COMPLEX SYSTEM MODELING AND SIMULATION
ISSN 2096-9929 02/06 pp 91–108
Volume 1, Number 2, June 2021
DOI: 10.23919/CSMS.2021.0010

A Novel Cooperative Multi-Stage Hyper-Heuristic for Combination Optimization Problems

Fuqing Zhao*, Shilu Di, Jie Cao, Jianxin Tang, and Jonrinaldi

Abstract: A hyper-heuristic algorithm is a general solution framework that adaptively selects the optimizer to address complex problems. A classical hyper-heuristic framework consists of two levels, including the high-level heuristic and a set of low-level heuristics. The low-level heuristics to be used in the optimization process are chosen by the high-level tactics in the hyper-heuristic. In this study, a Cooperative Multi-Stage Hyper-Heuristic (CMS-HH) algorithm is proposed to address certain combinatorial optimization problems. In the CMS-HH, a genetic algorithm is introduced to perturb the initial solution to increase the diversity of the solution. In the search phase, an online learning mechanism based on the multi-armed bandits and relay hybridization technology are proposed to improve the quality of the solution. In addition, a multi-point search is introduced to cooperatively search with a single-point search when the state of the solution does not change in continuous time. The performance of the CMS-HH algorithm is assessed in six specific combinatorial optimization problems, including Boolean satisfiability problems, one-dimensional packing problems, permutation flow-shop scheduling problems, personnel scheduling problems, traveling salesman problems, and vehicle routing problems. The experimental results demonstrate the efficiency and significance of the proposed CMS-HH algorithm.

Key words: hyper-heuristic algorithm; Multi-Armed Bandits (MAB); relay hybridization technology; combinatorial optimization

1 Introduction

Combinatorial Optimization Problems (COPs) widely exist in actual applications, including flight itineraries, scheduling, economic management, transportation, and logistics management[1]. The applications in these domains promote the rapid development of enterprises and the national economy.

The objective of COPs is to search for the optimal solution from the feasible solution set of combinatorial problems. COPs are normally difficult to be solved because of a widespread and heavily constrained search space. Numerous COPs are considered as the NP-hard problems, which are difficult to be addressed with exact methods. Classical mathematical methods were initially used by researchers to address COPs. Nevertheless, the performance of the mathematical methods is limited by the problem scales. In contrast, meta-heuristics have been used to solve COPs because this method can reasonably find a feasible solution within an acceptable timeframe[2–4]. The Swarm Intelligence (SI) algorithms, which are used to address continuous optimization problems or COPs, are widely studied and effective to address large-scale problems[5,6]. The SI algorithms are

• Fuqing Zhao, Shilu Di, Jie Cao, and Jianxin Tang are with School of Computer and Communication, Lanzhou University of Technology, Lanzhou 730050, China. E-mail: Fzhao2000@hotmail.com; 1373590969@qq.com; caoj2976016@qq.com; 582971672@qq.com.
• Jonrinaldi is with the Department of Industrial Engineering, Universitas Andalas, Padang 25163, Indonesia. E-mail: jonrinaldi@eng.unand.ac.id.
* To whom correspondence should be addressed.
Manuscript received: 2021-03-26; revised: 2021-04-29; accepted: 2021-05-11

inspired by the laws of human intelligence and social or natural phenomena in biological groups. Representative SI algorithms include the Genetic Algorithm (GA)[7,8], Particle Swarm Optimization algorithm (PSO)[9], and Ant Colony optimization algorithm (ACO)[10].

Studies of the SI algorithm have been sustained for decades. The SI algorithms have achieved significant success in distributed flow-shop scheduling[11–13]. A Discrete Water Wave Optimization algorithm (DWWO) was proposed for the blocking flow-shop scheduling problem. The multi-constraint blocking flow-shop scheduling optimization method, DWWO, first uses a heuristic method and disturbance process to initialize the population. Then, the two-step refraction process is used to search the solution space effectively, and the path reconnection mechanism is integrated into the refraction process. Finally, a variable neighborhood search algorithm is implemented to enhance the local mining ability of DWWO[14]. A Discrete Fruit fly Optimization Algorithm (DFOA) was proposed for the distributed blocking flow-shop scheduling problem. Three stages are included in a DFOA. In the initialization stage, the central position of all populations is initialized by the construction heuristic method. In the smell-based foraging stage, a neighborhood structure based on insertion is used to guide the algorithm for global search. In the vision-based foraging stage, a local search algorithm is used to strengthen the local search mining capacity of the scope. At the same time, the simulated annealing-like receiving mechanism is used to prevent the algorithm from falling into a local optimum[15]. A Multi-Objective Discrete Invasive Weed Optimization (MODIWO) algorithm was proposed to address the multi-objective blocking flow-shop scheduling problem with a due date. In MODIWO, the initial population is first constructed using certain heuristic methods. Then, the amounts of seeds produced by each individual are determined by a breeding strategy based on a reference line. At the same time, a spatial diffusion model based on sliding insertion is used to spread all the seeds to the whole solution space. In addition, a self-tuning phase in its main framework is added to promote the local mining ability of the algorithm. Finally, the next-generation population is generated through the competition exclusion mechanism based on Pareto[16]. The Discrete Pigeon-Inspired Optimization algorithm (DPIO) that uses the metropolis acceptance criterion was proposed to address Traveling Salesman Problems (TSP)[17]. The effectiveness of the DPIO is verified by numerous experimental results. The optimal foraging algorithm was proposed to address multi-objective Permutation Flow-shop Scheduling Problems (PFSP)[18]. In this algorithm, a PFSP model that contains four objectives is established. These four objectives include the make-span, total tardiness, energy consumption cost, and inventory holding cost. A variant of Vehicle Routing Problems (VRP) that considers carbon emissions was proposed in fresh food e-commerce[19]. A variable neighborhood search approach is introduced to address the VRP. An Improved Artificial Bee Colony algorithm (IABC) was proposed to address VRP with time windows (VRPTW)[20]. In the IABC, two problem-specific lemmas are derived to address the cross-synchronization problem. Furthermore, a local search static, which is based on variable length, is introduced to promote the exploitation ability. The effectiveness of the IABC is verified by a statistical analysis of 55 instances.

Most of the optimization methods are custom-tailored to specific problems. Usually, the tailored heuristic methods depend on problem-specific knowledge. Therefore, the specific methods do not always perform well when applied to other problems without significant modification. On the basis of the above shortcomings, the operations of specific problems are encapsulated as Low-Level Heuristics (LLHs). The High-Level Heuristics (HLHs) are used to control the selection of LLHs. The proposed algorithm is not used to optimize a single problem, but to optimize multiple problems without making major parameter adjustments. A classical hyper-heuristic framework consists of a control layer and a set of LLHs. A hyper-heuristic is used to explore a solution space composed of a given set of LLHs. The LLH is applied in solutions directly and a new solution is created subsequently[21]. The pivotal motivation behind hyper-heuristic research is to enhance the level of generality of solution methods for computational search problems[22]. One of the motivations of this paper is to use one algorithm to optimize multiple problems. Another purpose is to integrate the advantages of several heuristic components into a hyper-heuristic framework. In the classical hyper-heuristic framework, the domain barrier is used to logically separate the problem domain from the control layer[23]. Early work on hyper-heuristic focuses on the research of selection hyper-heuristic algorithms, which consist of the heuristic selection mechanism and acceptance criteria. In the iterative process, a suitable heuristic is chosen from a set of LLHs and applied to

the incumbent solution. Then, the candidate solution is judged whether to be accepted according to the acceptance criteria[24]. The LLHs are certain problem-specific operators that are diverse for different problem domains[25]. The meta-heuristics and hyper-heuristics are used as LLHs with the development of hyper heuristics. The size of an LLH set is controlled by specific approaches that are relay hybridization or tabu search. Because each instance has a different landscape, the components of HLHs have a great effect on the performance of hyper-heuristics[26]. Therefore, interest is growing in designing a novel heuristic selection mechanism or developing different acceptance criteria. A good high-level heuristic design requires that an appropriate LLH is selected at any particular point according to the current state of the solution, and a good design of acceptance criteria guides the search process toward an optimistic region[27].

