The Potential and Limits of LLMs as Evolutionary Operators for Electric Vehicle Charging Scheduling

1st Karim Kamel
IRIMAS UR 7499
Université de Haute-Alsace
F-68100 Mulhouse, France
karim.kamel@uha.fr

2nd Abdennour Azerine
IRIMAS UR 7499
Université de Haute-Alsace
F-68100 Mulhouse, France
abdennour.azerine@uha.fr

3rd Mahmoud Golabi
IRIMAS UR 7499
Université de Haute-Alsace
F-68100 Mulhouse, France
mahmoud.golabi@uha.fr

4th Lhassane Idoumghar
IRIMAS UR 7499
Université de Haute-Alsace
F-68100 Mulhouse, France
lhassane.idoumghar@uha.fr

***Abstract*—Large Language Models (LLMs) are increasingly being explored for optimization and decision-making tasks due to their remarkable reasoning and generalization abilities. However, their suitability for complex combinatorial optimization remains uncertain. This paper investigates whether open-source LLMs can effectively function as evolutionary operators for solving the Electric Vehicle Charging Scheduling Problem (EVCSP), a challenging NP-hard problem that seeks to maximize the number of accepted charging requests under charger and grid capacity constraints. We design a prompt-driven framework where LLMs iteratively evolve populations of candidate schedules, and we further integrate heuristic correction and local search mechanisms to enhance feasibility and performance. Experimental results reveal that, although LLMs can generate feasible solutions, they struggle to maintain population diversity and convergence, leading to premature stagnation and suboptimal results. Conversely, the hybrid LLM–heuristic approach significantly improves solution quality but at the expense of high computational cost. These findings highlight both the potential and the current limitations of LLMs as autonomous optimization engines. These findings offer guidance for future research on enhancing LLM-driven optimization, particularly through improved prompt strategies and domain-adaptive training.**

***Index Terms*—Electric Vehicle Charging Scheduling Problem (EVCSP), Large Language Models (LLMs), Mixed-Integer Linear Programming, Evolutionary Algorithms, Prompt Engineering.**

I. Introduction

The adoption of electric vehicles (EVs) is widely recognized as a key strategy for mitigating emissions in the transportation sector. However, the rapid growth in EV deployment is placing increasing pressure on charging infrastructure [1], necessitating not only its expansion but also efficient operational management. In this context, the Electric Vehicle Charging Scheduling Problem (EVCSP), which focuses on optimizing the allocation of charging resources under operational constraints, has emerged as a central optimization challenge for ensuring efficient utilization of charging stations [2].

The literature on EVCSP encompasses a wide range of approaches, differing in formulations, objectives, and solution techniques. Exact mathematical programming models have been widely used, including nonlinear [3], stochastic [4], and mixed-integer formulations [5] to address operational constraints and uncertainties. While these approaches can provide optimal or near-optimal solutions, their applicability is often limited by scalability issues in large-scale or dynamic settings.

To overcome these limitations, heuristic and metaheuristic approaches have been widely investigated. In [6], heuristic methods are proposed to maximize the number of satisfied charging demands when exact methods become impractical. Jiang et al. [7] develop a heuristic scheduling approach under uncertain renewable generation, integrating an urgency index and simulation-based policy improvement. In [8], sequential and global heuristic strategies are designed for large-scale EV scheduling with cost minimization objectives. Jin et al. [9] address both static and dynamic variants, combining linear programming benchmarks with tailored heuristics. Rigas et al. [10] propose a hybrid framework combining centralized optimization and decentralized heuristic control. Finally, [2] combines mathematical programming with hybrid metaheuristics, including Tabu Search, to optimize charging schedules under operational constraints.

More recently, artificial intelligence (AI) techniques have been explored for optimization. For example, [11] introduces a diffusion-based approach for generating optimal decisions, while [12] combines surrogate modeling with stochastic optimization for multi-variable optimization. In the context of EV charging, [13] formulates the problem as a Constrained Markov Decision Process and proposes a Safe Deep Reinforcement Learning approach to handle uncertainty in arrivals and departures.

