An LLM-evolved adaptive operator control mechanism for evolutionary Sudoku solving

Tam Minh Nguyen
Ho Chi Minh City University of Technology (HCMUT), Ho Chi Minh City, Vietnam
Vietnam National University Ho Chi Minh City, Ho Chi Minh City, Vietnam
tam.nguyen272@hcmut.edu.vn

***Abstract*—Evolutionary algorithms have demonstrated strong performance on combinatorial problems such as Sudoku; however, their effectiveness heavily depends on handcrafted local search operators and fixed control heuristics. Designing adaptive operator control mechanisms that generalize across problem instances remains a challenging and largely manual process. This paper proposes an LLM-evolved adaptive operator control framework for Sudoku solving, in which a large language model (LLM) is integrated into the evolutionary loop to evolve state-aware local search control heuristics. Rather than directly constructing solutions or performing search, the LLM generates symbolic heuristic policies that regulate swap selection and mutation pressure based on runtime indicators of the search process. These heuristics are encoded as lightweight control rules and embedded into a local search-based genetic algorithm (LSGA). To ensure robustness and mitigate hallucination effects, all LLM-generated heuristics are subjected to an empirical verification procedure across multiple Sudoku instances, and only validated heuristics are retained through an evolutionary selection mechanism. Experimental results on standard Sudoku benchmarks demonstrate that the proposed method achieves competitive performance compared to state-of-the-art approaches and consistently improves convergence speed relative to the original LSGA.**

***Index Terms*—Combinatorial optimization, large language model, genetic algorithm, Sudoku puzzle**

I. Introduction

Sudoku is a popular logic-based combinatorial puzzle. A standard Sudoku consists of 81 cells arranged in a 9×9 grid, partially filled with given numbers. The objective is to complete the grid such that each row, each column, and each 3×3 sub-grid contains all digits from 1 to 9 exactly once. In recent years, more complex and high-dimensional Sudoku variants have emerged, with grid sizes ranging from 16×16 and 25×25 to even 100×100. Despite its simple and intuitive rules, solving Sudoku is computationally non-trivial and has been proven to be NP-complete [1].

Early research on Sudoku solving primarily employed exact methods. Among them, Knuth's Algorithm X combined with the Dancing Links (DLX) data structure [2] is widely regarded as one of the most effective exact-cover formulations for Sudoku. While such exact approaches guarantee optimal solutions, they suffer from severe scalability limitations. As the puzzle dimension increases, the search space grows exponentially, resulting in prohibitive computational time and rendering exact solvers impractical for large-scale Sudoku instances.

To overcome these limitations, subsequent studies explored heuristic and metaheuristic approaches, including genetic algorithm (GA) [3], hill climbing [4], simulated annealing [5] and various hybrid methods [6], [7]. These approaches relax completeness guarantees in exchange for improved scalability and flexibility. Although metaheuristics have demonstrated promising performance on many Sudoku instances, they often require substantial computational effort, may stagnate in local optima, and sometimes fail to find feasible solutions within acceptable time limits. Moreover, the effectiveness of these methods heavily relies on carefully handcrafted heuristic components. Designing such heuristics typically requires extensive domain expertise and repeated empirical tuning, limiting their generalizability across different Sudoku instances and dimensions.

Recent advancements in artificial intelligence (AI) have renewed interest in solving combinatorial problems such as Sudoku. In particular, large language models (LLMs) have demonstrated remarkable potential in reasoning [8] and combinatorial problem-solving tasks [9]. Nevertheless, existing studies applying LLMs directly to Sudoku remain limited, and naive applications frequently suffer from logical inconsistencies, hallucinated steps, or poor scalability.

This gap is noteworthy because LLMs possess strong capabilities in pattern abstraction, symbolic reasoning, and heuristic synthesis, which are highly relevant to combinatorial optimization. Rather than using LLMs as direct solvers, a more promising direction is to exploit their ability to generate and adapt heuristic knowledge that can guide traditional optimization algorithms. However, effectively integrating LLMs into evolutionary computation while ensuring robustness and avoiding unreliable heuristic behavior remains an open challenge.