Most of the hyper-heuristic algorithms published are selection hyper-heuristics. The LLHs, which are categorized as construction heuristics and perturbation heuristics, are operators related to specific problems. The high-level heuristic intelligently selects an LLH that is appropriate for the state of the solution. The heuristic selection mechanism and acceptance criteria are two crucial components of hyper-heuristics. For example, the Monte Carlo Tree Search (MCTS) was introduced to explore a search space that is modeled as a tree[28]. In the tree, each LLH is considered as a node. A few steps of the MCTS are performed to update the binary tree. The Monte Carlo acceptance criterion, which is introduced to the hyper-heuristics, is used to accept the novel solution. The Gene Expression Programming (GEP) algorithm was introduced to generate HLHs[29]. In the framework, a population formed by HLHs was used to evolve. Each high-level heuristic is divided into the heuristic selection mechanism and acceptance criteria according to specific rules. The roulette wheel tactic is used to choose the individual according to the fitness values. A hyper-heuristic, which circularly uses two interactional hyper-heuristics in multiple stages, was proposed[30]. In the algorithm, the dominance-based hyper-heuristic is used to decrease the number of LLHs, which helps to exclude the inferior heuristic from the set of the LLHs. In addition, the relay hybridization, which uses the second heuristic for the novel solution generated by the previous heuristic, is introduced in another stage. The purpose of relay hybridization is to obtain more potential heuristics by pairing two existing heuristics. Furthermore, a method based on the adaptive threshold acceptance was introduced to accept the novel solution. Reinforcement learning was embedded in a hyper-heuristic framework to address the multi-objective optimization problems[31]. In the algorithm, a robust selection method of LLHs and multiple acceptance methods were used to generate adequate solutions.

The selection of LLHs depends on the feedback from the subsequent decisions[32]. The feedback mechanism embedded in diverse search methods was used to guide the next search direction. The online learning static is embedded in the hyper-heuristic framework to handle feedback during the search process. A choice function based on online learning was proposed to address aircraft flight deck operations scheduling[33]. The online learning mechanism, which is based on dynamic Thompson sampling, has a considerable effect on the performance of local search[26]. Offline learning is another learning mechanism that is trained on sample problem instances[34]. The parameters are determined according to the feedback information and used to guide test instances. The mixed learning mechanism, which combines online and offline learning, was proposed to select the appropriate low-heuristics. In this study, the online learning mechanism is used in the single-point search. Furthermore, the relay hybridization is embedded in the single-point search to promote the search ability of the Cooperative Multi-Stage Hyper-Heuristic algorithm (CMS-HH). The contributions of this paper are summarized as follows:

A GA is introduced to perturb the initial solution to increase the diversity of the solution.

An online learning mechanism based on the Multi-Armed Bandits (MAB) mechanism and relay hybridization technology is proposed to improve the quality of the solution.

The multi-point search is introduced to cooperatively search with a single point when the state of the solution does not change in continuous time.

The remainder of this paper is described below. The description of problems is introduced in Section 2. The proposed algorithm is introduced in Section 3, and Section 4 is the experiment and discussion. Conclusions and future work are presented in Section 5.

2 Problem Description

The Hyper-heuristic Flexible framework software (HyFlex) is used to study the hyper-heuristic

algorithm. HyFlex is an interface that is used to support the research of hyper-heuristics. The interface, which involves six domains, is achieved by Java[21]. The purpose of the Cross-domain Heuristic Search Challenge (CHeSC) is to determine the best algorithm that is well generalized to various examples of cross-domain problems. This hyper-heuristic flexible framework contains six problem domains. Five instances need to be optimized for each domain. The researchers only focus on the design of the high-level strategies.

In HyFlex, each domain contains a set of LLHs, initial solution construction methods, and optimization functions. A set of different LLHs is used to disturb the complete solution, and a complete candidate solution is established subsequently. These heuristics are classified as mutation heuristics that modify a solution by moving, exchanging, adding, or deleting components of the solution, not ensuring the quality of the solution. The ruin-recreate heuristic is used to destroy a part of a complete solution, rebuild a novel, and complete solution. The local search heuristic iteratively generates a neighborhood solution and then receives a solution of equal or better quality until the termination conditions are met. The difference between the two heuristics is that the local search heuristic is an iterative process that merely accepts improved solutions. The crossover heuristic combines the partial components of two given solutions to produce a novel solution. In HyFlex, each LLH is related to two parameters. The behavior of an LLH is controlled by two parameters: the search depth and mutation intensity. The number of heuristics provided by HyFlex for each supported problem domain are shown in Table 1. In Table 1, the variable "Xover" represents the crossover heuristics, "HC" represents the hill climbing heuristics, "R&R" represents the ruin-recreate heuristics, "M" represents the mutational heuristics, "Total" represents total number of heuristics for each problem domain. The diverse instances in HyFlex are derived from eminent benchmark suites. Six optimization problem domains are provided via HyFlex.

Table 1 Number of different types of low-level heuristic in different problem domains.

Number	Domain	Xover	HC	R&R	M	Total
1	SAT	2	2	1	6	11
2	BP	1	2	2	3	8
3	PFSP	4	4	2	5	15
4	PS	3	5	3	1	12
5	TSP	4	3	1	5	13
6	VRP	2	3	2	3	10

(1) Boolean SATisfiability (SAT) problems: The SAT determines the maximum number of clauses in the conjunctive normal form of a given Boolean form. This result is achieved by assigning truth values to the formula variables. The goal is to minimize the number of items that are not met[35]. The properties of the problem instances are exhibited in Table 2. In Table 2, "Variables" represents the number of variables in a given Boolean logic formula, and "Clauses" represents the number of clauses of Boolean formula. The initial solution is constructed by randomly assigning a true or false value to each variable of the conjunctive normal form formula, and the quality of the solution is measured by the number of unsatisfied terms in the given formula. In SAT, the physical meaning of the objective function value is the number of clauses in the conjunctive normal form of a given Boolean form.

(2) One-dimensional Bin Packing problems (BP): Given items with stationary weight and bin with a specific capacity, the purpose is to put all the items into the bins using a minimum number of bins. This optimization process must satisfy the specific constraints. Every item is only packed into one bin. The overall size of items for every bin is not allowed to exceed the capacity of the bin. The ultimate objective of the optimization is to use the minimum number of bins as much as possible[36]. The properties of the problem instances are exhibited in Table 3, where "Capacity" represents the capacity of single bin, and its unit is kg. "Pieces" represents the total number of goods. The initial

Table 2 Boolean satisfiability problems instance.

Number	Instance	Name	Variables	Clauses
1	SAT1	parity-games/instance-n3-i3-pp	525	2276
2	SAT2	parity-games/instance-n3-i4-pp-ci-ce	696	3122
3	SAT3	parity-games/instance-n3-i3-pp-ci-ce	525	2336
4	SAT4	jarvisalo/eq.atree.braun.8.unsat	684	2300
5	SAT5	highgirth/3SAT/HG-3SAT-V300-C1200-4	300	1200

Table 3 One-dimensional bin packing instance.

Number	Instance	Name	Capacity (kg)	Pieces
1	BP1	triples2004/instance1	1000	2004
2	BP2	falkenauer/u1000-01	150	1000
3	BP3	test/testdual7/binpack0	100	5000
4	BP4	50–90/instance1	150	2000
5	BP5	test/testdual10/binpack0	100	5000

solution is constructed by the first fit heuristic method. A sequence of items is randomly generated, and the items are put into bins in turn until the constraints of the bin are violated. The quality of the solution is assessed in the following:

$$\text{quality} = 1 - (1/n_b) \times \sum_{i=1}^{n_b} (f_i/C)^2 \qquad (1)$$

where n_b is the number of bins, f_i is the total size of the items in bin i, and C is the bin capacity. In BP, the physical meaning of the objective function value is the percentage of empty parts of bins in the capacity of all used bins.

(3) PFSP: The PFSP is described as follows: given n jobs, each job is processed on m machines according to the same process route. The mission is to seek the sequence of n jobs on m machines with the minimal completion time of the last job. The following constraints must be respected by the generated sequence. Each job must be processed on one machine at a time, and each machine cannot process multiple jobs at the same time. Following the processing sequence of the jobs, the machine cannot remain idle when a job is ready to be processed[37]. The properties of the problem instances are exhibited in Table 4. In Table 4, "Jobs" represents the total number of jobs for processing, and "Machines" represents the total number of machines in a single factory. The initial solution is generated by the Nawaz-Enscore-Ham(NEH) algorithm. It first assigns priority according to the total processing time of the jobs and then continuously inserts the job to obtain a complete schedule. In PFSP, the physical meaning of the objective function value is the completion time of the last job in a job sequence. The unit of the objective function value is hour.

