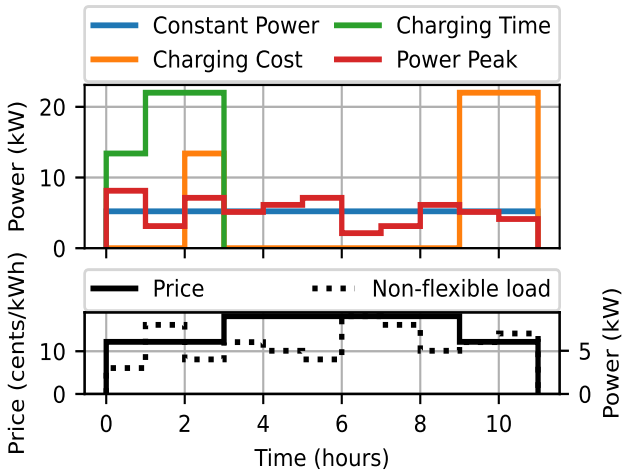

Fig. 7: Final power scheduling vector generated by the proposed agent chain for different classes of voice requests (CC, CT, PP) compared to a basic constant power charging policy. The results constitute a **proof of concept** for the proposed methodology when applied to EV charging: user's voice requests are translated to a very suitable power scheduling policy for very diverse requests (800 VRQs).

V. Conclusion

This paper proposes, for the first time in the literature, how to exploit LLMs to convert an arbitrary VRQ into a power vector. We develop an efficient multi-agent architecture that relies on existing LLM models. We corroborate the efficacy of the proposed methodology by a thorough performance analysis for the EV charging problem. To conduct this analysis, we create a database of VRQs and proposed approaches to handle practical implementation aspects, including OP parameter identification. The corresponding results provide key insights. Having a larger set of possible OPs to model a real-world problem can be detrimental to the model choice problem, creating a tradeoff between accurately modeling a physical problem and the model's ability to correctly recognize the type of problem. The proposed architecture is original and opens a broad avenue for improvements. In particular: the design of the agent which models the physical problem can be improved by having a more diverse set of OPs and by being fine-tuned for wireless/energy networks; the agent which solves the selected OP can be improved by better coupling the capabilities of standard OP solvers with creative LLM-based solvers. We believe that the approach introduced in this paper will be key for humans to interact with wireless/energy networks.

VI. Appendix

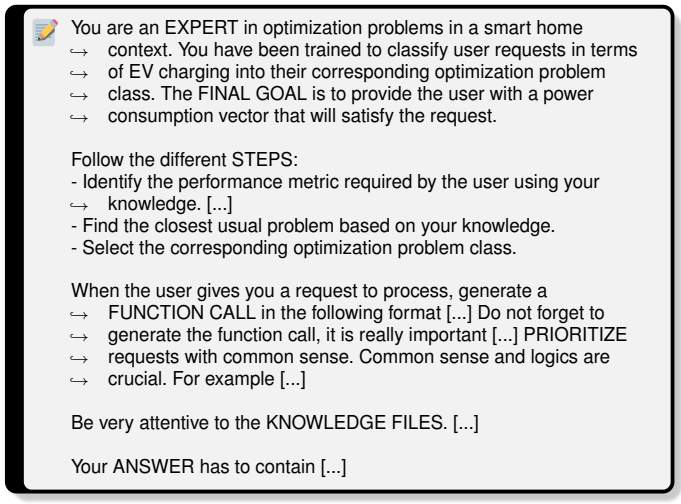

Fig. 8: Extract from a system prompt given to Agent 1.

References

[1] Fattah, H., and Leung, C. (2002). An overview of scheduling algorithms in wireless multimedia networks. IEEE Wireless Comm., 9(5), 76-83.

[2] ElBatt, T., and Ephremides, A. (2004). Joint scheduling and power control for wireless ad hoc networks. IEEE Transactions on Wireless Communications, 3(1), 74-85.

[3] Hohlt, B., Doherty, L., and Brewer, E. (2004, April). Flexible power scheduling for sensor networks. In Proc. of the 3rd International Symposium on Information Processing in Sensor Networks (pp. 205-214).

[4] Radunovic, B., and Le Boudec, J. Y. (2004). Optimal power control, scheduling, and routing in UWB networks. IEEE Journal on Selected Areas in Communications, 22(7), 1252-1270.

[5] Makhadmeh, S. et al. (2019). Optimization methods for power scheduling problems in smart home: Survey. Renewable and Sustainable Energy Reviews, 115, 109362. Elsevier Journal.

[6] Floridi L, Chiriatti M. GPT-3: Its nature, scope, limits, and consequences. Minds and Machines. 2020 Dec;30:681-94.

[7] Touvron H. et al., Llama 2: Open foundation and fine-tuned chat models. arXiv:2307.09288. 2023 Jul 18.

[8] Koncel-Kedziorski, R. et al. (2016, June). MAWPS: A math word problem repository. Proc. of the Conf. of the North American Chap. of the Assoc. for Computational Linguistics: Human Language Technologies (pp. 1152-1157).

[9] Wang, Y., Liu, X., and Shi, S. (2017, September). Deep neural solver for math word problems. In Proceedings of the 2017 conference on empirical methods in natural language processing (pp. 845-854).

[10] He-Yueya, J. et al. (2023). Solving math word problems by combining language models with symbolic solvers. arXiv:2304.09102.

[11] Yang C. et al., Large language models as optimizers. arXiv:2309.03409. 2023 Sep 7.

[12] Talebirad, Y., and Nadiri, A. (2023). Multi-agent collaboration: Harnessing the power of intelligent LLM agents. arXiv:2306.03314.

[13] Task-driven Autonomous Agent Utilizing GPT-4, Pinecone, and LangChain for Diverse Applications. Yohei Nakajima. 2023.

[14] Auto-GPT: An Autonomous GPT-4 Experiment. https://github.com/Significant-Gravitas/Auto-GPT. 2023.

[15] Xiao Z. et al., Chain-of-Experts: When LLMs Meet Complex Operations Research Problems. InThe Twelfth International Conference on Learning Representations 2023 Oct 13.

[16] Larminie J, and Lowry J. Electric vehicle technology explained. John Wiley & Sons; 2012 Sep 17.

[17] Cao Y. et al., An optimized EV charging model considering TOU price and SOC curve. IEEE Trans. on Smart Grid. 2011 Aug 8;3(1):388-93.

[18] AhmadiTeshnizi A, Gao W, Udell M. OptiMUS: Scalable Optimization Modeling with (MI) LP Solvers and Large Language Models. arXiv:2402.10172. 2024 Feb 15.

[19] Li B. et al., Large language models for supply chain optimization. arXiv:2307.03875. 2023 Jul 8.

[20] Tang Z. et al., ORLM: Training Large Language Models for Optimization Modeling. arXiv:2405.17743. 2024 May 28.

[21] "LLaMA 3 tested". Matthew Berman. https://www.youtube.com/watch?v=0AaNT7XO41I&t=166s