In this paper, we propose an LLM-evolved adaptive operator control framework for Sudoku solving. The proposed method integrates LLM into a local search-based genetic algorithm (LSGA) to evolve symbolic control heuristics that adaptively regulate local search behaviors based on runtime search indicators. Instead of constructing solutions or performing explicit search, the LLM operates at the meta-level by generating heuristic policies that guide the evolutionary process. To mitigate hallucination and ensure reliability, all LLM-generated heuristics are subjected to an empirical verification procedure across multiple Sudoku instances, and only validated heuristics

are retained through an evolutionary selection mechanism.

The remainder of this paper is organized as follows. Section II reviews related work on Sudoku solving and evolutionary approaches. Section III describes the proposed LLM-evolved adaptive operator control framework in detail. Section IV presents experimental results and comparative evaluations on standard Sudoku benchmarks. Finally, Section V concludes the paper and outlines directions for future research.

II. Related Work

The charm of Sudoku is that it is easy to learn but difficult to master, which has attracted substantial research interest since its popularization. Numerous methods have been proposed to tackle Sudoku puzzles from different computational perspectives.

One of the most widely used classical approaches is Dancing Links (DLX), a brute-force backtracking algorithm proposed by Donald E. Knuth for solving exact cover problems [2]. In this formulation, Sudoku constraints are encoded as an exact cover matrix, and solutions are obtained via systematic backtracking. Although DLX is highly effective for standard 9×9 puzzles, its computational cost increases dramatically with puzzle size. As a result, brute-force approaches become impractical for high-dimensional Sudoku variants due to excessive runtime requirements.

To address the limitations of brute-force methods, various heuristic-based approaches have been investigated. Jones et al. [4] employed a modified steepest-ascent hill-climbing algorithm, in which an objective function guides the search by always selecting the highest-scoring successor within the neighborhood of the current state. Betar et al. [10] introduced an enhanced hill climbing approach, termed the β-Hill Climbing algorithm, which incorporates random operators to escape local optima. Sevkli and Hamza [11] proposed two variable neighborhood search (VNS) models, namely unfiltered-VNS and filtered-VNS, to improve solution quality and search diversification. While these heuristic methods can perform well under carefully tuned parameters, they remain sensitive to initial conditions and often struggle with the enormous search space of high-dimensional Sudoku puzzles.

Given these challenges, metaheuristic algorithms have been increasingly adopted due to their strong global search capabilities. Lewis [5] presented the first application of simulated annealing to Sudoku solving, demonstrating the potential of probabilistic metaheuristics for this problem. Geem [12] proposed a harmony search algorithm inspired by musical improvisation processes, which has been successfully applied to Sudoku as an evolutionary optimization method. More recently, Wang et al. [6] introduced a local search-based GA (LSGA), which incorporates column-wise and sub-block local search operators to accelerate convergence and improve solution quality. Despite their effectiveness, metaheuristic approaches often suffer from long runtime and slow convergence when applied to large-scale or high-dimensional Sudoku instances.

With the rapid development of AI, researchers have also explored AI-driven approaches for Sudoku solving. Giannoulis et al. [13] proposed a trial-and-error framework in which candidate solutions are iteratively generated by LLMs and validated using verifiers. Their approach integrates imitation learning of basic Sudoku rules with an explicit depth-first search strategy, enabling the solution of harder puzzles with limited guessing. Maiya et al. [14] evaluated the performance of multiple LLMs on solving and explaining 6×6 Sudoku puzzles, showing that LLMs achieve only limited success on higher-dimensional instances.

A more intriguing line of inquiry is to leverage LLMs' capacity to generate, refine, and enhance heuristic information that can inform conventional optimization algorithms, rather than relying on them as direct solvers. However, effectively integrating LLMs into evolutionary computation while ensuring robustness and mitigating unreliable or hallucinated heuristic behavior remains a significant challenge. This research gap directly motivates the proposed method in this paper.

III. The Proposed Method

We propose an LLM-evolved adaptive operator control framework for solving Sudoku puzzles, as illustrated in Fig. 1. The method consists of two tightly coupled components:

1) VeEvo+, a modified version of verification-driven evolutionary framework originally proposed by Zhang et al. [15], which evolves symbolic local search control heuristics with the assistance of LLM.
2) LLM-Guided LSGA, an enhanced version of the LSGA [6], in which swap and mutation operators are dynamically biased by heuristics evolved through VeEvo+.

A. VeEvo+: LLM-Evolved Heuristic Optimization

The VeEvo+ framework evolves a population of heuristic policies through an evolutionary process guided by LLM-generated refinements and empirical verification. The pseudocode of VeEvo+ is shown in Algorithm 1.