(4) Personnel Scheduling problem (PS): Personnel scheduling is a famous combinatorial optimization problem. The personnel scheduling is described as follows. The detailed arrangement shifts of a working day, predefined working days, and specific work are categorized to some employees. The task is to arrange for all employees to satisfy specific requirements and some preferences within a reasonable period[38]. The properties of the problem instances are exhibited in Table 5. In Table 5, "Staff" represents the number of employees, "Shift types" represents the number of shift types, and "Days" represents the predefined working days of a single shift. The initial solutions are created by a method based on the neighborhood operator. In this method, the new shifts are gradually added to the roster until all staff are scheduled. The quality of the solutions is measured by the number of satisfied soft constraints. In PS, the physical meaning of the objective function value is the time required to complete all shifts. The unit of the objective function value is hour.

(5) TSP: Given some cities and related coordinates, the purpose is to seek the shortest path. Every city is reached only one time, and the route is ended at the starting city. The target is to minimize the sum of traveling distances[39]. The properties of the problem instances are exhibited in Table 6. In Table 6, "Cities" represents the total number of cities. The initial solution is created by randomly generating permutation sequences. The quality of the solution is measured by the total traveling distance of the solution path. The value of the objective function of a given solution is calculated as follows:

$$f = \sum_{i=1}^{n_c} \sum_{i \neq j, j=1}^{n_c} D_{ij} x_{ij} \qquad (2)$$

where x_{ij} is the path from city i to j, D_{ij} is the distance between cities i and j, and n_c is the number of cities. In TSP, the physical meaning of the objective function value is the total traveling distance of connecting all cities. The unit of the objective function value is km.

(6) VRP: Various customers have different demands and service times. A few vehicles have a definite capacity.

Table 4 Permutation flow shop instance.

Number	Instance	Name	Jobs	Machines
1	PFSP1	100 × 20/2	100	20
2	PFSP2	500 × 20/2	500	20
3	PFSP3	100 × 20/4	100	20
4	PFSP4	200 × 20/1	200	20
5	PFSP5	500 × 20/3	500	20

Table 5 Personnel scheduling problem instance.

Number	Instance	Name	Staff	Shift types	Days
1	PS1	Ikegami-3Shift-DATA1.2	25	3	30
2	PS2	MER-A	54	12	42
3	PS3	ERRVH-B	51	8	42
4	PS4	BCV-A.12.1	12	5	31
5	PS5	ORTEC01	16	4	31

Table 6 Traveling salesman problem instance.

Number	Instance	Name	Cities
1	TSP1	pr299	299
2	TSP2	usa13 509	13 509
3	TSP3	rat575	575
4	TSP4	d2152	2152
5	TSP5	D1291	1291

The mission is to seek the minimum cost sets of routes that serve all customers. The starting and ending points for each vehicle must be the same depot. The vehicle capacity cannot exceed the total demand of each route. Furthermore, every customer may only be visited once by one vehicle during its time window[40]. The properties of the problem instances are exhibited in Table 7. In Table 7, "Vehicle" represents the total number of vehicles, and "Capacity" represents the capacity of single vehicle. The unit of Capacity is kg. The set of an initial solution are created by the following rules: An empty route is created first, and any customer who does not breach any constraints will be added to the present route until all customers are visited. Furthermore, a new route will be created if no customer is added to the current route. The procedure is reduplicated until the customers who meet the constraints are allocated to the related route. The quality of the solution is measured by the total travel distance, which is calculated in the following:

$$F = n_v \times C_v + \sum_{i=1}^{n_v} d_i \tag{3}$$

where n_v is the number of vehicles, C_v is a fixed constant with a value of 1000, and d_i is the distance traveled by the vehicles. In VRP, the physical meaning of the objective function value is the total length traveled by all vehicles. The unit of the objective function value is km.

3 Proposed CMS-HH Algorithm

3.1 Proposed hyper-heuristic framework

The HLHs and LLHs are included in the hyper-heuristic framework. The high-level heuristic consists of two ingredients: a heuristic selection mechanism and acceptance criterion. A series of perturbative LLHs, initial solution construction, objective function, and the memory mechanism are included in the framework. The proposed algorithm is started from an initial solution, and the neighborhood structure is iteratively explored by applying perturbed LLHs. This hyper-heuristic framework continuously calls the following steps at a certain stage:

Table 7 Vehicle routing problem instance.

Number	Instance	Name	Vehicles	Capacity (kg)
1	VRP1	Homberger/RC/RC2-10-1	250	1000
2	VRP2	Solomon/RC/RC103	25	200
3	VRP3	Homberger/C/C1-10-1	250	200
4	VRP4	Solomon/R/R101	25	1000
5	VRP5	Homberger/RC/RC1-10-5	250	200

- The selection mechanism is invoked, and a disturbance heuristic is chosen from the set of LLHs.
- A solution is randomly selected from the memory mechanism.
- The selected LLH is applied to the incumbent solution to create a candidate solution.
- The candidate solution is assessed by invoking the objective function. If the candidate solution is superior to the existing solution, accept it. If not, decide whether to accept it by using the selection mechanism.
- The memory mechanism and related parameters are updated, and the next-generation iteration is continued.

The entire framework of the multi-stage hyper-heuristic is presented in Fig. 1. In this framework, the control layer of the proposed hyper-heuristic algorithm is separated from the problem domain through the domain barrier. The multi-stage level is employed to cyclically select two interacting HLHs, namely S1HH and S2HH. Where "Stage$_k$" represents the k-th stage, "SiHH" means that S1HH or S2HH is selected in Stage$_k$. Each HLH consists of a heuristic selection mechanism and moving acceptance criteria. The heuristic selection is used to choose one heuristic (LLH$_k$) from the set of LLHs according to the Roulette Wheel Strategy (RWS). The selected LLH is applied to the incumbent solution ($S_{\text{incumbent}}$). A new candidate solution($S_{\text{candidate}}$) is generated subsequently, "S_{best}" represents the best solution obtained at a certain stage. The acceptance criteria are used to accept the candidate solution. The pseudocode of proposed hyper-heuristics is shown in Algorithm 1, where p is a random probability within interval (0,1). The calculation of probability p_r is shown in Eq. (4).

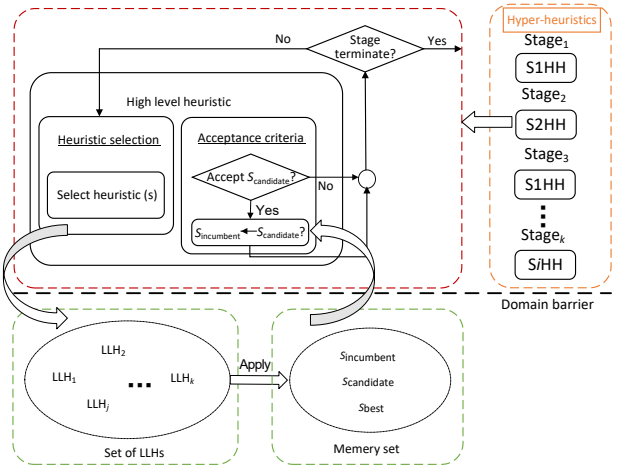


Fig. 1 Framework of the multi-stage hyper-heuristic.

Algorithm 1 Pseudocode of hyper-heuristic

```
S_incumbent ← init()
while notSatisfied (terminationCriteria) do
    count ← count + 1;
    if notSatisfied (restart condition) then
        Single-point search;
    else
        Multi-point search;
    end
    Update p_r;
    if p > p_r then
        h_index ← select one heuristic by MAB;
        S_candidate ← apply_heuristic(h_index, S_incumbent);
        Update the set of probability for h_index;
    else
        h_r1 ← select the first heuristic by RWS;
        S_candidate ← apply_heuristic (h_r1, S_incumbent);
        if S_candidate ←not best since the last re-init then
            h_r2 ← select the second heuristic by RWS;
            S_candidate ← apply heuristic (h_r2, S_candidate);
            Update the set of probability for h_r1 and h_r2;
        end
    end
    if accept (S_candidate) then
        S_incumbent ← S_candidate;
    end
    if count == pl then
        count ← 0;
end while
```

3.2 Proposed high-level strategy

The proposed hyper heuristic performs the search in a stage-based method. During the search process, the operation mechanism used in different stages is switched iteratively. The initial solution is formed according to the initialization method of diverse problem domains. A memory mechanism that contains multiple initial solutions is introduced to retain the diversity of the solution. The MAB and relay hybridization technology are introduced to single-point search[41,42]. During the single-point search, the switching of two searching methods is determined by the probability p_r, which is calculated as follows:

$$p_r = (n_{\text{pit}}/\text{pl})^{(n_{\text{total}}-n_{\text{relay}}+1)/(n_{\text{relay}}+1)} \tag{4}$$

where n_{pit} is the present number of iterations, pl is the total number of iterations, n_{total} is the number of improved solutions obtained by the single-point search, and n_{relay} is the number of improved solutions obtained by the relay hybridization technology.

