SEHH: A Search Economics-Based Hyper-Heuristic for Single-Objective Real-Parameter Optimization

Shao-Jhang Wu, Cheng-Hsun Chang, Yu-Xi Liu, and Chun-Wei Tsai
Department of Computer Science and Engineering
National Sun Yat-sen University, Kaohsiung, Taiwan
{b123040048, b123040053, b123040049}@student.nsysu.edu.tw, cwtsai@cse.nsysu.edu.tw

***Abstract*—The basic idea of the hyper-heuristic (HH) algorithm is to integrate multiple search algorithms to leverage their strengths for optimization problems. One of the main challenges in HH design is selecting or generating low-level heuristics (LLHs) effectively within a limited number of evaluations. To address this challenge, this paper proposes an effective HH method based on the machine learning process for solving single-objective real-parameter optimization problems (SOPs). During the offline stage, a set of benchmark functions and an effective metaheuristic algorithm (i.e., search economics) will be used to automatically generate a "good" sequence of LLHs and a set of suitable hyperparameters for them. This good sequence can be regarded as a combination of LLH sets, which will construct a "new" metaheuristic algorithm for solving SOPs during the online stage. The experiments are conducted using the CEC 2024 and CEC 2025 benchmarks to evaluate the performance of the proposed method. The results conclusively demonstrate that this approach significantly outperforms the state-of-the-art algorithms.**

***Index Terms*—hyper-heuristic, metaheuristic, search economics, and optimization problem**

I. Introduction

Optimization plays a fundamental role in modern engineering and scientific research. Some of these optimization tasks can typically be formulated as single-objective real-parameter optimization problems (SOPs), which aim to find the global minimum or maximum of a continuous function. They are particularly challenging when the search space of the objective function is high-dimensional and multimodal [1]. In such complex solution space landscapes, traditional greedy methods such as hill climbing are likely to get trapped in local optima during the convergence process. That is why various metaheuristic algorithms have been proposed as effective alternatives, as most of them include one or more advanced stochastic mechanisms to avoid falling into local optima early in the search and to explore the search space globally. However, Wolpert and Macready in the study [2] pointed out that no single optimization algorithm can outperform all others among all possible problems.

The hybrid metaheuristic algorithm [3] is a prominent way that combines multiple metaheuristic algorithms to integrate their strengths and complement each other. Although designing a good hybrid metaheuristic algorithm manually is intuitive, it typically requires extensive experimentation and some expert experience. The hyper-heuristic (HH) algorithm [4] is a representative branch of hybrid-heuristic algorithms, which uses a high-level heuristic (HLH) to automatically select or generate a heuristic algorithm based on LLHs during the convergence process on the fly. Since most HH algorithms use only one LLH for a period (a couple of iterations) and then use multiple LLHs during the convergence process, an HH algorithm can leverage the strengths of many different (meta)heuristic algorithms. An insufficient allocation of a specific LLH prevents it from fully exploiting the search space, while an excessive allocation may lead to wasted computational resources. Similarly, the ordering of LLHs is crucial, as different LLHs might perform better at different stages of the optimization process. To handle these obstacles regarding resource distribution and sequencing, an effective metaheuristic algorithm (search economics; SE [5]) is adopted as the HLH in this study.

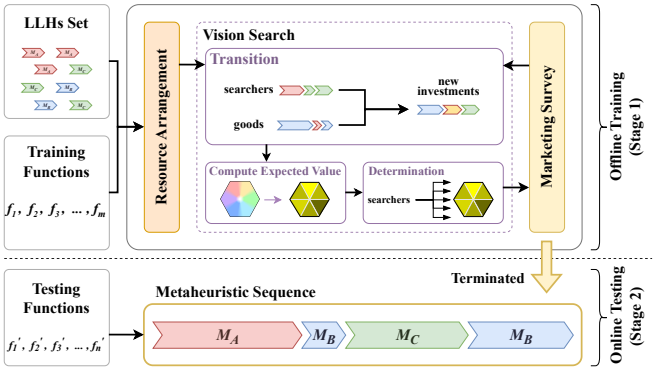

Fig. 1. The architecture of the proposed SEHH.

Since the SE will first divide the solution space into a set of subspaces (i.e., regions) and invest the limited computational resources (i.e., searches) into valuable regions, thereby avoiding wasting computational resources for searching for inferior solutions or getting stuck in particular regions for a long time. That is why we use SE as the HLH of the proposed method, called SE-based HH algorithm (SEHH), to solve SOPs. As illustrated in Fig. 1, SEHH can be divided into two stages: an offline training stage for algorithm generation and an online testing stage for problem solving. During the offline

*All the authors contributed equally to this work.

training stage, SEHH takes in a set of LLHs and training functions as inputs to evolve metaheuristic sequences. This automated generation process relies on the core operations of SE: vision search explores the vast solution space by deploying searchers and sampling goods in promising regions, while the calculation of expected values evaluates the potential of different LLH regions to guide the search process strategically. Once the training is terminated, SEHH outputs the best-evolved metaheuristic sequence, which is applied to solve unseen SOPs in the online testing stage.