In parallel, large language models (LLMs) have demonstrated emerging capabilities in reasoning and optimization-related tasks [14]. Recent studies suggest that LLMs can assist in improving optimization algorithms by generating candidate solutions or guiding search processes [15], [16]. However, existing work has primarily focused on proprietary models, leaving the capabilities of open-source LLMs for combinatorial optimization largely unexplored. This gap is particularly important in practical applications such as EV charging systems, where local deployment is often required due to privacy and cost constraints.

To address this gap, our paper investigates the use of open-source LLMs as evolutionary operators for solving the EVCSP.

979-8-3315-3431-8/25/$31.00 ©2025 IEEE

Unlike traditional evolutionary algorithms, where variation operators are explicitly designed, LLMs offer a data-driven generative mechanism capable of implicitly capturing structural patterns from previously evaluated solutions. This raises a fundamental research question: can LLMs effectively emulate and enhance population-based search dynamics in complex combinatorial optimization problems?

To answer this question, we propose a prompt-driven evolutionary framework in which LLMs iteratively generate and refine candidate solutions based on feedback on solution quality and population diversity. Furthermore, we design hybrid architectures that integrate LLM-based generation with heuristic correction and local search, enabling a controlled analysis of the contribution of each component. Through extensive computational experiments, we evaluate multiple open-source LLMs and provide a detailed empirical assessment of their capabilities and limitations. Our results reveal that, while LLMs can produce feasible and structured solutions, they struggle to maintain diversity and sustain convergence, highlighting fundamental challenges in using generative models as optimization engines. These findings provide new insights into the interaction between language models and evolutionary search, and establish a foundation for future research on LLM-assisted optimization.

The remainder of this paper is organized as follows. Section II defines the EVCSP. Section III presents the proposed methodology. Section IV reports and discusses the computational results, and Section V concludes the paper.

II. Problem definition

This study considers the scheduling of EV charging requests at a station equipped with a limited number of chargers and subject to a global grid capacity constraint. Let $J = \{1, \dots, n\}$ denote the set of charging requests and $M = \{1, \dots, m\}$ the set of available chargers. Each charger $i \in M$ delivers power w_i (kW), while the total power consumption across all active chargers is limited by a global capacity W_g (kW).

Each request $j \in J$ is characterized by an arrival time r_j, a departure time d_j, and an energy requirement e_j. The corresponding charging duration on charger i is given by $p_{ij} = \left\lceil \frac{e_j}{w_i} \right\rceil$, expressed in discrete time slots. The problem consists in assigning each accepted request to a single charger such that charging is non-preemptive and executed entirely within its time window $[r_j, d_j)$. Each charger can serve at most one request at a time, and no temporal overlap is allowed. In addition, at any time, the total power drawn by all active chargers must not exceed W_g.

The objective is to maximize the number of accepted requests while satisfying all operational constraints, yielding a resource-constrained scheduling problem with both temporal and capacity limitations.

III. Methodology: LLM as an Evolutionary Operator

This study proposes a framework for solving the EVCSP in which open-source LLMs act as generative operators within an evolutionary process. Each solution is encoded as a vector of length n, where position j corresponds to demand j, and each element takes a value in $[0, m]$, with 0 indicating rejection and values 1 to m representing assignments to chargers. The LLM iteratively generates and evolves a population of such solutions through structured prompts incorporating the current population, fitness values, and problem parameters.

Two configurations are considered. In **LLM-NoLS**, a heuristic correction is applied after each generation to enforce feasibility, whereas **LLM-LS** further integrates a local search phase to refine solutions. In both cases, candidate solutions are evaluated using a Mixed-Integer Linear Programming (MILP) model, and the resulting performance feedback is used to guide subsequent generations. The process is repeated until a stopping criterion is met, forming a closed-loop search procedure that emulates key principles of evolutionary algorithms.