1) Heuristic Representation and Population Initialization: VeEvo+ maintains a population of heuristic candidates $\mathcal{H} = \{h_1, h_2, \ldots, h_{N_h}\}$. Each heuristic is represented as a symbolic control policy consisting of: (i) a weight formula identifier f, and (ii) a set of continuous parameters Θ. Formally, a heuristic is defined as $h = \langle f, \Theta \rangle$. A heuristic does not directly modify Sudoku solutions. Instead, it computes position-level weights that bias operator decisions during the evolutionary search process. The initial heuristic population is generated by querying the LLM using randomly sampled training Sudoku instances.

2) Fitness Evaluation: Let $\mathcal{S}_{\text{train}} = \{s_1, \ldots, s_M\}$ denote the training Sudoku set. The fitness of a heuristic h is evaluated by embedding it into the LLM-Guided LSGA and solving each puzzle $s_i \in \mathcal{S}_{\text{train}}$. For each puzzle s_i, let $g_i(h)$ denote the number of generations required to reach a feasible solution, or $g_{\max}$ if convergence is not achieved. The primary fitness objective is the average number of generations to convergence:

$$G(h) = \frac{1}{M} \sum_{i=1}^{M} g_i(h).$$

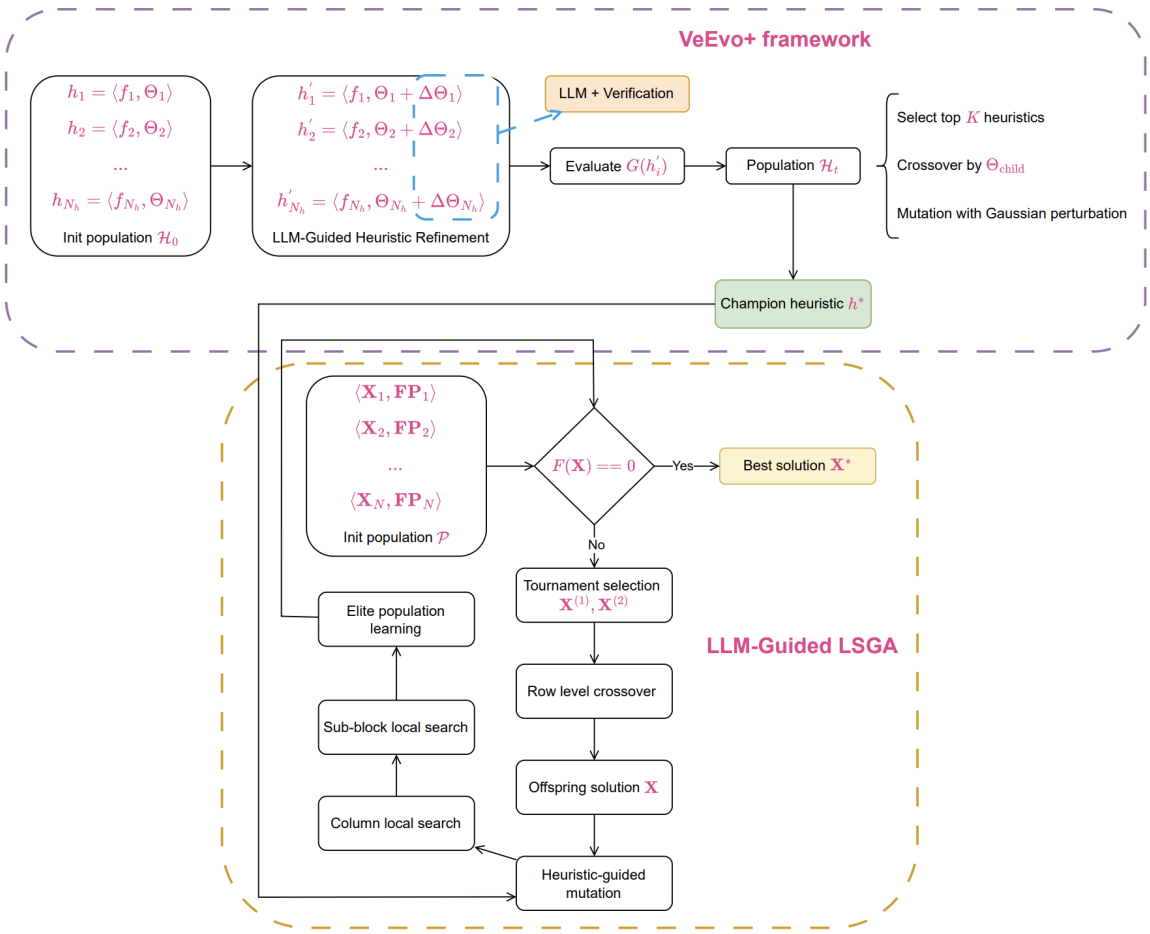

Fig. 1. The overview of our LLM-evolved adaptive operator control framework

In addition, the following auxiliary metrics are recorded for analysis: (i) the average final Sudoku fitness value $F(h)$, (ii) the solve rate $R(h)$, defined as the proportion of puzzles solved within $g_{\max}$ and (iii) the average runtime $T(h)$ measured in wall-clock time. Among these metrics, $G(h)$ is used as the sole optimization objective during heuristic evolution, as it directly reflects convergence speed and operator efficiency.

3) LLM-Guided Heuristic Refinement: Given a heuristic $h = \langle f, \Theta \rangle$, VeEvo+ applies an LLM-based refinement operator $\mathcal{R}_{\text{LLM}}$ that proposes parameter-level modifications while preserving the heuristic structure f. Formally, $\mathcal{R}_{\text{LLM}}(h) \rightarrow \Delta\Theta$, where $\Delta\Theta$ denotes a bounded parameter update. The refined heuristic is constructed as $h' = \langle f, \Theta + \Delta\Theta \rangle$.