The main contributions of this work are summarized as follows:

1) Automated generation of HH algorithms: SEHH provides a flexible offline generation method capable of evolving specialized metaheuristic sequences that fit specific optimization goals. By incorporating a customizable fitness function within the HLH, the proposed method can automatically design algorithms optimized for either solution accuracy or convergence speed.
2) Integration of performance-enhancing mechanisms: To maximize the performance of the generated sequences, SEHH incorporates two key mechanisms:
   - Centralized population control mechanism: Inspired by L-SHADE [6], we implement a population-size reduction strategy that reduces the population size by a fixed ratio whenever the LLH changes, shifting the focus from exploration to exploitation.
   - Joint Optimization of LLH Sequences and Hyperparameters: SEHH jointly optimizes LLHs' sequencing, time allocation, and hyperparameters. This allows SEHH to automatically identify the most effective parameter configurations for each stage of the optimization process, instead of relying on manual parameter tuning.

The remainder of this paper is arranged as follows. Section II begins with related works to our proposed method. Section III starts with the basic idea and followed by the detailed description of the proposed algorithm. The experimental environment and simulation results are given in Section IV. Finally, the conclusions are drawn in Section V.

II. Related Work

A. Problem Definition

An SOP aims to find the global optimum solution for the given objective function within the given set of constraints. The representation of a SOP is defined as follows [1]:

$$\min_{x\in\Omega} f(x), \text{ subject to } x \in \Omega, \tag{1}$$

where $x = (x_1, x_2, \ldots, x_D)$ is a D-dimensional real-valued decision vector, $\Omega = \{x \in \mathbb{R}^D | l_i \leq x_i \leq u_i, i = 1, 2, \ldots, D\}$ is the feasible solution space, $f(x) : \mathbb{R}^D \to \mathbb{R}$ is the objective function, l_i and u_i denote the lower and upper bounds of the i-th decision variable, respectively. The goal of the optimization algorithm is to identify a global optimum $x^* \in \Omega$ such that $f(x^*) \leq f(x)$ for all feasible solutions $x \in \Omega$.

B. From Metaheuristic to Hyper-Heuristic for SOPs

Metaheuristic algorithms have gained widespread adoption in solving SOPs in recent years. Some of these, such as genetic algorithm (GA) [7], differential evolution (DE) [8], and particle swarm optimization (PSO) [9] have been widely adopted for various optimization tasks. Among the numerous metaheuristics, DE and its variants have demonstrated excellent performance in solving SOPs. In 2014, Tanabe and Fukunaga proposed a variant of DE equipped with the linear population size reduction (LPSR) method [6], which dynamically reduces the population size during the optimization process to enhance convergence speed and solution quality. Inheriting the LPSR strategy, [10] proposed L-SRTDE, which uses success rate-based adaptation method to dynamically adjust the control parameters of DE. By integrating multiple mechanisms, the algorithm can adopt different strategies in different steps of the search to balance exploration and exploitation, thereby enhancing the overall performance of the algorithm.

Hybrid metaheuristic algorithms have gained attention for their ability to integrate two or more search algorithms to combine their advantages [3]. To overcome the manual designing issue, HH algorithms [4] have been proposed to automate the design of hybrid metaheuristics. Burke et al. [11] proposed a more unified definition, in which HH algorithms can be classified into two main parts: heuristic selection and heuristic generation. Heuristic selection aims to choose the best existing LLH from the candidate pool, while heuristic generation focuses on creating new LLHs by combining the components of existing LLHs. An interesting example of selecting a lower-level heuristic algorithm during the convergence of a hyper-heuristic algorithm can be found in [12], which consists of an explorative operator pool and an exploitative operator pool as low-level components to balance exploration and exploitation. As for heuristic generation, one example is the LLM evolutionary algorithm (LLaMEA) [13] framework. Apart from selection-based hybrid metaheuristics, LLaMEA has also been explored to automate the design of hybrid algorithms. In their work, large language model (LLM) is employed as an HLH to generate selection and mutation algorithms of the DE algorithm.

III. The Proposed Algorithm

A. Basic Idea

In the proposed method, SE [5] will be adopted as the HLH to construct sequences of LLHs as candidate solutions (e.g., individuals) composed of a set of segments, each of which represents an LLH with two types of adjustable parts:

1) Algorithm hyperparameters: Such as the mutation rate in GA [7] or the memory size in L-SHADE [6].
2) Computational budget: The execution time proportion in the entire sequence.