Algorithm 1: Heuristic Algorithm to Correct Charger Assignments

Input: Assignment array `assignment` of size n
Output: Validated assignment array A, where each element is a valid charger index or 0

$A[j] \leftarrow 0$ for all $j = 1, \dots, n$;
chargers_demands $\leftarrow$ [ ] for $i = 1$ to m;
for *each demand* $j = 1$ *to* n **do**
  if $1 \leq$ `assignment`$[j] \leq m$ **then**
    Append j to *chargers_demands*[`assignment[j]`];
for *each charger* $i = 1$ *to* m **do**
  D_i $\leftarrow$ *chargers_demands*$[i]$;
  Sort *D_i* in ascending order of arrival time $r[j]$;
  $t \leftarrow 0$;
  for *each demand* $j \in$ *D_i* **do**
    if $r[j] \geq t$ **then**
      $A[j] \leftarrow i$;
      $t \leftarrow d[j]$;
return A;

A. Heuristic Algorithm for Assignment correction

A heuristic procedure is used to enforce partial feasibility of charger assignments, as described in Algorithm 1. It ensures that assigned charger indices are valid, that each charger serves at most one demand at a time, and that accepted demands are scheduled within their availability windows.

Given a candidate assignment vector, invalid assignments are first discarded. For each charger, assigned demands are grouped and sorted by arrival time. They are then scheduled sequentially by tracking charger availability: a demand is accepted only if its start time does not overlap with previously scheduled demands; otherwise, it is rejected and assigned 0. The resulting vector A contains feasible assignments in $[1, m]$, with 0 denoting rejected demands.

B. Heuristic-Based Local Search for Assignment Refinement

A heuristic-based local search is applied to improve a corrected assignment, as described in Algorithm 2. Its objective is to reassign previously rejected demands to available chargers without introducing conflicts, using a precomputed conflict graph.

Algorithm 2: Heuristic-Based Local Search to Refine Assignment

Input: Assignment array `assignment` of size n
Output: Improved assignment array

chargers_demands ← [ ] for $i = 1$ to m;
refused_demands ← [ ];
for *each demand* $j = 1$ *to* n **do**
  if $0 \leq$ `assignment`$[j] < m$ **then**
    Append j to the list corresponding to charger `assignment[`j`]`;
  else
    Append j to *refused_demands*;
Extract from `conflict_graph` the rows corresponding to *refused_demands* and store in *conflict_rows*;
for *each* (idx, j) *in enumerate(refused_demands)* **do**
  conflict_j ← row number *idx* of *conflict_rows*;
  random_chargers ← random permutation of {0, …, m-1};
  for *each* i *in random_chargers* **do**
    assigned_demand_indices ← row corresponding to charger i in *chargers_demands*;
    if *assigned_demand_indices is empty* **then**
      assignment[j] ← i;
      append j in *chargers_demands*[i];
      break;
    conflicts ← *conflict_j* values corresponding to *assigned_demand_indices*;
    if *all elements of conflicts are 0* **then**
      assignment[j] ← i;
      append j in *chargers_demands*[i];
      break;
return *assignment*;

The algorithm first constructs, for each charger, the set of demands currently assigned to it, and identifies the set of refused demands resulting from the initial correction step. Then, for each refused demand j, chargers are explored in random order in an attempt to find a feasible assignment. If a charger i has no assigned demands, j is assigned directly. Otherwise, the assignment is accepted only if it does not conflict with any demand already assigned to charger i, where conflicts are defined based on temporal overlap or other constraint violations encoded in the conflict graph. This procedure returns an updated assignment vector of the same size as the input, with an improved allocation of demands to chargers.

C. Mathematical model for energy allocation

The first constraint, which limits the number of available chargers, is addressed by the validation heuristic. However, the energy constraint must still be enforced: at any time slot, the total power supplied by all chargers must not exceed the grid capacity. This constraint is validated through a mathematical model solved using `CPLEX`. Let A denote a potential solution generated by the heuristic. Based on A, an array of accepted charging demands is constructed. Two binary decision variables are introduced:

- $x_j = 1$ if demand j is included in the accepted set; $x_j = 0$ otherwise.
- $y_{j,t} = 1$ if time slot t satisfies $r_j \leq t < d_j$, indicating that demand j is being charged at time t; $y_{j,t} = 0$ otherwise.

$$\max \sum\nolimits_j x_j, \tag{1}$$

$$\sum\nolimits_t y_{j,t} = P_{C_j,j} \cdot x_j \quad \forall j, \tag{2}$$

$$\sum\nolimits_{j \in A,\, t \in [r_j, d_j)} w_{C_j,j} \cdot y_{j,t} \leq W, \quad \forall t \in T. \tag{3}$$

The objective (1) maximizes the number of accepted demands. Constraint (2) ensures that each accepted demand receives its required charging duration $P_{C_j,j}$ on charger C_j. Constraint (3) enforces the grid capacity by limiting, at each time slot t, the total power drawn by all active charging sessions.