4) Empirical Verification of Refinements: Each refined heuristic h' is evaluated on the same training set $\mathcal{S}_{\text{train}}$ used for its parent heuristic h. A verification function $\mathcal{V}$ is defined as:

$$\mathcal{V}(h, h') = \begin{cases} 1, & \text{if } G(h') < G(h), \\ 0, & \text{otherwise.} \end{cases}$$

Only refinements satisfying $\mathcal{V}(h, h') = 1$ are accepted. Rejected refinements are discarded and do not participate in subsequent evolutionary updates.

5) Evolutionary Update: After refinement and verification, the heuristic population is updated using evolutionary operators. Let $\mathcal{H}_t$ denote the population at generation t.

a) Elitist Selection: A subset $\mathcal{H}_t^{\text{elite}} \subset \mathcal{H}_t$ containing the top K heuristics with respect to $G(h)$ is preserved unchanged to the next generation.

b) Crossover: New heuristics are generated by combining parameter vectors of parent heuristics h_{p_1} and h_{p_2}:

$$\Theta_{\text{child}} = \alpha\Theta_{p_1} + (1 - \alpha)\Theta_{p_2}, \quad \alpha \in [0, 1].$$

c) Mutation: To maintain diversity, Gaussian perturbation is applied: $\Theta \leftarrow \Theta + \epsilon, \quad \epsilon \sim \mathcal{N}(0, \sigma^2)$. The next-generation population $\mathcal{H}_{t+1}$ is formed by combining elite heuristics, verified and refined heuristics, and offspring generated through crossover and mutation.

B. LLM-Guided Local Search-Based Genetic Algorithm

The proposed LLM-Guided LSGA extends the classical LSGA framework by introducing adaptive, heuristic-driven operator control, as illustrated in Algorithm 2. The core solution representation and constraint-preserving local search mechanisms are retained to ensure feasibility, while mutation and reinitialization operators are dynamically biased using

Algorithm 1 VeEvo+: Verification-Driven Heuristic Evolution

Input: Training Sudoku set $\mathcal{S}_{\text{train}}$
Input: Population size N_h, elitism size K
Output: Champion heuristic $h^\star$