The objective of the HLH is to find an optimal sequence of LLH segments that achieves the best performance on SOPs. Unlike traditional methods that may pass only the best solution found in the previous LLH to the next LLH, SEHH passes the

previous state to the next LLH segment in the sequence. This mechanism allows the subsequent LLH to continue the search from the solution found by the preceding LLH. Furthermore, to effectively explore the combinatorial space of sequences, SE divides the search space into distinct regions. In the proposed SEHH, the region is defined as the major metaheuristic in a sequence, as shown in Fig. 2. This definition of regions enables SE to avoid allocating excessive computational resources to only a few LLHs and improves the likelihood of finding better HHs in other regions. In addition, the population reduction mechanism inspired by L-SHADE [6] is also adopted in SEHH. The population size is dynamically reduced during the switching process between LLHs. In this way, SEHH accelerates convergence in the later stages while retaining solution diversity in the early stages.

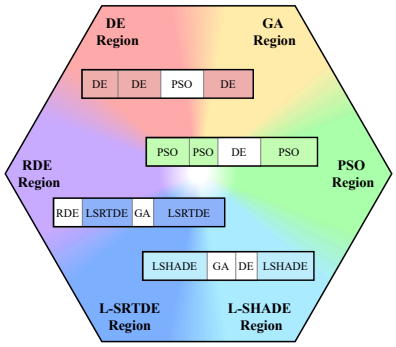

Fig. 2. The regions of the proposed SEHH.

B. Search Economics-Based Hyper-Heuristic (SEHH)

To simplify the description of the proposed SEHH algorithm, the following notations are defined and used throughout this paper. For some of the notations that have been used in SE [5], these notations are reserved for consistency while others are newly defined for SEHH.

- h the number of LLHs, which is equal to the number of regions.
- n the number of searchers.
- ω the number of sample solutions in each region.
- r set of regions; that is, $r = \{r_1, r_2, \ldots, r_h\}$, where r_k is the k-th region and the number of regions equals h.
- s_j^i set of searchers (i.e., solution), where s_j^i is the j-th segment of the i-th searcher which is one of the LLHs.
- m_j^i set of goods (i.e., sampled solutions) in all regions; that is, $m_j^i = \{m_1^1, \ldots, m_\omega^1, \ldots, m_1^h, \ldots, m_\omega^h\}$, where m_j^i is the j-th sampled solution from region r_i and ω is the number of sample solutions from each region.
- v_j^i set of investments (i.e., probe solutions) of the searcher s^i; that is, $v_j^i = \{v_{j1}^i, v_{j2}^i, \ldots, v_{jm}^i\}$, where $v_{jk}^i = s^i \otimes m_k^j$ and $\otimes$ denotes the transition operation.
- r_j^b the best solution found so far in region r_j.
- e_j the expected value for region r_j.
- t_j^a the numbers of times j-th region has been explored by searchers.
- t_j^b the numbers of times j-th region hasn't been explored by searchers.

SEHH mainly preserves the original structure of SE [5] while modifying the internal operations to fit the HH generation task. The algorithm contains the Initialization(), ResourceArrangement(), VisionSearch(), and MarketingSurvey() operators.

C. Definition of the Sequence

The metaheuristic sequence is defined as an ordered sequence of LLHs selected from the LLH set, along with their hyperparameters and time weights. Formally, a metaheuristic sequence S can be represented as $S = \langle(\ell_i,\ \theta_i,\ \tau_i)\rangle_{i=1}^{\beta}$, where $\ell_i \in L$ is the i-th LLH, θ_i denotes the hyperparameters associated with the ℓ_i, τ_i represents the time weight of the ℓ_i, and β is the length of the sequence. Each ℓ_i is allocated $(\tau_i / \sum_{j=1}^{\beta} \tau_j) \cdot \xi_{max}$ evaluations, where ξ_{max} is the total number of evaluations assigned to S. The major metaheuristic of sequence S is defined as the metaheuristic with the largest time weight sum in S, which is used to determine the region of S in SEHH. In the very beginning of the execution of S, the solutions in the first LLH ℓ_1 are generated randomly. At each subsequent LLH ℓ_i $(i > 1)$, where the population size is N_i, the best solution found so far in ℓ_{i-1} is always included in the initial solutions of ℓ_i and the remaining $N_i - 1$ solutions are selected from the final solutions of ℓ_{i-1}. The population size $\langle N_i \rangle_{i=1}^{L}$ is an arithmetic sequence decreasing with a predefined common difference in order to enhance the convergence ability of the metaheuristic sequence. For some LLHs that have a population reduction mechanism (e.g., L-SHADE), their hyperparameters of final population size will be set as the initial population size of the next LLH. Some algorithms incorporate mechanisms that learn during the search process, such as adaptive control parameters or an external archive. These states will be preserved and passed to the next LLH, which supports these mechanisms to avoid learning from scratch.

D. Resource Arrangement

After initializing the parameters in Initialization(), the RandomAssignRegion() function will create a solution within a specific region. It assigns a stochastic solution to each m_j^i within region r_i, which means that the LLH corresponding to region r_i has the largest time weight sum in this solution. Furthermore, the searchers are evenly distributed across regions, with each region receiving approximately the same number of searchers. In this paper, the i-th searcher is assigned to region $r_{(i \bmod h)+1}$.