D. Prompt Structure

This section describes the prompt design used to interact with the LLM. Two types of prompts are employed: a *system prompt*, which defines the role and task context, and a *user prompt*, which provides the input required for each generation. The system prompt instructs the LLM to act as an *optimization expert in EVCSP*, thereby specifying the expected behavior and objective, while the user prompt supplies the current population for further evolution.

Two prompting strategies were evaluated: (i) embedding both instructions and problem data within the user prompt, and (ii) assigning all instructions to the system prompt while using the user prompt solely to transmit the current population. No significant performance differences were observed, and the latter configuration was adopted for clarity and efficiency.

The prompt content is structured around three complementary components. First, *exploitation* encourages the LLM to analyze the current population and corresponding fitness values to identify and refine promising solution patterns. Second, *exploration* promotes the generation of diverse alternatives to mitigate premature convergence. Third, *strict rules* enforce hard constraints, including the output format, solution size, and admissible value range.

In addition, a dynamic feedback mechanism is incorporated to guide the search process. The LLM receives two metrics: the normalized population diversity, computed as the average pairwise Hamming distance divided by n, and the normalized average fitness $f_{\text{norm}} = f_{\text{current}}/n$, where f_{current} is the mean population fitness. Based on these values, adaptive instructions are appended to the prompt: higher fitness levels ($f_{\text{norm}} > 0.7$) trigger intensified exploitation, while lower values ($f_{\text{norm}} < 0.5$) encourage exploration; similarly, low diversity (below 0.3) prompts the generation of more varied solutions. This mechanism enables an adaptive balance between exploration and exploitation across generations.

To limit prompt size, which is critical for models with constrained token capacity, communication is restricted to a message queue of at most three entries. In practice, the LLM receives a system message and a single assistant message containing the latest population, ensuring both efficiency and clarity.

E. System Integration and Execution Flow

This section describes the workflow of the proposed hybrid framework, which integrates four components: an LLM acting as an evolutionary operator, a heuristic correction procedure, a heuristic local search, and a mathematical evaluation model. At each iteration, the LLM generates a population of candidate solutions from a structured prompt. Due to the generative nature of the LLM, these solutions may violate structural or feasibility constraints, requiring subsequent processing.

The generated population is first preprocessed to ensure structural validity. A text-to-array conversion extracts valid solution representations, while a dimension correction enforces consistency in both solution length n and population size m. Missing solutions are replaced by the best individuals from the previous generation, and inconsistent vectors are either truncated or padded with valid values.

Each candidate solution is then passed through the heuristic correction algorithm to enforce feasibility with respect to charger availability and time window constraints. In the **LLM-LS** configuration, a local search procedure is subsequently applied to refine the corrected solutions, whereas in **LLM-NoLS**, only the correction step is performed.

The resulting population is evaluated using the mathematical model, which enforces energy constraints and computes final fitness values. The evaluated population, together with its performance metrics, is then used to construct the prompt for the next iteration. This closed-loop process is repeated until a stopping criterion is met, and the best solution encountered across all generations is returned. The detailed structures of the system and user prompts, together with the definition of the symbols used within them, are provided below.

- {`sorted_demands`}: List of charging demands sorted by priority.
- {`fitness_eval`}: Average fitness value of the last population.
- {`fitness_feedback_sentence`}: the fitness feedback sentence that is choosed based on the fitness value.
- {`distance`}: Diversity measure of the last population.
- {`distance_feedback_sentence`}: Feedback on diversity that is choosed based on the diversity measure.
- {`population_size`}: Size of the population.
- {`population`}: Last generated population.