Initialize heuristic population $\mathcal{H}_0 = \{h_1, \ldots, h_{N_h}\}$ via LLM
Evaluate $G(h)$ for all $h \in \mathcal{H}_0$ using LLM-Guided LSGA
$t \leftarrow 0$
while termination condition not satisfied **do**
    ▷ LLM-guided refinement with empirical verification
    for all $h = \langle f, \Theta \rangle \in \mathcal{H}_t$ **do**
        $\Delta\Theta \leftarrow \mathcal{R}_{\text{LLM}}(h)$
        $h' \leftarrow \langle f, \Theta + \Delta\Theta \rangle$
        Evaluate $G(h')$ on $\mathcal{S}_{\text{train}}$
        if $G(h') < G(h)$ **then**
            $h \leftarrow h'$ ▷ Verified refinement
        end if
    end for
    ▷ Evolutionary population update
    $\mathcal{H}_t^{\text{elite}} \leftarrow \textsc{SelectTop}_K(\mathcal{H}_t)$
    $\mathcal{H}_t^{\text{offspring}} \leftarrow \textsc{Crossover}(\mathcal{H}_t \setminus \mathcal{H}_t^{\text{elite}})$
    $\mathcal{H}_t^{\text{offspring}} \leftarrow \textsc{Mutate}(\mathcal{H}_t^{\text{offspring}})$
    $\mathcal{H}_{t+1} \leftarrow \mathcal{H}_t^{\text{elite}} \cup \mathcal{H}_t^{\text{offspring}}$
    $t \leftarrow t + 1$
end while
$h^\star \leftarrow \arg\min_{h \in \mathcal{H}_t} G(h)$
return $h^\star$

heuristics evolved by VeEvo+. After the VeEvo+ training process converges, the champion heuristic $h^\star = f(\cdot; \Theta^\star)$ is fixed and used to guide all heuristic-based operator decisions in LLM-Guided LSGA.

1) Solution Representation and Initialization: Each individual solution is represented by two matrices:

- Solution Matrix: $\mathbf{X} \in \{1, \ldots, N\}^{N \times N}$, where $X_{i,j}$ denotes the value assigned to row i and column j.
- Fixed-Position Matrix: $\mathbf{FP} \in \{0, 1\}^{N \times N}$, where $FP_{i,j} = 1$ indicates a given (immutable) cell in the Sudoku puzzle, and $FP_{i,j} = 0$ denotes a mutable position.

Initialization is performed row-wise. For each row i, all non-fixed positions are assigned a random permutation of the missing digits, ensuring that row constraints are satisfied by construction for all individuals.

2) Fitness Function: Since row feasibility is always preserved, fitness evaluation focuses exclusively on column and sub-block constraints. The fitness of an individual $\mathbf{X}$ is defined as:

$$F(\mathbf{X}) = \sum_{i=1}^{N} c_i + \sum_{j=1}^{N} s_j,$$

where:

$$c_i = \begin{cases} 1, & \text{if column } i \text{ violates the constraint,} \\ 0, & \text{otherwise,} \end{cases}$$

Algorithm 2 LLM-Guided Local Search-Based Genetic Algorithm (LLM-Guided LSGA)

Input: Sudoku instance ($\mathbf{F}$), population size N, elite size K, crossover probabilities P_{c_1}, P_{c_2}, champion heuristic $h^\star = f(\cdot; \Theta^\star)$
Output: Best solution $\mathbf{X}^\star$

Initialize population $\mathcal{P} = \{(\mathbf{X}_1, \mathbf{FP}_1), \ldots, (\mathbf{X}_N, \mathbf{FP}_N)\}$ using row-wise permutation respecting $\mathbf{F}$
Evaluate fitness $F(\mathbf{X})$ for all $\mathbf{X} \in \mathcal{P}$
Initialize elite archive $\mathcal{E} \leftarrow \emptyset$
while termination condition not satisfied **do**
    $\mathcal{E} \leftarrow \textsc{SelectTop}_K(\mathcal{P})$
    $\mathcal{P}' \leftarrow \emptyset$
    while $|\mathcal{P}'| < N - K$ **do**
        Select parents $\mathbf{X}^{(1)}, \mathbf{X}^{(2)}$ via tournament selection
        if rand $< P_{c_1}$ **then**
            for each row $r = 1, \ldots, N$ **do**
                if rand $< P_{c_2}$ **then**
                    Swap row r between parents
                end if
            end for
        end if
        $\mathbf{X} \leftarrow$ offspring solution
        for each position (i, j) such that $FP_{i,j} = 0$ **do**
            Compute features (v_i, s_i, n_i)
            $w_i \leftarrow h^\star(v_i, s_i, n_i)$
        end for
        Sample mutation position(s) according to $P(i)$
        if high aggregated $h^\star$ score **then**
            Reinitialize row r by permuting values at positions where $FP_{r,j} = 0$
        else
            Apply swap mutation between two positions $(i, j_1), (i, j_2)$ such that $FP_{i,j_1} = FP_{i,j_2} = 0$
        end if
        Apply column local search using swaps restricted to positions with $FP_{i,j} = 0$
        Apply sub-block local search on $\mathbf{X}$
        Evaluate $F(\mathbf{X})$
        Add $\mathbf{X}$ to $\mathcal{P}'$
    end while
    for each worst individual $\mathbf{X}_w$ in $\mathcal{P}'$ **do**
        if rand $< \rho$ **then**
            Replace $\mathbf{X}_w$ with random elite from $\mathcal{E}$
        else
            Reinitialize $\mathbf{X}_w$
        end if
    end for
    $\mathcal{P} \leftarrow \mathcal{E} \cup \mathcal{P}'$
end while
$\mathbf{X}^\star \leftarrow \arg\min_{\mathbf{X} \in \mathcal{P}} F(\mathbf{X})$
return $\mathbf{X}^\star$