E. Vision Search

Vision search is the core operator of SEHH, which is responsible for generating new solutions to explore the solution space. There are three main steps in the VisionSearch() operator, which are Transition(), ComputeExpectedValue(), and Determination() in order. As shown in Algorithm 1, the

Transition() function employs crossover and mutation operations, inspired by GA [7], to create probe solutions v based on searchers s and goods m. First, it applies SequenceCrossover() which is one-point crossover operation on s^i and m_k^j, to exchange the segments between the two solutions. Then, SegmentMutate() is applied, and it is composed of three operations:

1) Segment change: Each segment has a small probability of being reinitialized stochastically.
2) Segment swap: Each sequence has a small probability of randomly selecting two segments and swapping their positions.
3) Segment reversal: Each sequence has a small probability of randomly selecting two segments and reversing the order of the segments between them.

Algorithm 1: Transition()

Input: searchers set s and goods set m
Output: probe solutions set v

$v = \phi$;
for $i = 1$ ***to*** n **do**
  for $j = 1$ ***to*** h **do**
    for $k = 1$ ***to*** $|m^j|$ **do**
      $s' =$ SequenceCrossover(s^i, m_k^j);
      $s' =$ SegmentMutate(s');
      $s' =$ ArgumentMutate(s');
      $r' =$ GetRegion(s');
      $v_{r'}^i = v_{r'}^i \cup \{s'\}$;
    end
  end
  for $j = 1$ ***to*** h **do**
    ProbSolutionNormalization(v_j^i);
  end
end

Segment swap and segment reversal are applied to the entire sequence, whereas segment change is applied to individual segments. Although segment-level mutations can help solutions escape local optima, they may disrupt solution structures on a large scale. To address this issue, ArgumentMutate() was introduced to perform argument-level mutation only on the hyperparameters of the LLHs in the sequence. Every segment has a small probability of randomly reinitializing its hyperparameters. Segment mutation and argument mutation methods are both used in SEHH. The former dominates in the early stage to encourage exploration, while the latter is emphasized in the later stage for convergence. After crossover and mutation, the major LLH of the new solution s' may not be the same as m_k^j. Thus, GetRegion() is used to determine the region r' of s' based on its major LLH and place it in the corresponding probe solutions set $v_{r'}^i$. There is one more issue that needs to be addressed. Since each $v_{r'}^i$ may contain a different number of probe solutions, ProbSolutionNormalization() is used to ensure the number of probe solutions is in a reasonable range.

Before discussing how the expected values are computed, it is necessary to define the fitness of a solution. In SEHH, the function value of f optimized by metaheuristic sequence S in the i-th iteration is denoted $G_i(S, f)$. Note that $G_i(S, f)$ is a random variable because the metaheuristic algorithm is stochastic, so it is evaluated L times to reduce the influence of randomness. Due to the fact that convergence speed is also an important performance criterion for metaheuristic sequences, the proposed method uses $R_i(S, f) = E_i(S, f)/\xi_{max}$ to measure the convergence speed of S, where $E_i(S, f)$ is the number of evaluations for S if it reaches the global minimum of f in the i-th iteration, otherwise it is set to ξ_{max}. The definition of fitness Γ for S on F is given in Eq. (2), where η is a constant that determines the influence of convergence speed on fitness, and ψ is the index of the median of Eq. (3) sequence $\langle u_i \rangle$. In Eq. (3), the definitions of $\bar{G}_i(S, F)$ and $\bar{R}_i(S, F)$ are the means of $G_i(S, f)$ and $R_i(S, f)$ over all $f \in F$, respectively. To compute the median in sequence $\langle u_i \rangle$, a predefined relative ordering is required. Given i and j, where $i \neq j$, we say that u_i is ordered before u_j if and only if $\bar{G}_i(S, F) < \bar{G}_j(S, F)$ or $\bar{G}_i(S, F) = \bar{G}_j(S, F)$ and $\bar{R}_i(S, F) < \bar{R}_j(S, F)$.

$$\Gamma(S, F) = \bar{G}_\psi(S, F) + \eta \cdot \bar{R}_\psi(S, F), \tag{2}$$

$$\langle u_i \rangle_{i=1}^L = \left\langle \begin{pmatrix} \bar{G}_i(S, F) \\ \bar{R}_i(S, F) \end{pmatrix} \right\rangle_{i=1}^L. \tag{3}$$

With the probe solutions v and the definition of fitness function, the expected value can be computed as follows. The ComputeExpectedValue() function calculates the expected value e for each searcher based on the probe solutions v, the goods set m, and each region's information t^a and t^b. The expected value e_j for the region r_j is defined as $e_j = \mathcal{T}_j \cdot \mathcal{M}_j$, which is different from the original SE [5], where the expected value is calculated only based on the status of investments and the proportion of the objective value in each region. The original calculation method which computes the objective value of every searcher is discarded in SEHH. Because the value is obtained from all of the probe solutions set v which costs us lots of computational resources. The remaining two factors $\mathcal{T}_j$ and $\mathcal{M}_j$ are defined as follows:

1) The investment status factor of the j-th region is defined by $\mathcal{T}_j = t_j^a / t_j^b$, which is used to record the amount of investments in region r_j.
2) The objective value proportion factor of the j-th region is defined by $\mathcal{M}_j = \sum_{k=1}^{h} \sum_{i=1}^{\omega} \Gamma(m_i^k, F) / \Gamma(r_j^b, F)$, which represents the current performance of region r_j compared to other regions.