Two tactics are used to select appropriate LLHs in the single-point search. The MAB is a sequential decision-making problem, and the purpose of this strategy is to maintain the balance between exploitation and exploration. The exploitation, which is to ensure the best return on past decisions, is used to select the LLH that continuously improves on the incumbent solution. The exploration, which is for obtaining a bigger payoff in the future, is used to select the other LLH effectively. Relay hybridization is a successful tactic used in the literature. The pairwise heuristics are employed in the incumbent solution to obtain a better solution. In other words, the first LLH is applied in the incumbent solution to obtain a poorer solution, and the second LLH is used in the candidate solution to produce a better solution than the incumbent solution. The MAB and relay hybridization technique strategies are described below.

3.2.1 MAB

The MAB is an online learning mechanism. The principle adopted by MAB is the upper confidence bound. The number of LLHs is represented by N_{op} in a certain problem domain. MAB selects an LLH that maximizes the accumulated reward,

$$\max_{i=1,\ldots,N_{\text{OP}}} = \left(q_{i(t)} + c \sqrt{\frac{2\log \sum_{j=1}^{N_{\text{op}}} n_{j(t)}}{n_{i(t)}}} \right) \tag{5}$$

where c (c = 12) is a scaling factor, which maintains the balance between the LLHs with a high probability of reward and the LLHs that are infrequently applied. $n_{i(t)}$ is the number of times that the i-th LLH has been applied up to time t. $n_{j(t)}$ is the number of times that a low-level heuristic has been called up to time t. The $q_{i(t)}$ is the average reward obtained by the i-th LLH at time t, which is calculated in the following:

$$q_{i(t+1)} = \frac{n_{i(t)} \times q_{i(t)} + r_{i(t)}}{n_{i(t)}} \tag{6}$$

where $r_{i(t)}$ is the value of the cumulative reward of the i-th LLH at time t, which is calculated in the following:

$$r_{i(t)} = ((f_{\text{in}} - f_n)/f_{\text{in}}) \times 100\% \tag{7}$$

where f_{in} is the function value of the incumbent solution. f_n is the function value of the new solution generated by applying the i-th LLH to the incumbent solution. The number of times that each low-level heuristic n_i calls is initialized to 0, and the selection probability of each low-level heuristic q_i is initialized to 0. The pseudocode of MAB is exhibited in Algorithm 2.

3.2.2 Relay hybridization technique

The relay hybridization technique is employed as the number of current iterations increases. Two heuristics

Algorithm 2 Pseudocode of MAB

```
for i = 1 to N do
  n_i ← 0;
  q_i ← 0;
end
while notSatisfied (terminationCriteria) do
  if LLHs are not applied then
    LLH ← Select one LLH randomly;
    Apply the LLH;
  end
  else
    Select one LLH using Eq. (5);
    Apply the LLH;
  end
  Update n_i and n_{i+1};
  Update q_i using Eq. (6);
end while
```

are successively applied in an iteration. The selection of the second heuristic depends on the choice of the first heuristic. The strategy of roulette selection is introduced to select the first heuristic. The reward mechanism is used to update the selection probabilities of all LLHs. The initial selection probability of all heuristics is initialized as $1/M$. M is the number of the LLHs for each problem domain. The roulette wheel strategy is applied to select an appropriate heuristic, and it is subsequently applied to the incumbent solution. The punishment and reward mechanisms are used to update the probabilities of LLHs. If the quality of the generated solution is improved, the heuristic is rewarded to update the probability of an LLH being selected in the heuristic set with Eq. (8), while the $M-1$ heuristics are penalized with Eq. (9),

$$pro_i = \eta \times (1 - pro_i) + pro_{i-1} \tag{8}$$

$$pro_j = pro_{j-1} - \eta \times pro_{j-1} \tag{9}$$

where η is the penalty coefficient, and $\eta = 0.5$ here.

The second heuristic choice is based on the fact that the first heuristic does not find a superior solution to the incumbent solution. For each low-level heuristic, a list of length 10 is retained. The subscript of the LLH that has obtained the best solution is retained in the list. The oldest element is replaced when a new superior solution is found, or the elements of the list are full. The heuristic index is allowed to appear multiple times in the list, because this combination helps to promote the quality of the incumbent solution again. Finally, if the LLH matched with the above conditions is found in the list, the second heuristic is selected randomly from the set of the LLHs.

3.2.3 Acceptance mechanism

The improved solution generated by each iteration is used to update the elements in the list (see Algorithm 3, lines 3 and 4). The moving acceptance based on the list is used to accept the worsening solution (see Algorithm 3, lines 12–23), where l is the maximum number (default 6) of each threshold value that accepts the worsening solution, index is the subscript index of the elements in the list, and $f(\cdot)$ represents the objective function for different problem domains. If the maximum number of worsening solutions that the list can accept is exceeded, the multi-point search method is used for the next phase of the search. The pseudocode of the moving acceptance mechanism is presented in Algorithm 3.

In this study, the multi-point search, which employs a GA, is used to cooperatively explore the search space with the single-point search. In the population, the individual is coded using the index of a set of LLHs for a particular problem domain. An individual is considered a chromosome in the population. Each individual is randomly initialized with a different length of the chromosome. Two-parent individuals selected by tournament rules and crossover operations are executed on the parent individuals to complete the evolution of the population. In the process of evolution, the LLHs for each individual are applied to the incumbent solution in

Algorithm 3 Pseudocode of the moving acceptance

```
1:  Input: l, index, count
2:  if (f(S_candidate) < f(S_incumbent)) then
3:    if f(S_candidate) < runbest_list(0)) then
4:      runbest_list.push(f(S_candidate));
5:      index ← 0;
6:      S_incumbent ← S_candidate;
7:      l ← 0;
8:    end
9:  else if f(S_candidate) = f(S_incumbent) then
10:   S_incumbent ← S_candidate;
11: else
12:   if f(S_candidate) <runbest_list.get(index) then
13:     S_incumbent ← S_candidate;
14:     count ← count + 1;
15:   end
16:   if count ⩾ l then
17:     count ← 0;
18:     index ← index + 1;
19:   end
20:   if index ⩾ bestlist_size then
21:     perform multi-point search;
22:     index ← index − 1;
23:   end
24: end
```

turn, and the incumbent solution is updated whenever the quality of the solution is improved.

4 Experimental Results and Analysis

The experimental section mainly compares the performance of the CMS-HH algorithm with other algorithms. The objectives are as follows: (1) To evaluate the benefits of combining MAB with relay hybridization. (2) To verify the consistency and generality of the CMS-HH in six various domains. In this paper, two sets of experiments are conducted for each problem instance. (1) The first set of experiments compares the performance of the CMS-HH without relay hybridization technology (denoted as HH1), the CMS-HH without the MAB strategy (denoted as HH2), and the combination of the two strategies used in the hyper-heuristic separately. (2) The performances of the CMS-HH and five other top algorithms in the CHeSC competition are compared in the second set of experiments.

The CMS-HH is run 31 times independently using various random seeds and initial solutions according to CHeSC rules. When the termination condition of the algorithm is reached, the runtime is over, which is 600 seconds. The standard software provided by the CHeSC competition website is used to fairly compare among diverse algorithms for various platforms. Furthermore, the percentage deviation of the best value from the comparison algorithms is calculated as follows:

$$\Delta(\%) = (\text{best}_{\text{CMS-HH}} - \text{best}_*)/\text{best}_* \times 100\% \quad (10)$$

where $\text{best}_{\text{CMS-HH}}$ is the minimum value obtained by the CMS-HH, and best_* is the minimum value produced by other comparison algorithms. The performance of the CMS-HH is compared with HH1, HH2, and other existing hyper-heuristics to validate the consistency, generality, and effectiveness of CMS-HH algorithms. The Formula One System is used to assess the performance of the CMS-HH algorithm with other compared algorithms. The same method is used by the CHeSC organizers to rank diverse algorithms.