System Prompt:
You are an optimization expert in the Electric Vehicle Charging Scheduling Problem (EVCSP). Your task is to propose optimal schedules for the following charging demands sent to a charging station: **{sorted_demands}**. A schedule must be represented as an array of size **n**, where each index corresponds to a demand and each item in the array indicates the charger assigned to that demand. The value can be any integer from 0 to **m**. If a demand is assigned the value **m**, it means that demand is refused. Exploitation: The fitness of your last population was **{fitness_eval}**/1.
{fitness_feedback}.
Exploration: The repetition (diversity) feedback for your last population is **{distance}**/1.
{distance_feedback}
Rules: The population must contain **{population_size}** arrays (solutions). Each array (solution) must be of size **n** and contain only numbers from 0 to **m**. You must generate ONLY the population. Do NOT add text, explanations, code, or any other content.

User Prompt: The last population you generated was: **{population}**

TABLE I
INTERVALS FOR α BASED ON CHARGING TIMES p_{1j}

p_{1j}	$[0.5, 1)$	$[1, 2)$	$[2, 3)$	$[3, 4)$	$[4, 5)$	$[5, 6]$
α	$[0.1, 1]$	$[0.1, 0.9]$	$[0.1, 0.8]$	$[0.1, 0.7]$	$[0.1, 0.6]$	$[0.1, 0.5]$

IV. RESULTS AND DISCUSSION

All experiments were conducted on a workstation equipped with an Intel Xeon W-2295 CPU (3.0 GHz), 128 GB RAM, and 18 cores, running Windows 11. The algorithms were implemented in Python, using the CPLEX API for solving the linear models and the Ollama API for interacting with the LLMs.

A custom dataset was generated to simulate realistic scenarios. The evaluation comprises different instance groups, each containing 10 randomly generated instances, characterized by different numbers of demands, chargers, and grid capacities. Specifically, the number of demands n, chargers m, and grid capacity G were selected from $\{40, 50, 100\}$, $\{24, 27, 30\}$, and $\{75, 100, 125\}$, respectively.

Each instance includes three types of chargers, evenly distributed with power levels of 11, 22, and 43 kW, ensuring a balanced representation of charging capabilities. Arrival times r_j are sampled uniformly from $[0, 0.2n]$, and energy demands e_j from $[5.5, 66]$ kWh. The charging duration on a type-1 charger is given by $p_{1j} = \frac{e_j}{11}$, and the departure time is defined as $d_j = r_j + (1 + \alpha)p_{1j}$, where α is drawn from intervals dependent on p_{1j} (see Table I).

Two methodologies are investigated: (i) an evolutionary algorithm guided solely by LLMs, and (ii) a hybrid framework integrating the evolutionary algorithm with a local search refinement applied to the assignment. Both approaches are compared in terms of solution quality and computational performance. The evolutionary process terminates upon meeting any of the following criteria: (i) a maximum of 200 generations is reached, (ii) the best fitness value shows no improvement over 20 consecutive generations, indicating stagnation, or (iii) the population diversity remains constant for three consecutive generations, defined by a variance in Hamming distance of less than 0.01.

TABLE II
RESULTS WITH $n = 40$ DEMANDS AND $m = 24$ CHARGERS

Inst	LM-NoLS			LM-LS			DS-NoLS			DS-LS			SA
	Obj	T.	G.	Obj	T.	G.	Obj	T.	G.	Obj	T.	G.	Obj
11	28	550.1	19	33	589.1	21	28	4.3	4	33	52.6	21	32
12	26	43.9	9	30	592.8	21	26	25.7	15	30	118.9	21	29
13	25	537.0	18	30	330.9	21	26	71.3	8	30	46.2	21	28
14	27	22.5	5	30	118.7	21	25	29.7	17	30	53.5	21	27
15	24	33.0	7	28	358.4	22	23	33.6	19	27	108.9	21	26
16	25	17.2	4	31	534.1	22	26	25.8	13	31	33.0	21	29
17	24	277.41	11	27	594.07	23	24	12.14	9	27	39.95	23	25
18	28	486.17	12	32	96.15	23	27	26.45	16	32	52.63	23	30
19	27	343.35	23	30	1041.71	23	27	40.85	23	30	121.55	23	29
20	29	242.89	6	33	333.95	24	29	21.62	11	33	69.07	23	31