In the final step of vision search, the Determination() function uses tournament selection method to determine the region where each searcher will move to based on the expected value e. Each searcher s^i randomly samples N^t regions and selects the one with the highest expected value as its destination.

F. Marketing Survey

For the final marketing survey procedure, the functionalities remain similar to the original SE [5]. The marketing survey operation is responsible for updating the information of solutions and recalculating the investment status t^a and t^b for each region. Another important new feature added in SEHH in this procedure is to dynamically adjust the mutation rate for the HLH based on market information. If the top k_{best} solutions have not improved for a specific number of generations, the mutation rate is increased by a small value until it reaches the upper bound. Conversely, if any of the top k_{best} solutions improves, the mutation rate is restored to its initial value.

IV. The Experimental Results

A. Experimental Environment and Parameter Settings

The experiment was divided into two parts: the training stage (to generate the LLH sequence) and the testing stage (to evaluate the performance of the trained sequence). Both parts were carried out on a server operating on Ubuntu 20.04.3 LTS with Intel(R) Core(TM) i9-10900 CPU 2.80GHz and 64GB of RAM. The algorithms were implemented in C++ and compiled using GCC 9.4.0 with the -O3 optimization flag.

1) Training Stage: To avoid overfitting during training, we used the CEC 2014 benchmark [14] with 30-dimensional functions as the training set. The population size for LLH was initially set to 600 and reduced to a minimum of 10 using the population size reduction strategy. The total computational budget for the HLH was set to 30000 sequence evaluations. This means SEHH generated and evaluated 30000 distinct candidate sequences during the training process. For each candidate sequence, the number of evaluations for sequence ξ_{max} was set to 300000, the iteration number L for one sequence on F was set to 5. It is important to note that sequence evaluations count the number of times a candidate sequence is evaluated by a searcher, excluding the function evaluations consumed by the LLH during the assessment process. The SEHH used 10 searchers exploring $h = 6$ distinct LLH regions ($r = \{r_1, r_2, \ldots, r_h\}$): PSO [9], GA [7], DE [8], L-SHADE [6], L-SRTDE [10], and RDE [15]. In SEHH, each type of LLH is associated with specific hyperparameters. Each region r_k was initialized with $\omega = 5$ random samples ($m^k = \{m_1^k, \ldots, m_\omega^k\}$). The investment history for each region, represented as t_k^a (times explored) and t_k^b (times unexplored), was initialized to 0 and 1, respectively. The number of samples in each region is dynamically maintained between 2 and 8 samples to ensure diversity. The length of the sequence was set to $\beta = 3$. For the transition operators, crossover rate was set to 0.9. The mutation rate was initialized at 0.2 and increased by 0.05, up to a maximum of 0.4 if no improvement was observed in the top k_{top} solutions for 6 consecutive generations, where k_{top} was set to 3. The determination section used a tournament size N^t of 3. Targeting different optimization goals, two distinct convergence speed influence factors η were set. One with η set to 0 focuses on achieving the best solution quality, the other with η set to 4 emphasizes rapid convergence.

2) Testing Stage: After the offline training, SEHH generated an optimal sequence adapted for the problem landscape. Two different sequences were generated for different objectives: one prioritizes the quality of the solution and the other emphasizes the convergence speed. These two sequences are the subject for the subsequent evaluation. The configurations of the two sequences are shown in Fig. 3. With η set to 0, the first sequence, denoted as SEHH, is optimized for achieving the best solution quality. With η set to 4, the second sequence, denoted as SEHH-Fast, focuses on rapid convergence. L-SRTDE consists of memory size (N_{mem}), the perturbation control parameters of the crossover rate (σ_{CR}) and scaling factor (σ_F). RDE includes a population size control factor (P_{size}), a discrete selection mode (Sel) that controls whether rank-based selection pressure is used, a binary switch for enabling constraint-control parameters (G_{flag}), and a binary switch for enabling the EB hybrid mechanism (EB_{flag}). The two sequences consist of three stages, each employing different LLHs and parameter settings. Both of sequences appear to follow a similar time distribution pattern, only differing slightly in the beginning stage and the parameter settings of each LLH.

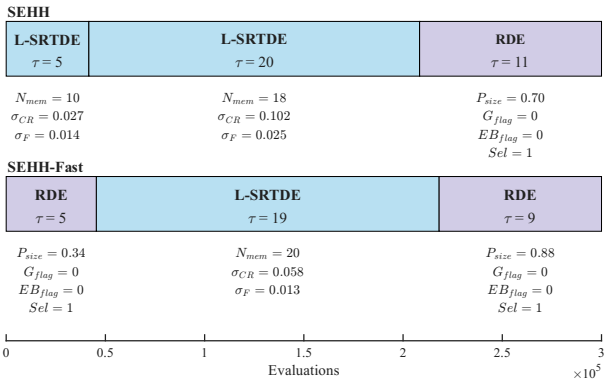

Fig. 3. Configuration of the Trained Optimal Sequences.