$$s_j = \begin{cases} 1, & \text{if sub-block } j \text{ violates the constraint,} \\ 0, & \text{otherwise.} \end{cases}$$

The global optimum corresponds to $F(\mathbf{X}) = 0$, indicating a valid Sudoku solution.

3) Selection and Crossover: Parent individuals are selected using tournament selection. Crossover is performed at the row level to preserve feasibility:

- With probability P_{c_1}, two parent individuals are selected;
- For each row, the row content is exchanged between parents with probability P_{c_2}.

4) Heuristic-Guided Mutation: Unlike the original LSGA, mutation in the proposed framework is guided by heuristics evolved by VeEvo+. For each mutable position i, a heuristic weight is computed as

$$w_i = h^\star(v_i, s_i, n_i),$$

where:

- v_i denotes the number of constraint violations involving position i;
- s_i denotes the stagnation duration since the last fitness improvement;
- n_i denotes the number of conflicts in the local neighborhood of position i;
- $h^\star(\cdot) = f(\cdot; \Theta^\star)$ is the champion heuristic evolved by VeEvo+ and fixed during execution.

Mutation positions are sampled according to the normalized distribution:

$$P(i) = \frac{w_i}{\sum_j w_j}.$$

Two mutation strategies are employed:

- **Swap mutation**: exchanges two non-fixed positions within the same row;
- **Row reinitialization**: randomly redistributes non-fixed values within a row.

The choice between mutation strategies is also probabilistically biased by heuristic-derived weights. Specifically, rows or positions with higher aggregated $h^\star$ scores are more likely to undergo disruptive reinitialization, while lower scores favor swap-based refinement.

5) Column and Sub-block Local Search: Following mutation, the same deterministic local search operators as the original LSGA are also applied to reduce constraint violations:

- Column Local Search: Repeated values within a column are resolved by swapping positions located in the same row, provided that both positions are mutable.
- Sub-block Local Search: Repeated values within a sub-block are resolved using similar row-preserving swap operations.

6) Elite Population Learning: An elite archive is maintained to store the best-performing individuals across generations. At each generation, the worst individual in the population is either replaced by a randomly sampled elite individual, promoting exploitation or reinitialized, promoting exploration, followed by the strategy proposed by Wang et al. [6].

IV. Experimental Results

A. Experimental Setup

1) Datasets and Problem Instances: We evaluate the proposed framework on six benchmark Sudoku datasets from the Sudoku repository [1], covering a wide range of difficulty levels:

- **kaggle** (D0): 100,000 randomly generated easy puzzles with diverse clue distributions.
- **unbiased** (D1): 1 million uniformly sampled minimal puzzles, conditioned on clue count.
- **17_clue** (D2): 49,158 known 17-clue minimal Sudoku puzzles.
- **magictour_top1465** (D3): 1,465 puzzles widely regarded as challenging benchmark instances.
- **forum_hardest_1905** (D4): 1,905 extremely difficult puzzles curated from solver forums.
- **forum_hardest_1106** (D5): 1,106 puzzles selected for their difficulty with respect to heuristic and local search solvers.

Datasets D0–D3 are used during the heuristic synthesis phase, while D4-D5 are reserved exclusively for final evaluation to assess generalization to unseen and extreme difficulty regimes.