TABLE III
RESULTS WITH $n = 50$ DEMANDS AND $m = 27$ CHARGERS

Inst	LM-NoLS			LM-LS			DS-NoLS			DS-LS			SA
	Obj	T.	G.	Obj	T.	G.	Obj	T.	G.	Obj	T.	G.	Obj
21	38	533.3	20	44	585.5	23	35	30.9	19	44	28.1	15	42
22	39	577.1	25	44	361.5	23	35	35.9	22	44	15.2	9	42
23	34	576.7	24	42	796.7	23	33	17.2	12	42	92.3	13	40
24	36	563.3	24	46	331.2	23	34	28.9	18	46	31.7	23	45
25	41	556.0	25	47	563.4	24	38	15.9	15	46	44.6	23	45
26	32	567.2	16	42	571.3	21	34	33.4	16	43	51.1	22	40
27	42	204.3	9	48	378.0	13	37	28.3	14	48	11.2	8	46
28	33	336.8	11	41	1197.1	21	33	25.7	13	41	47.0	19	39
29	37	577.6	24	44	998.9	22	36	38.2	21	43	60.8	21	41
30	40	344.7	22	49	246.6	7	40	53.0	23	49	5.3	4	48

TABLE IV
RESULTS WITH $n = 100$ DEMANDS AND $m = 30$ CHARGERS

Inst	LM-NoLS			LM-LS			DS-NoLS			DS-LS			SA
	Obj	T.	G.	Obj	T.	G.	Obj	T.	G.	Obj	T.	G.	Obj
31	65	183.9	21	88	238	21	65	5.2	4	88	134.1	21	85
32	69	88.5	17	90	248.6	21	65	6.7	5	90	148.4	21	88
33	69	63.8	23	87	739.3	21	64	11.2	8	88	208.7	22	84
34	67	132.8	21	94	378.3	22	67	7.5	6	94	189.7	22	91
35	67	100.9	23	90	529.5	21	64	5.7	5	90	107.7	21	86
36	65	125.5	21	85	658.7	22	67	16.9	7	85	256.3	21	82
37	66	378.6	22	87	373.4	21	65	8.2	7	87	247.6	22	83
38	67	92.6	22	84	750.5	22	64	7.5	6	84	88.1	21	81
39	69	225.4	21	95	170.9	21	69	5.1	4	95	77.2	21	91
40	66	482.2	21	88	312.5	22	70	33.3	22	87	72.0	21	83

We tested five open-source LLMs deployed locally: LLaMA 3.1 70B, DeepSeek-R1 32B, Mistral 7B, Qwen 14B, and Phi-3 14B. Due to space constraints, we present detailed results for the two best-performing models, LLaMA (*llama4:16x17b*) and DeepSeek (*deepseek-r1:32b*), which consistently outperformed the others across all instance groups. The remaining three models showed 15-25% lower solution quality and higher failure rates in population generation.

Tables II–IV report instance-level results. The first column indicates the instance number, followed by four blocks corresponding to the tested methods: LM-NoLS and LM-LS (referring to llama4:16x17b without and with local search, respectively), and DS-NoLS and DS-LS (deepseek-r1:32b, likewise). Each block includes three sub-columns: objective value (Obj), execution time in minutes (T.), and the generation at which the stopping condition was met (G.). The final column reports the objective value obtained using the Simulated Annealing (SA) method from [17], serving as a baseline.

In Table II, models without local search consistently yield objective values below those obtained by SA, indicating limited capability to improve solutions. In contrast, incorporating local search systematically produces higher objective values, often surpassing the SA baseline, confirming the effectiveness of the refinement phase. Performance variability across instances is also observed: while several instances yield identical objective values, discrepancies arise in specific cases, such as instance 13 where DeepSeek outperforms LLaMA, and instance 14 where the reverse occurs. Given a common initial population, such inconsistencies are unexpected, particularly as LLaMA generally matches or exceeds DeepSeek due to its larger parameter capacity, suggesting that factors beyond model size influence the search dynamics. Regarding computational time, all models exhibit prohibitively high runtimes; although DeepSeek is generally faster, this is largely attributable to earlier termination, often triggered by stagnation in the generated populations. This behavior points to a reduced ability to sustain exploration and may also reflect difficulties in effectively processing complex prompts that combine problem data, population information, and feedback.