The testing protocol follows the CEC 2024 [16] and CEC 2025 [17] competition rules. The search range for all variables is fixed within $[-100, 100]^D$. The maximum number of function evaluations is set to $10000 \times D$, where D is the problem dimension. For each problem instance, the algorithms are executed for 25 independent runs with a maximum computational resource of $10000 \times D$ function evaluations. Any error value smaller than 10^{-8} is treated as zero.

B. Simulation Results

To evaluate the performance of SEHH and SEHH-Fast, we compared the proposed method with the state-of-the-art algorithms in CEC 2024 and CEC 2025, including BlockEA [18], IEACOP [19], RDEx [20], RDE [15], mLSHADE_RL [21], L-SRTDE [10], and jSOa [22]. The results of the compared algorithms were obtained from CEC 2024 and 2025 competitions [23], [24] for the comparisons in this study. The experimental results were analyzed from three perspectives:

solution accuracy, overall efficiency (combining accuracy and convergence speed), and component contribution.

1) Wilcoxon Rank-Sum Test Analysis: To assess the solution accuracy of the proposed methods, Wilcoxon rank-sum test [25] was conducted to identify significant performance differences between SEHH and other competitors. The significance level α was set at 0.01. The testing was performed on the final error values obtained from 25 independent runs for each algorithm on each problem, and TABLE I presents the median results. The symbols '+', '-', and '=' indicate that SEHH is significantly better than, significantly worse than, or statistically similar to the competitor, respectively.

TABLE I
SUMMARY OF WILCOXON RANK-SUM TEST RESULTS ($\alpha = 0.01$)

Competitor	+	=	-
SEHH-Fast	8	20	1
BlockEA	19	3	7
IEACOP	28	0	1
RDEx	3	24	2
RDE	18	10	1
mLSHADE_RL	22	6	1
L-SRTDE	7	19	3
jSOa	23	5	1

Note: The target algorithm is SEHH

The results demonstrate that SEHH outperforms all competitors across the benchmark functions. Only RDEx shows comparable performance, where SEHH performs significantly better in 3 functions, similar in 24, and worse in 2. Also, the comparison between SEHH and SEHH-Fast shows that SEHH has a clear advantage in solution accuracy due to the different design objectives of the two algorithms.

2) U-Score Analysis: The performance evaluation strictly follows the U-Score (rank sum) proposed in the CEC 2024 [16] and CEC 2025 [17] competitions. Unlike traditional metrics that evaluate speed and accuracy separately, the U-Score takes both into consideration. For a specific problem, all 25 runs across all algorithms are pooled and ranked. A run is considered successful if it reaches the target error value (TGT_EV) within the maximum number of function evaluations (Max_FEs). Successful runs are strictly ranked better than unsuccessful ones and are ordered based on the number of function evaluations taken to reach TGT_EV. Unsuccessful runs are ranked based on their final error values, with lower errors receiving better scores. CEC 2024 sets target error value TGT_EV as the median of final error values from all algorithms, while CEC 2025 uses the mean.

Under CEC 2024 rules, the U-Score rank sum results are shown in Fig. 4. SEHH-Fast achieves the lowest rank sum of 75, outperforming SEHH (rank sum: 88), the CEC 2025 champion RDEx (rank sum: 93), and the CEC 2024 champion L-SRTDE (rank sum: 116). Under CEC 2025 rules, the U-Score rank sum results are shown in Fig. 5. The dominance of SEHH-Fast continues under the mean-based criterion, achieving an outstanding rank sum of 62. The standard SEHH records a rank sum of 90, exhibiting better results than RDEx, which has a rank sum of 106. Although SEHH demonstrated superior solution accuracy in the Wilcoxon rank-sum test (as shown in TABLE I), its U-Score is impacted by its slower convergence rate compared to the fast variant. This indicates that while SEHH is more likely to find the global optimum, SEHH-Fast is optimized to reach acceptable solutions more rapidly.

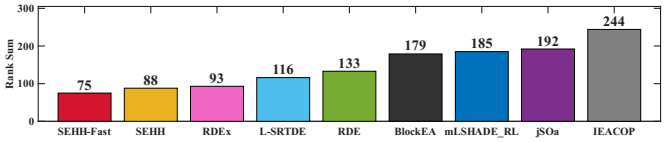

Fig. 4. CEC 2024 U-score rank sum results

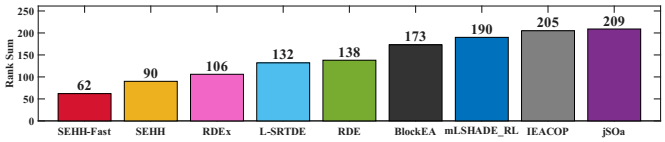

Fig. 5. CEC 2025 U-score rank sum results

3) Ablation Study: To validate the contribution of each component in SEHH, we conducted an ablation study comparing the proposed SEHH with four distinct variants. The experimental settings for all variants were the same as those in the main experiment (25 runs on 29 functions), but the performance was evaluated by using the rank sum of mean errors to assess the final accuracy of each configuration. This metric ranks algorithms based on their average final error on each function (rank 1 represents the best accuracy) and sums these ranks across all problems.