2) VeEvo+ Training and Champion Heuristic Generation: Training proceeds in a curriculum-style manner: instances are progressively sampled from D0 to D3, moving from easy to hard puzzles. For each difficulty level, a subset of instances is used to evaluate candidate heuristics, while the remaining instances are held out to prevent overfitting and to monitor generalization during evolution. After convergence, the best-performing heuristic across all difficulty levels is selected as the champion heuristic $h^\star$ and remains fixed during all subsequent LSGA experiments.

3) Algorithm Configuration and Baselines: LLM-Guided LSGA maintains a population of $N = 100$ candidate Sudoku solutions with elite size $K = 10$. Row-wise permutation encoding is used, and local search operators (column and sub-block swaps) are embedded directly within the evolutionary cycle. Crossover probabilities are set to $P_{c_1} = 0.9$ and $P_{c_2} = 0.3$. The champion heuristic $h^\star$, generated by VeEvo+ and refined by the LLM, guides (i) mutation position selection, (ii) adaptive row reinitialization decisions and (iii) prioritization of local search moves. We compare LLM-Guided LSGA against DLX [2], baseline LSGA [6], simulated annealing (SA) [5], hill climbing (HC) [4] and two LLM-based approaches from Giannoulis et al. [13] and Maiya et al. [14]. All stochastic methods are executed for 10 independent runs per instance. Experiments are conducted on a MacBook Pro (M1) with 32GB RAM. Source code of the proposed framework is available at the project repository [2].

B. Overall Performance Comparison

Tables I and II summarize the performance of all methods in terms of efficiency and robustness, measured by the mean

[1] https://github.com/t-dillon/tdoku/tree/master/benchmarks

[2] https://github.com/GiangHoGoVap/VeEvo-LSGA

TABLE I
MEAN NUMBER OF FITNESS EVALUATIONS ACROSS BENCHMARK DATASETS

Method	D0	D1	D2	D3	D4	D5
DLX	12,450	8,920	11,340	48,720	156,890	189,450
LSGA	3,120	2,780	2,940	11,230	36,890	42,560
SA	15,680	11,230	14,560	52,340	178,920	205,670
HC	13,120	12,760	12,940	40,340	134,560	193,870
Giannoulis et al.	9,810	8,990	9,630	35,210	120,300	150,700
Maiya et al.	10,210	9,450	10,100	38,560	128,910	162,410
Ours	**2,480**	**2,210**	**2,460**	**7,020**	**18,940**	**21,670**

TABLE II
SOLVE RATES (%) ACROSS BENCHMARK DATASETS

Method	D0	D1	D2	D3	D4	D5
DLX	100	100	100	95	89	80
LSGA	100	100	100	100	85	75
SA	100	100	100	94	80	70
HC	100	100	100	93	82	71
Giannoulis et al.	100	100	96	93	78	60
Maiya et al.	100	100	95	94	80	62
Ours	**100**	**100**	**100**	**100**	**92**	**80**

number of fitness evaluations and solve rates, respectively, required to reach a valid solution or the best solution found within a fixed time budget.

On easy datasets (D0–D2), all methods achieve near-perfect solve rates, while LLM-Guided LSGA consistently outperforms vanilla LSGA, reducing the number of fitness evaluations by 16–21%. The advantage becomes substantially more pronounced on harder instances. On Dataset D3, our method reduces evaluations by 37.5% while maintaining a 100% solve rate. On the hardest datasets (D4-D5), performance gains increase to 48.7% and 49.1%, respectively. In addition to efficiency improvements, LLM-Guided LSGA demonstrates stronger robustness. While baseline LSGA exhibits noticeable degradation in solve rate (85% on D4 and 75% on D5), our method maintains higher success rates (92% and 80%), outperforming both classical and LLM-based baselines in both efficiency and reliability.

V. CONCLUSION AND FUTURE DIRECTION

This paper introduced an LLM-Guided LSGA for solving Sudoku puzzles, in which adaptive heuristic control is achieved through a learned champion heuristic evolved by the proposed VeEvo+ framework. Unlike conventional metaheuristics that rely on fixed or manually designed operator strategies, our approach synthesizes heuristic policies using a combination of evolutionary search and LLM guidance, and subsequently embeds the resulting heuristic directly into the mutation, reinitialization, and local search components of LSGA.

The experimental results demonstrate that the VeEvo+-generated champion heuristic substantially improves search efficiency, particularly on hard and extreme Sudoku instances. Compared to vanilla LSGA, classical solvers, and recent LLM-based approaches, LLM-Guided LSGA consistently reduces the number of fitness evaluations required to reach a solution.