Regarding computational time, all models exhibit impractically high runtimes. DeepSeek is generally faster, but this is largely due to earlier termination, often caused by stagnation in the generated populations. This behavior suggests a limited ability to sustain exploration of the solution space or difficulties in effectively processing complex prompts that combine problem specifications, population data, and feedback.

As reported in Tables III and IV, the performance gap between the LLM-only approach and both the SA and hybrid methods is substantial. Across all tested instances, the hybrid method consistently outperformed SA. In contrast, the LLM-only approach, without local refinement, produced significantly lower-quality solutions, similar to those obtained by other instances. These results reinforce the importance of local search in improving solution quality and reveal current LLMs' limitations in accurately interpreting optimization prompts and exploring diverse solution spaces.

To statistically validate the observed performance trends, Wilcoxon signed-rank tests were conducted at a 0.05 significance level. The results indicate that the LLM-only methods (LLM-NoLS and DS-NoLS) perform significantly worse than the SA baseline ($p < 0.001$), with mean objective deficits of 8.8 and 9.9, respectively. This is further reflected by LLM-NoLS yielding inferior solutions in 27 out of 30 instances.

In contrast, the hybrid methods show a statistically significant improvement over SA. Both LM-LS ($p = 0.023$) and DS-LS ($p = 0.031$) achieve higher mean objective values, corresponding to an average gain of 4.4%, with LM-LS outperforming SA in 21 of 30 instances. A direct comparison between LLaMA and DeepSeek reveals no significant differ-

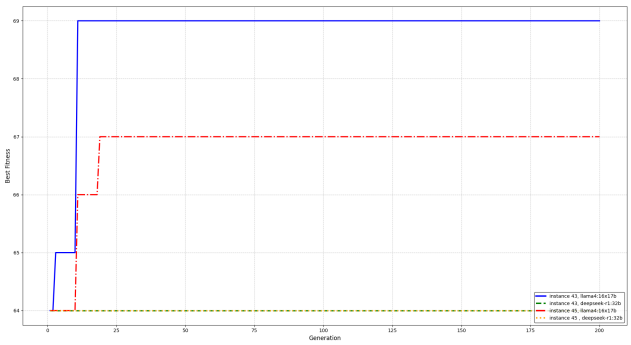

Fig. 1. Fitness convergence of instances 33 and 35.

ence in either configuration ($p > 0.3$), indicating comparable optimization capabilities and favoring DeepSeek as the more practical option due to its lower runtime.

Despite these improvements, the relatively modest effect size combined with a substantial computational overhead (5–240×) limits the practical applicability of the hybrid approach. It is therefore more suitable for (i) offline optimization settings, (ii) problems requiring enhanced solution diversity, or (iii) exploratory studies on LLM-based optimization.

These statistical findings are consistent with the observed convergence behavior. As shown in Figure 1 for instances 33 and 35, the LLaMA-based approach exhibits only a few improvements in early generations and typically stabilizes around generation 20, indicating premature stagnation. While this suggests some ability to improve solution quality, it also reveals limitations in sustaining long-term search. In contrast, DeepSeek shows no improvement across generations, confirming its inability to effectively evolve the population.

V. Conclusion

This work presents an empirical evaluation of Large Language Models (LLMs) as evolutionary operators for the Electric Vehicle Charging Scheduling Problem (EVCSP). Two variants were investigated: a purely LLM-driven evolutionary approach and a hybrid LLM–heuristic framework incorporating local search. While the hybrid approach achieved competitive solution quality, the LLM-only variant frequently stagnated and generated low-diversity populations, highlighting its limited ability to sustain effective population-based search. Moreover, the high computational cost of LLM queries remains a significant practical limitation, restricting scalability compared to classical optimization methods.

These results indicate that, for the considered scheduling problem, current LLMs are not yet reliable as standalone optimization engines. Their performance is hindered by difficulties in maintaining exploration and diversity, as well as by limited integration of feedback across iterations, particularly in structured combinatorial settings.