The complete sequence using "SEHH" defined in Fig. 3 is denoted as "SEHH (Proposed)". The first variant, "SEHH-NoMem", tests the impact of the state preservation and information passing mechanism between LLH segments by only passing the best solution found so far to the next segment, without retaining any other state information. The second variant, "SEHH-NoPSR", evaluates the effect of the population size reduction mechanism by resetting the population size to its initial value when switching between LLH segments. The last two variants, "Single-RDE" and "Single-LSRTDE", assess the importance of hybridization by using only a single LLH (either L-SRTDE or RDE) throughout the entire process, with their hyperparameters set according to the trained sequence's first segment and third segment, respectively. The results of the ablation study are summarized in TABLE II.

The results indicate that every component of SEHH contributes positively to its overall performance. Notably, removing the state preservation and information passing mechanism (SEHH-NoMem) leads to the most significant performance degradation. Additionally, the hybrid LLH structure plays a crucial role, as using a single LLH (either RDE or L-SRTDE) results in a performance drop. This shows the effectiveness of combining multiple LLHs to leverage their strengths. The result also highlights the importance of the population size reduction mechanism in enhancing overall performance.

TABLE II
ABLATION STUDY RESULTS BASED ON RANK SUM OF MEAN ERRORS

Algorithm Variant	Rank Sum	Diff.
SEHH (Proposed)	**55.5**	**-**
SEHH-NoMem	142.0	+86.5
SEHH-NoPSR	64.0	+8.5
Single-RDE	100.0	+44.5
Single-LSRTDE	73.5	+18.0

Note: "Diff." indicates the increase in rank sum compared to the proposed method.

V. CONCLUSION

In this paper, we proposed SEHH, an algorithmic method that used SE as HLH to automatically generate metaheuristic sequences based on LLHs for SOP. The experimental results show that SEHH is able to achieve better performance compared to state-of-the-art algorithms on the CEC 2024 and CEC 2025 benchmarks. The comparison between SEHH and SEHH-Fast also demonstrates the adaptability of the proposed method through its customizable fitness function design. SEHH can be flexibly tuned to meet diverse requirements, whether prioritizing solution accuracy or convergence speed. The ablation study confirmed that the LLH sequences, memory passing and population reduction mechanism are crucial for this success, and also proved that hybridizing multiple metaheuristics generates better results than relying on a single algorithm. Future work will focus on extending the SEHH to constrained and multi-objective optimization problems to further validate its effectiveness in different optimization scenarios.

VI. ACKNOWLEDGMENT

This work was supported in part by the National Science and Technology Council of Taiwan, R.O.C., under Contract NSTC112-2628-E-110-001-MY3, NSTC114-2634-F-110-001-MBK, and NSTC114-2221-E-110-023-MY3. Also, the authors used Gemini 3.1 Pro for checking the grammar and polishing the language of the paper.

REFERENCES

[1] G. Wu, R. Mallipeddi, and P. N. Suganthan, "Problem Definitions and Evaluation Criteria for the CEC 2017 Competition on Constrained Real-Parameter Optimization," National University of Defense Technology, Kyungpook National University and Nanyang Technological University, Tech. Rep., 2017.

[2] D. Wolpert and W. Macready, "No free lunch theorems for optimization," *IEEE Transactions on Evolutionary Computation*, vol. 1, no. 1, pp. 67–82, 1997.

[3] C. Blum, J. Puchinger, G. R. Raidl, and A. Roli, "Hybrid metaheuristics in combinatorial optimization: A survey," *Applied Soft Computing*, vol. 11, no. 6, pp. 4135–4151, 2011.

[4] E. K. Burke, G. Kendall, J. Newall, E. Hart, P. Ross, and S. Schulenburg, "Hyper-heuristics: An emerging direction in modern search technology," in *Handbook of Metaheuristics*, F. Glover and G. A. Kochenberger, Eds. Boston, MA: Springer US, 2003, pp. 457–474.

[5] C.-W. Tsai, "Search Economics: A Solution Space and Computing Resource Aware Search Method," in *Proceedings of 2015 IEEE International Conference on Systems, Man, and Cybernetics*, 2015, pp. 2555–2560.

[6] R. Tanabe and A. S. Fukunaga, "Improving the search performance of SHADE using linear population size reduction," in *Proceedings of IEEE Congress on Evolutionary Computation*, 2014, pp. 1658–1665.