4.1 Parameters setting

Six instances selected from diverse problem domains are tested to obverse the performance of the proposed algorithm under the different parameter settings. The proposed algorithm is run 10 times individually for selected instances. Each LLH in HyFlex is related to a parameter that affects its behavior to a certain extent. The performance of the proposed algorithm is affected by two numerical parameters: α and $\beta(0 \leqslant [\alpha, \beta] \leqslant 1)$, representing the intensity of mutation and depth of search, respectively, which control the behavior of certain LLHs. Through a lot of simulation experiments, two numerical parameters in the CMS-HH are suggested as follows: $\alpha = 0.4$ and $\beta = 0.3$.

4.2 Computational results of the CMS-HH compared to HH1 and HH2

The comparison of the CMS-HH, HH1, and HH2 across six domains is presented in the first set of experiments. Five instances are contained in each domain for a total of 30 instances. The experimental results of the CMS-HH, HH1, and HH2 for the 30 instances are summarized in Table 8, where "Min" represents the minimum value of objective function. "Avg" represents the average values of 31 runs for each instance. "Std" represents the standard deviation, and "Median" represents the median of objective function over 31 trials for each instance. For the minimum values of objective function, the CMS-HH outperforms, matches, and underperforms HH1 on 21, 7, and 2 instances, respectively. CMS-HH outperforms HH2 on 18 instances. In addition, the consistency of CMS-HH is analyzed by the standard deviation and the median. the standard deviation is calculated in the following:

$$\text{std} = \sqrt{\frac{1}{N_1}\sum_{i=1}^{N_1}(X_i - \mu)^2} \quad (11)$$

where N_1 ($N_1 = 31$) is the run times, X_i is the objective value of each run, and μ is objective average value of N_1 runs.

The standard deviation of CMS-HH is superior to those of HH1 and HH2 in most instances. The following conclusions are obtained regarding the median: CMS-HH outperforms, matches, and underperforms HH1 on 19, 5, and 6 instances, respectively. CMS-HH obtained 29 better results than HH2 and matched HH2 on 1 instance.

The various statistical conclusions considering the performance of CMS-HH, HH1, and HH2 are obtained. In addition to the above results, the Wilcoxon test is used to verify the statistical performance differences between the CMS-HH algorithm and two comparison algorithms in Table 9. Expressly, two pairwise comparisons between CMS-HH and the other algorithms are designed to visualize the significance of the proposed MAB strategy and relay hybridization technology. The Wilcoxon test uses the sign test of paired observation data to deduce the probability when the difference appears. R^+ represents

Table 8 Performance comparison of CMS-HH, HH1, and HH2 based on the minimum (Min), average (Avg), associated standard deviation (Std), and minimum median (median) of the objective values over 31 trials in different instances from six diverse problem domains.

Domain	Instance	CMS-HH				HH1				HH2			
		Min	Avg	Std	Median	Min	Avg	Std	Median	Min	Avg	Std	Median
SAT	SAT1	**0**	3.8	2.4	**4**	1	6.4	4.3	5	3	15.0	4.6	9
	SAT2	**1**	7.5	4.6	**6**	3	21.3	13.7	10	18	44.8	9.8	28
	SAT3	**1**	3.1	1.7	**2**	**1**	7.1	5.3	6	**1**	26.3	14.0	18
	SAT4	**1**	4.4	2.8	**4**	**1**	5.7	3.3	5	12	21.4	4.6	16
	SAT5	**7**	7.5	0.7	**7**	**7**	10.4	1.2	8	13	15.	1.7	14
BP	BP1	**1.06**	1.36	0.0170	1.35	1.09	1.39	0.0210	**1.29**	1.60	1.98	0.0015	1.78
	BP2	**0.27**	0.34	0.0070	**0.35**	0.32	0.46	0.0090	0.36	0.78	1.04	0.0021	0.87
	BP3	**0.04**	0.27	0.0017	0.35	0.35	0.45	0.0013	**0.34**	1.04	1.28	0.0011	1.15
	BP4	**10.83**	1.83	0	**10.83**	**10.83**	10.83	0	**10.83**	10.84	10.84	0	10.84
	BP5	**0.03**	0.25	0.0013	**0.32**	0.23	0.37	0.0012	**0.32**	1.87	2.10	0.0015	2.08
PFSP	PFSP1	**6205**	6247.06	24.12	**6244**	6224	6274	28.73	6257	6301	6353.3	29.8	6328.1
	PFSP2	**26 692**	26 781.48	47.97	26 776	26 729	26 757.0	50.1	**26 734**	26 849	26 976.9	54.7	26 937.4
	PFSP3	**6303**	6323.32	20.16	**6323**	**6303**	6324.71	22.49	6326	6369	6405.5	23.7	6389.6
	PFSP4	**11 311**	11 364.58	30.54	**11 359**	11 359	11 412.7	33.6	11 378	11 436	11 529.3	35.9	11 489.4
	PFSP5	**26 479**	26 564.26	51.43	**26 568**	26 532	26 603.8	48.9	26 601	26 702	26 779.1	49.8	27 634.0
PS	PS1	**13**	21.48	3.39	**22**	15	21.83	4.16	23	22	32.6	4.9	27
	PS2	**9259**	9429.77	97.76	**9415**	9362	9558.47	135.60	9423	9283	9528.0	106.7	9430
	PS3	**3134**	3186.35	50.33	**3165**	3148	3208.45	50.11	3197	3195	3324.9	108.2	3231
	PS4	**1425**	1696.77	128.75	**1709**	1495	1714.21	152.70	1710	1495	1836.0	187.3	1590
	PS5	**280**	300.48	17.93	**295**	**280**	303.84	17.48	**295**	360	810.7	621.5	540.3
TSP	TSP1	**48 194.9**	48 195.99	5.8	**48 194.9**	**48 194.9**	48 194.9	0	**48 194.9**	**48 194.9**	48 208.9.9	31.8	**48 194.9**
	TSP2	**2.057×10^7**	2.08×10^7	284 276.0	**2.065×10^7**	2.126×10^7	2.138×10^7	373 168.0	2.135×10^7	2.142×10^7	2.166×10^7	534 583.0	2.149×10^7
	TSP3	**6795.97**	6811.45	13.4	**6808.8**	6922.94	6941.26	17.8	6933.5	6988.6	7040.2	31.3	7013.0
	TSP4	**65 704.0**	66 526.47	648.6	**66 320.5**	68 160.2	68 594.8	469.7	68 448.3	68 791.0	70 241.9	704.6	69 898.2
	TSP5	**52 272.0**	52 901.03	505.4	**52 658.8**	52 473.6	53 102.8	847.5	52 855.2	53 992.4	55 814.8	946.4	54 896.8
VRP	VRP1	57 876.0	60 343.85	2268.9	59 961.8	**57 694.9**	59 298.6	1746.2	**58 135.1**	68 958.3	84 103.9	7225.8	83 094.9
	VRP2	**12 298.9**	13 132.04	392.4	13 319.2	12 302.4	13 137.4	401.2	**12 317.4**	13 320.0	13 695.8	473.9	13 431.0
	VRP3	**142 488.6**	144 740.01	1042.9	**145 274.0**	143 901.1	145 039.6	1625.8	145 294.8	145 362.7	149 553.2	2377.8	146 834.9
	VRP4	**20 650.8**	20 653.37	1.3	**20 653.5**	20 652.1	20 653.1	0.8	**20 653.5**	20 657.5	21 131.9	510.3	20 683.5
	VRP5	144 558.1	146 312.05	1134.5	146 136.4	**143 980.9**	145 929.5	1272.5	**145 127.39**	146 666.9	15 282.6	1616.3	149 107.9

Note: The unit for BP is %; for PFSP is hour; for PS is hour; for TSP is km; and for VRP is km, so do Tables 11–13.

Table 9 Statistical analysis of Wilcoxon's rank-sum test.

CMH-HH vs.	R^+	R^-	Z	p-value	$\gamma = 0.05$ p-value$<\gamma$?	$\gamma = 0.1$ p-value$<\gamma$?
HH1	377	88	−2.972	2.96×10^{-3}	Yes	Yes
HH2	408	570	−3.610	3.06×10^{-3}	Yes	Yes

positive rank. R^- represents negative rank. In the Wilcoxon test, p-value is the probability of sample observation results when the original hypothesis is true, if p-value is less than the p-value obtained under γ, significant differences are present in the pairwise algorithms. Z is the normalized value of Willcoxon statistic. "Yes" means that CMS-HH algorithm is superior to other comparison algorithms in 90% ($\gamma = 0.1$) and 95% ($\gamma = 0.05$) confidence intervals. Table 9 shows that all p-values are less than the p-value obtained under γ. Therefore, the CMS-HH is significantly better than HH1 and HH2.