Nevertheless, this study provides insight into the interaction between LLMs and evolutionary search and identifies several promising research directions. These include the development of adaptive prompting strategies that better encode problem structure, hierarchical or multi-stage prompting schemes separating feasibility and improvement, domain-specific fine-tuning to align model behavior with optimization objectives, and tighter integration of LLMs with symbolic optimization methods. Such advances may help bridge the gap between natural language reasoning and effective algorithmic search in future LLM-assisted optimization frameworks.

References

[1] International Energy Agency. (2025) Global EV outlook 2025. Paris. Licence: CC BY 4.0. [Online]. Available: https://www.iea.org/reports/global-ev-outlook-2025

[2] A. Azerine, A. Oulamara, M. Basset, and L. Idoumghar, "Improved methods for solving the electric vehicle charging scheduling problem to maximize the delive," in *2024 IEEE Congress on Evolutionary Computation (CEC)*. IEEE, 2024, pp. 1–8.

[3] Y. Dahmane, R. Chenouard, M. Ghanes, and M. Alvarado-Ruiz, "Optimized time step for electric vehicle charging optimization considering cost and temperature," *Sustainable Energy, Grids and Networks*, vol. 26, p. 100468, 2021.

[4] Z. Wang, P. Jochem, and W. Fichtner, "A scenario-based stochastic optimization model for charging scheduling of electric vehicles under uncertainties of vehicle availability and charging demand," *Journal of Cleaner Production*, vol. 254, p. 119886, 2020.

[5] O.-B. Salah and A. Oulamara, "Simultaneous electric vehicles scheduling and optimal charging in the business context: case study," in *5th IET Hybrid and Electric Vehicles Conference (HEVC 2014)*. IET, 2014, pp. 6–3.

[6] I. Zaidi, A. Oulamara, L. Idoumghar, and M. Basset, "Maximizing the number of satisfied charging demands of electric vehicles on identical chargers," *Omega*, vol. 127, p. 103106, 2024.

[7] Z. Jiang, Q.-S. Jia, and X. Guan, "On large action space in ev charging scheduling optimization," *Science China Information Sciences*, vol. 65, no. 2, p. 122201, 2022.

[8] O. Sassi and A. Oulamara, "Electric vehicle scheduling and optimal charging problem: complexity, exact and heuristic approaches," *International Journal of Production Research*, vol. 55, no. 2, pp. 519–535, 2017.

[9] C. Jin, J. Tang, and P. Ghosh, "Optimizing electric vehicle charging: A customer's perspective," *IEEE Transactions on vehicular technology*, vol. 62, no. 7, pp. 2919–2927, 2013.

[10] E. S. Rigas, S. D. Ramchurn, N. Bassiliades, and G. Koutitas, "Congestion management for urban ev charging systems," in *2013 IEEE International Conference on Smart Grid Communications (SmartGridComm)*. IEEE, 2013, pp. 121–126.

[11] H. Du, Z. Li, D. Niyato, J. Kang, Z. Xiong, H. Huang, and S. Mao, "Generative ai-aided optimization for ai-generated content (aigc) services in edge networks," *CoRR*, 2023.

[12] B. Wang, B. Xie, J. Xuan, and K. Jiao, "Ai-based optimization of pem fuel cell catalyst layers for maximum power density via data-driven surrogate modeling," *Energy conversion and management*, vol. 205, p. 112460, 2020.

[13] H. Li, Z. Wan, and H. He, "Constrained ev charging scheduling based on safe deep reinforcement learning," *IEEE Transactions on Smart Grid*, vol. 11, no. 3, pp. 2427–2439, 2019.

[14] H. Yu and J. Liu, "Deep insights into automated optimization with large language models and evolutionary algorithms," *arXiv preprint arXiv:2410.20848*, 2024.

[15] C. C. Sartori and C. Blum, "Combinatorial optimization for all: Using llms to aid non-experts in improving optimization algorithms," *arXiv preprint arXiv:2503.10968*, 2025.

[16] G. Ghiani, E. Manni, and S. Zacchino, "Improving adaptability in optimization-based decision support systems through large language models," *IEEE Access*, 2025.

[17] I. Zaidi, A. Oulamara, L. Idoumghar, and M. Basset, "Maximizing the number of satisfied charging demands in electric vehicle charging scheduling problem," in *International Conference on Artificial Evolution (Evolution Artificielle)*. Springer, 2022, pp. 89–102.