[7] J. H. Holland, "Genetic Algorithms," *Scientific American*, vol. 267, no. 1, pp. 66–73, 1992.

[8] R. Storn and K. Price, "Differential Evolution – A Simple and Efficient Heuristic for global Optimization over Continuous Spaces," *Journal of Global Optimization*, vol. 11, no. 4, pp. 341–359, 1997.

[9] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proceedings of International Conference on Neural Networks*, vol. 4, 1995, pp. 1942–1948.

[10] V. Stanovov and E. Semenkin, "Success Rate-based Adaptive Differential Evolution L-SRTDE for CEC 2024 Competition," in *Proceedings of IEEE Congress on Evolutionary Computation*, 2024, pp. 1–8.

[11] E. K. Burke, M. Hyde, G. Kendall, G. Ochoa, E. Özcan, and J. R. Woodward, "A Classification of Hyper-heuristic Approaches," in *Handbook of Metaheuristics*, M. Gendreau and J.-Y. Potvin, Eds. Boston, MA: Springer US, 2010, pp. 449–468.

[12] R. Zhong and J. Yu, "A novel evolutionary status guided hyper-heuristic algorithm for continuous optimization," *Cluster Computing*, vol. 27, no. 9, pp. 12 209–12 238, 2024.

[13] N. van Stein and T. Bäck, "LLaMEA: A Large Language Model Evolutionary Algorithm for Automatically Generating Metaheuristics," *IEEE Transactions on Evolutionary Computation*, vol. 29, no. 2, pp. 331–345, 2025.

[14] J. Liang, B. Qu, and P. Suganthan, "Problem Definitions and Evaluation Criteria for the CEC 2014 Special Session and Competition on Single Objective Real-Parameter Numerical Optimization," Zhengzhou University, Zhongyuan University of Technology, and Nanyang Technological University, Tech. Rep. 2, 2013.

[15] S. Tao, R. Zhao, K. Wang, and S. Gao, "An Efficient Reconstructed Differential Evolution Variant by Some of the Current State-of-the-art Strategies for Solving Single Objective Bound Constrained Problems," *arXiv, 2404.16280*, 2024.

[16] K. Qiao, X. Wen, X. Ban, P. Chen, K. Price, P. Suganthan, J. Liang, G. Wu, and C. Yue, "Evaluation criteria for the CEC 2024 competition and special session on numerical optimization considering accuracy and speed," Zhengzhou University, Central South University, Henan Institute of Technology, and Qatar University, Tech. Rep., 2023.

[17] X. Ban, H. Lin, K. Qiao, X. Wen, P. Chen, K. Price, C. Yue, J. Liang, and P. Suganthan, "Evaluation criteria for CEC 2025 competition and special session on constrained single and multiobjective optimization considering accuracy and speed," Zhengzhou University, Henan Institute of Technology, and Qatar University, Tech. Rep., 2024.

[18] X. Qi, Z. Yang, S. Zhu, H. Geng, S. Yang, and Y. Tian, "Report for Block Evolutionary Algorithm," Tech. Rep., 2024, https://github.com/QiXuhong520/Algorithm-report.

[19] A. Tangherloni, V. Coelho, F. M. Buffa, and P. Cazzaniga, "A Modified EACOP Implementation for Real-Parameter Single Objective Optimization Problems," in *Proceedings of IEEE Congress on Evolutionary Computation*, 2024, pp. 1–8.

[20] S. Tao, Y. Yang, R. Zhao, K. Wang, S. Liu, and S. Gao, "RDEx: An effectiveness-driven hybrid Single-Objective optimizer that adaptively selects and combines multiple operators and strategies," Tech. Rep., 2025, https://github.com/SichenTao/IEEE-CEC-2025-Competition-RDEx.

[21] D. Chauhan, A. Trivedi, and Shivani, "A Multi-operator Ensemble LSHADE with Restart and Local Search Mechanisms for Single-objective Optimization," *arXiv, 2409.15994*, 2024.

[22] P. Bujok, "Progressive Archive in Adaptive jSO Algorithm," Tech. Rep., 2024, https://github.com/PetBuj/jSOa/blob/main/jSOaGitHub.pdf.

[23] K. Qiao, X. Ban, P. Chen, K. V. Price, P. N. Suganthan, X. Wen, J. Liang, G. Wu, and C. Yue, "Performance comparison of CEC 2024 competition entries on numerical optimization considering accuracy and speed," Zhengzhou University and Qatar University, Tech. Rep., 2024, https://github.com/P-N-Suganthan/2024-CEC.

[24] ——, "Performance comparison of CEC 2025 competition entries on numerical optimization considering accuracy and speed," Zhengzhou University and Qatar University, Tech. Rep., 2025, https://github.com/P-N-Suganthan/2025-CEC.

[25] H. B. Mann and D. R. Whitney, "On a Test of Whether one of Two Random Variables is Stochastically Larger than the Other," *The Annals of Mathematical Statistics*, vol. 18, no. 1, pp. 50–60, 1947.