The Friedman-test is used to make a statistical comparison between the CMS-HH and two compared algorithms in Table 10. The Friedman-test is a nonparametric statistical test for determining significant differences in multiple (related) samples. The results from the Friedman-test are shown in Fig. 2. From Fig. 2, CD represents the critical difference, which is obtained by the Friedman-test. The solid line in Fig. 2 represents the mean rank value when CD equals 0.578 and γ equals 0.02. The dotted line in Fig. 2 represents the mean rank value when CD equals 0.500 and γ equals 0.1. The mean rank of CMS-HH is the minimum. Furthermore, the performance of CMS-HH is excellent compared with other algorithms. In general, significant differences are found among multiple algorithms, and the mean rank of CMS-HH is smaller than those of other algorithms. In general, the results validate the superiority of the CMS-HH over HH1 and HH2 in terms of consistency, generality, and efficiency. The performance of the CMS-HH is reduced without the MAB strategy or the relay hybridization strategy, according to the experimental results.

Table 10 Friedman-test results of CMS-HH, HH1, and HH2.

Algorithm	Mean rank
CMS-HH	**1.23**
HH1	1.93
HH2	2.83

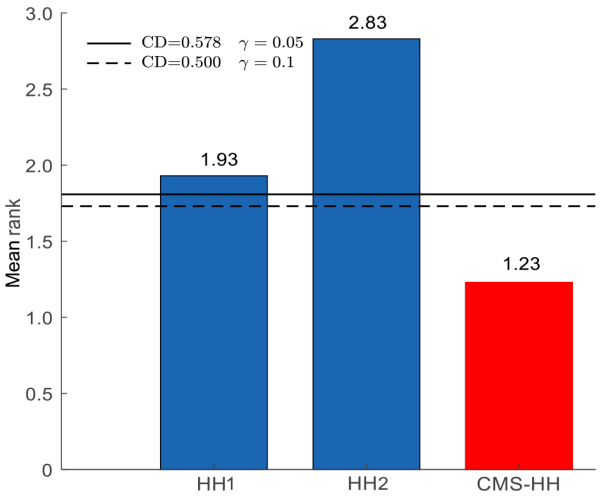


Fig. 2 Mean rank of the Friedman-test.

4.3 Computational results of CMS-HH compared to other hyper-heuristics

The results achieved by CMS-HH are listed in Table 11. The minimum values of objective function, percentage deviation, and instance ranking results of the comparison algorithms are provided. "Rank" represents the ranking of objective function value of CMS-HH algorithm for all current comparison algorithms. For example, "1" means that the objective function value of CMS algorithm is the smallest in all comparison algorithms for current instance. "=" means that there are other algorithms getting the same minimum objective value as CMS-HH algorithm for current instance. "2" means that CMS-HH algorithm obtains the second smallest objective function value in all comparison algorithms. The minimum values of the experimental results are listed in bold. New minimum values are produced by CMS-HH in 15 of 30 instances. CMS-HH matches the other algorithms in minimum value in 10 instances and yields inferior values in 5 instances. CMS-HH achieves superior results on SAT, BP, and PFSP instances. Furthermore, three superior objective values for TSP and VRP are obtained by CMS-HH. The optimization of CMS-HH for PS instances is inferior and only two good results are obtained. In Table 12, the median of objective function, percentage deviation, and instance ranking results of the comparison algorithms are provided. The best median results of the experimental results are listed in bold. New minimum values are produced by CMS-HH in 17 of 30 instances. CMS-HH matches the other algorithms in minimum value in 5 instances and yields inferior values in 8 instances. For these 8 instances, CMS-HH obtains the suboptimal median for 2 instances, the third median for 4 instances, the fourth median for 1 instance, and the fifth median for 1 instance. Although the CMS-

Table 11 Performance comparison of CMS-HH and the top-five comparison algorithms based on the minimum objective values over 31 trials in different instances from six diverse problem domains.

Domain	Instance	CMS-HH	Top five comparison algorithms					Δ (%)	Rank
			AdapHH	VNS-TW	ML	PHUNTER	EPH		
SAT	SAT1	**0**	1	1	1	1	4	−100	1
	SAT2	**1**	3	**1**	3	5	5	0	=
	SAT3	**1**	**1**	**1**	**1**	2	2	0	=
	SAT4	**1**	**1**	**1**	4	4	5	0	=
	SAT5	**7**	9	**7**	**7**	**7**	**7**	0	=
BP	BP1	**1.06**	1.31	2.98	3.23	3.97	4.30	−19.000	1
	BP2	**0.27**	0.28	0.36	0.67	0.34	0.34	−3.571	1
	BP3	**0.04**	**0.04**	1.36	1.24	1.78	0.80	0	=
	BP4	**10.83**	**10.83**	10.87	10.84	10.88	**10.83**	0	=
	BP5	**0.03**	0.31	2.38	1.78	3.18	1.36	−90.322	1
PFSP	PFSP1	**6205**	6214	6230	6226	6221	6232	−0.149	1
	PFSP2	**26 692**	26757	26 765	26 744	26 786	26 738	−0.172	1
	PFSP3	**6303**	**6303**	**6303**	6304	**6303**	6309	0	=
	PFSP4	**11311**	11318	11333	11338	11336	11328	−0.062	1
	PFSP5	**26 479**	26 541	26 535	26 559	26 600	26 569	−0.211	1
PS	PS1	13	17	13	**11**	13	16	18.18	2
	PS2	**9259**	9435	9347	9436	9624	9747	−0.966	1
	PS3	3134	3142	**3124**	3138	3142	3142	0.320	2
	PS4	1425	1448	1370	1384	**1350**	1469	5.556	4
	PS5	**280**	295	290	300	290	310	−3.448	1
TSP	TSP1	**48 194.9**	**48 194.9**	**48 194.9**	**48 194.9**	**48 194.9**	**48 194.9**	0	=
	TSP2	**2.05×10^7**	2.07×10^7	2.08×10^7	2.08×10^7	2.08×10^7	2.09×10^7	−0.966	1
	TSP3	**6796.0**	6797.5	**6796.0**	6805.3	**6796.0**	6799.2	0	=
	TSP4	**65 704.0**	66 277.1	66 830.2	66 428.2	66 641.4	65 958.6	−0.386	1
	TSP5	52 272.0	52 383.8	52 896.5	52 626.7	52 172.0	**52 053.4**	0.420	2
VRP	VRP1	**57 876.0**	58 052.1	68 340.4	67 622.1	61 139.3	63 932.2	−0.303	1
	VRP2	12 298.9	13 304.9	13 298.1	13 298.4	**12 263.0**	13 284.0	0.293	2
	VRP3	**142 488.6**	145 481.5	144 012.6	142 517.0	143 663.9	143 510.8	−0.020	1
	VRP4	**20 650.8**	20 652.3	20 651.1	20 651.1	**20 650.8**	**20 650.8**	0	1
	VRP5	**144 558.1**	146 154.0	146 513.6	146 200.8	146 472.9	145 976.5	−0.972	1

HH algorithm does not obtain the minimum value of objective function and minimum median for all instances, the percentage deviation of those instances is relatively small. Therefore, the CMS-HH outperforms the top five comparison algorithms. To validate the performance of the proposed algorithms, other related algorithms (GEP-HH, MSHH, MCTS-HH) are chosen to compare with CMS-HH. The average values and standard deviations of CMS-HH and the comparison algorithms are shown in Table 13. The optimal values of all algorithms are shown in boldface. The average values and standard deviations are derived from the results of 31 independent runs. For the best results, the CMS-HH algorithm produced 9 new minimum values and 8 results that match the minimum values of the other three algorithms. For the median of objective function, the CMS-HH algorithm obtained 15 new minimum values and only 4 instances that match the other three algorithms. The CMS-HH algorithm is inferior to GEP-HH in VRP instances and MSHH in SAT instances. In general, CMS-HH outperforms the three other algorithms.

The Average Relative Percentage Deviation (ARPD) is considered to evaluate the quality of the best results obtained by CMS-HH and is defined as

$$\text{ARPD} = (1/R)\sum_{1}^{R}((C_i - C_*)/C_*) \times 100\% \quad (12)$$

where C_i is the result of the i-th algorithm for the current instance, R is the number of independent runs, and C_* is the best result of the current instances. For a clear comparison of the experimental results of the top-five hyper-heuristics, a visualization of the experimental

Table 12 Performance comparison of CMS-HH and five comparison algorithms based on the median objective values over 31 trials in different instances from six diverse problem domains.