Several promising directions remain for future work. First, extending VeEvo-guided heuristic synthesis to other combinatorial optimization problems, such as scheduling or timetabling, would test the generality of the approach. Second, incorporating richer state representations or temporal features may enable the evolution of more context-aware heuristics. Finally, integrating explainable AI techniques to analyze and interpret the learned heuristic policies could provide deeper insight into why certain operator strategies are effective, further bridging the gap between human-designed heuristics and data-driven optimization.

ACKNOWLEDGMENT

We acknowledge Ho Chi Minh City University of Technology (HCMUT), VNU-HCM for supporting this study.

REFERENCES

[1] T. Yato and T. Seta, "Complexity and completeness of finding another solution and its application to puzzles," *IEICE transactions on fundamentals of electronics, communications and computer sciences*, vol. 86, no. 5, pp. 1052–1060, 2003.

[2] D. E. Knuth, "Dancing links," *arXiv preprint cs/0011047*, 2000.

[3] T. Mantere and J. Koljonen, "Solving and rating sudoku puzzles with genetic algorithms," in *New Developments in Artificial Intelligence and the Semantic Web, Proceedings of the 12th Finnish Artificial Intelligence Conference STeP*, 2006, pp. 86–92.

[4] S. K. Jones, P. A. Roach, and S. Perkins, "Construction of heuristics for a search-based approach to solving sudoku," in *International Conference on Innovative Techniques and Applications of Artificial Intelligence*. Springer, 2007, pp. 37–49.

[5] R. Lewis, "Metaheuristics can solve sudoku puzzles," *Journal of heuristics*, vol. 13, no. 4, pp. 387–401, 2007.

[6] C. Wang, B. Sun, K.-J. Du, J.-Y. Li, Z.-H. Zhan, S.-W. Jeon, H. Wang, and J. Zhang, "A novel evolutionary algorithm with column and sub-block local search for sudoku puzzles," *IEEE Transactions on Games*, vol. 16, no. 1, pp. 162–172, 2023.

[7] J.-L. Kim and E. Eor, "A genetic algorithm for solving sudoku based on multi-armed bandit selection," *IEEE Transactions on Games*, 2024.

[8] F. Xu, Q. Hao, C. Shao, Z. Zong, Y. Li, J. Wang, Y. Zhang, J. Wang, X. Lan, J. Gong *et al.*, "Toward large reasoning models: A survey of reinforced reasoning with large language models," *Patterns*, vol. 6, no. 10, 2025.

[9] Z. Iklassov, Y. Du, F. Akimov, and M. Takac, "Self-guiding exploration for combinatorial problems," *Advances in Neural Information Processing Systems*, vol. 37, pp. 130 569–130 601, 2024.

[10] M. A. Al-Betar, M. A. Awadallah, A. L. Bolaji, and B. O. Alijla, "β-hill climbing algorithm for sudoku game," in *2017 Palestinian International Conference on Information and Communication Technology (PICICT)*. IEEE, 2017, pp. 84–88.

[11] A. Z. Sevkli and K. A. Hamza, "General variable neighborhood search for solving sudoku puzzles: unfiltered and filtered models," *Soft Computing*, vol. 23, no. 15, pp. 6585–6601, 2019.

[12] Z. W. Geem, "Harmony search algorithm for solving sudoku," in *International conference on knowledge-based and intelligent information and engineering systems*. Springer, 2007, pp. 371–378.

[13] P. Giannoulis, Y. Pantis, and C. Tzamos, "Teaching transformers to solve combinatorial problems through efficient trial & error," *arXiv preprint arXiv:2509.22023*, 2025.

[14] A. Maiya, R. Alghamdi, M. L. Pacheco, A. Trivedi, and F. Somenzi, "Explaining puzzle solutions in natural language: An exploratory study on 6x6 sudoku," in *Findings of the Association for Computational Linguistics: ACL 2025*, 2025, pp. 3002–3009.

[15] J. Zhang, C. Luo, Z. Su, Q. Zhang, Z. Lü, J. Ding, and Y. Jin, "Ns4s: neighborhood search for scheduling problems via large language models," in *Proceedings of the Thirty-Fourth International Joint Conference on Artificial Intelligence*, 2025, pp. 8687–8695.