Domain	Instance	CMS-HH	Top five comparison algorithms					Δ (%)	Rank
			AdapHH	VNS-TW	ML	PHUNTER	EPH		
SAT	SAT1	4	**3**	**3**	5	5	7	33.30	3
	SAT2	6	5	**3**	10	11	11	100.00	3
	SAT3	**2**	**2**	**2**	3	4	6	0	=
	SAT4	4	**3**	**3**	9	9	15	33.34	3
	SAT5	**7**	8	10	8	8	13	−12.50	1
BP	BP1	**1.35**	1.61	3.70	4.21	4.79	5.04	−16.15	1
	BP2	**0.35**	0.36	0.72	0.75	0.36	0.36	−2.78	1
	BP3	**0.35**	0.36	1.67	1.46	2.01	1.13	−2.78	1
	BP4	**10.83**	**10.83**	10.88	10.85	10.91	10.87	0	=
	BP5	**0.32**	0.35	2.78	2.18	3.95	2.24	−8.57	1
PFSP	PFSP1	6244	**6240**	6251	6245	6253	6250	0.064	2
	PFSP2	**26 776**	26 814	26 803	26 800	26 858	26 816	−0.090	1
	PFSP3	**6323**	6326	6328	**6323**	6350	6347	0	=
	PFSP4	**11 359**	**11 359**	11 376	11 384	11 388	11 397	0	=
	PFSP5	**26 568**	26 643	26 602	26 610	26 677	26 640	−0.128	1
PS	PS1	22	24	19	**18**	25	22	22.22	3
	PS2	**9415**	9667	9628	9812	10 136	10 074	−2.21	1
	PS3	**3165**	3289	3223	3228	3255	3232	−1.80	1
	PS4	1709	1765	**1590**	1605	1595	1615	7.48	5
	PS5	**295**	325	320	315	320	345	−6.35	1
TSP	TSP1	**48 194.9**	**48 194.9**	**48 194.9**	**48 194.9**	**48 194.9**	**48 194.9**	0	=
	TSP2	**2.06×10^7**	2.08×10^7	2.10×10^7	2.11×10^7	2.12×10^7	2.11×10^7	−0.962	1
	TSP3	**6808.8**	6810.5	6819.1	6820.6	6813.6	6811.9	−0.025	1
	TSP4	**66 320.5**	66 879.8	67 378.0	66 894.0	67 136.8	66 756.2	−0.653	1
	TSP5	**52 658.8**	53 099.8	554 028.6	54 368.4	52 934.4	52 925.3	−0.504	1
VRP	VRP1	**59 961.8**	60 900.6	76 147.1	80 671.3	64 717.8	74 715.8	−1.542	1
	VRP2	13 319.2	13 347.6	13 367.9	13 329.8	**12 290.0**	13 335.6	8.374	2
	VRP3	**145 274.0**	148 516.8	148 206.2	145 333.5	146 944.4	162 188.5	−0.041	1
	VRP4	20 653.5	20 656.6	21 642.9	20 654.1	**20 650.8**	**20 650.8**	0.013	4
	VRP5	**146 136.4**	148 689.2	149 132.4	148 975.1	148 659.0	155 224.7	−1.697	1

results is provided in Fig. 3. The horizontal axis is the instances from different domains. The vertical axis is the ARPD values for various contrast algorithms. For the state-of-the-art algorithms, the ARPD values of best results and best median results from three problem domains are presented in Fig. 4, which shows a direct expression for measuring the effectiveness of the CMS-HH algorithm.

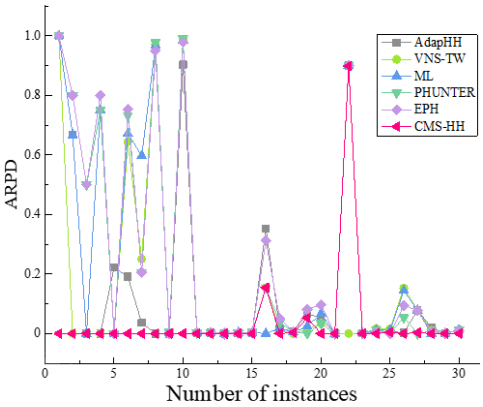
In this study, the Formula One Ranking System is used to determine the score of CMS-HH and the compared algorithms[29]. The ranking scores obtained by the CMS-HH and the other top-five comparison algorithms are shown in Table 14. The CMS-HH algorithms obtain the first rank compared with the top-five comparison algorithms.

The experimental results show that the CMS-HH algorithm is superior to other top-five comparison algorithms in six different complex optimization problems. Moreover, the proposed CMS-HH algorithm obtains new minimum values for 15 out of 30 instances and matches the other top-five comparison algorithms for 9 instances. For all instances from six different domains, the percentage deviation of CMS-HH is similar to those of the other comparison algorithms. The new results are due to the following two factors:

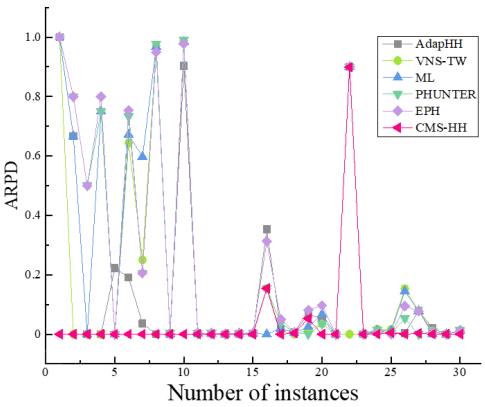
(1) The search space formed by LLHs is effectively explored by CMS-HH algorithms. Online learning, which is based on the multi-armed bandit, can select an appropriate heuristic according to the state of the current solution. The pairwise heuristic formed by relay hybridization helps to promote the quality of the solutions. By searching in a particular region of the

Table 13 Performance comparison of CMS-HH and other three comparison algorithms based on the minimum (Min), average (Avg), associated standard deviation (Std), and median (Median) of the objective values over 31 trials in different instances from six diverse problem domains.

Domain	Instance	CMS-HH				GEP-HH				MSHH				MCTS-HH			
		Min	Avg	Std	Median	Min	Avg	Std	Median	Min	Avg	Std	Median	Min	Avg	Std	Median
SAT	SAT1	**0**	3.8	2.4	4	1	4.4	1.7	3	**0**	0.9	0.7	**1**	**0**	4.4	2.1	4
	SAT2	**1**	7.5	4.6	6	**1**	13.0	11.0	3	**1**	3.1	3.9	2	**1**	3.9	5.6	3
	SAT3	1	3.1	1.7	2	1	3.8	2.4	2	**0**	0.7	0.5	**1**	**0**	3.3	1.2	2
	SAT4	**1**	4.4	2.8	4	4	8.0	4.6	4	**1**	1.7	1.0	1	**1**	4.1	2.3	3
	SAT5	**7**	7.5	0.7	**7**	**7**	7.8	0.9	**7**	**7**	7.6	0.9	**7**	**7**	7.8	0.9	**8**
BP	BP1	**1.06**	1.36	0.0170	**1.35**	1.31	2.90	0.013	1.92	1.36	1.63	0.0014	1.63	1.31	2.73	0.016	1.65
	BP2	0.27	0.34	0.0070	0.35	0.29	0.50	0.003	0.32	**0.25**	0.37	0.0015	**0.30**	0.28	0.35	0.005	0.36
	BP3	**0.04**	0.27	0.0017	**0.35**	0.11	0.30	0.002	0.39	0.25	0.50	0.0015	0.49	**0.04**	0.32	0.0031	0.39
	BP4	**10.83**	10.83	0	**10.83**	**10.83**	10.83	0.001	**10.83**	**10.83**	10.84	0	10.84	**10.83**	10.83	0	**10.83**
	BP5	**0.03**	0.25	0.0013	**0.32**	0.31	1.50	0.010	0.66	0.32	0.50	0.0019	0.44	0.31	0.45	0.010	0.48
PFSP	PFS1	**6205**	6247.06	24.12	6244	6212	6243.29	10.39	6245	6212	6239.8	14.9	**6239**	6212	6253.57	13.09	6248
	PFS2	**26 692**	26 781.48	47.97	**26 776**	26 721	26 821.00	83.79	26 898	26 775	26 895.2	55.3	26 889	26 720	26 851.00	73.74	26 840
	PFS3	6303	6323.32	20.16	**6323**	**6285**	6325.83	11.87	6326	6303	6333.8	19.0	6325	**6285**	6334.56	13.78	6334
	PFS4	**11 311**	11 364.58	30.54	**11 359**	11 320	11 376.70	24.56	11 377	11 320	11 363.8	32.7	**11 359**	11 320	11 376.10	26.54	11 362
	PFS5	**26 479**	26 564.26	51.43	26 568	26 530	26 616.00	40.70	26 634	26 630	26 711.9	47.0	26 709	26 528	26 643.13	57.40	26 621
PS	PS1	13	21.48	3.39	22	**11**	18.64	3.91	**21**	16	25.5	4.5	25	**11**	18.46	3.91	**19**
	PS2	9259	9429.77	97.76	**9415**	9345	10 830.40	1660.50	9628	**9184**	9668.9	217.8	9638	9328	9756.34	152.87	9631
	PS3	3134	3186.35	50.33	**3165**	3123	3312.16	83.10	3351	3132	3283.7	93.3	3270	**3120**	3251.38	84.70	3228
	PS4	1425	1696.77	128.75	1709	1364	1541.48	86.52	**1555**	1545	1786.3	172.1	1760	**1348**	1546.42	85.62	**1557**
	PS5	**280**	300.48	17.93	**295**	**280**	306.54	14.25	315	315	353.2	21.2	350	**280**	312.54	24.52	315
TSP	TSP1	**48194.9**	48195.99	5.80	**48 194.9**	**48 194.9**	48 222.72	38.41	**48 194.9**	**48 194.9**	48 208.1	31.8	**48 194.9**	**48 194.9**	48 194.90	0	**48 194.9**
	TSP2	2.057×10^7	2.082×10^7	284 276	$\mathbf{2.065\times10^7}$	2.075×10^7	2.122×10^7	255 264	2.127×10^7	2.077×10^7	2.095×10^7	905 823	2.091×10^7	$\mathbf{2041\times10^7}$	2.087×10^7	255 264	2.083×10^7
	TSP3	6795.97	6811.45	13.41	**6808.8**	6796.0	6828.34	13.80	6810.5	6796.6	6809.1	7.1	**6808.8**	**6790.5**	6813.34	13.58	6819.6
	TSP4	**65 704.0**	66 526.47	648.60	**66320.5**	65 952.1	67 118.90	493.71	67 105.2	66 236.8	66 840.2	276.5	66 843.6	65 961.1	67 028.60	439.73	66 983.6
	TSP5	52 272.0	52 901.03	505.40	**52 658.8**	**52 050.0**	54 393.66	1015.18	54 755.3	52 341.3	53 011.4	469.7	52 910.2	**52 050.0**	52 361.66	613.90	52 989.7
VRP	VRP1	**57 876.0**	60 343.85	2268.9	**59 961.8**	58 052.1	60 046.30	1444.7	60 720.0	63 948.2	70 998.4	3840.3	70 506.5	58 056.8	60 846.30	1474.1	60 928.4
	VRP2	12 298.9	13 132.04	392.4	13 319.2	**12 261.0**	12 814.52	519.7	**12 337.9**	13 303.9	13 421.8	251.6	13 359.6	**12 260.1**	12 812.64	341.8	**12 314.9**
	VRP3	142 488.6	144 740.01	1042.9	**145 274.0**	**142 479.1**	145 294.40	1622.3	145 418.9	145 466.5	148 498.2	1625.8	148 436.2	**142 461.0**	145 164.30	1246.3	145 390.3
	VRP4	**20 650.8**	20 653.37	1.3	**20 653.5**	**20 650.8**	20 653.60	1.3	20 653.8	**20 650.8**	21 016.4	488.2	20 671.4	20 651.6	20 653.60	1.3	20 654.0
	VRP5	144 558.1	146 312.05	1134.5	**146 136.4**	**144 258.1**	148 943.60	1365.3	149 007.9	146 334.6	148 813.7	1272.5	149 193.7	**144 256.5**	149 302.24	1365.3	149 007.9

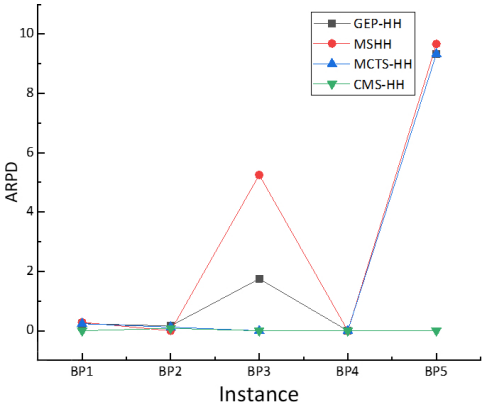


(a) ARPD values of the best results

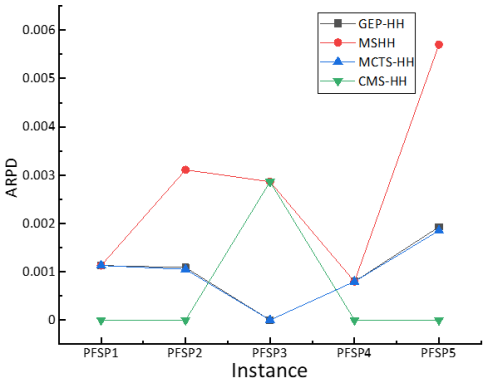


(b) ARPD values of the best median results

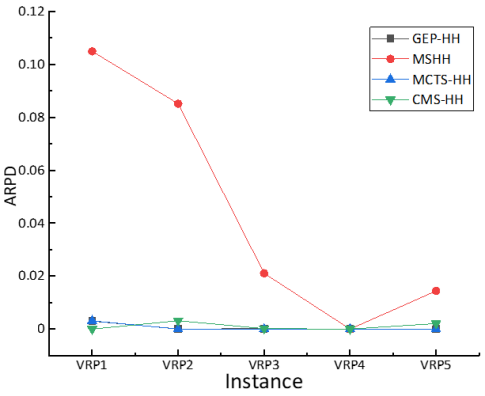
Fig. 3 ARPD values of experimental results of CMS-HH and top-five hyper-heuristics.



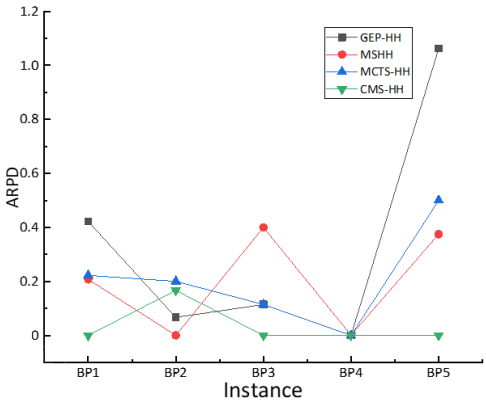
(a) ARPD values of the best results for BP



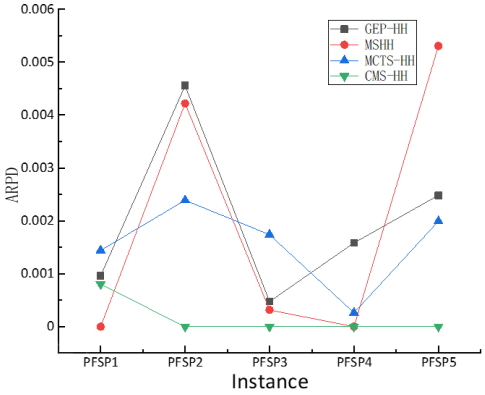
(b) ARPD values of the best median results for PFSP



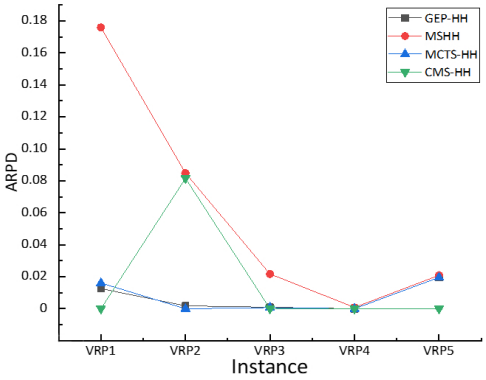
(c) ARPD values of the best results for VRP



(d) ARPD values of the best median results for BP



(e) ARPD values of the best median results for PFSP



(f) ARPD values of the best median results for VRP

Fig. 4 ARPD values of experimental results of CMS-HH and state-of-the-art algorithms.

Table 14 Ranking of CMS-HH with five comparison algorithms.

No.	Algorithm	Score
1	CMS-HH	175.73
2	AdapHH	163.25
3	VNS-TW	114.25
4	ML	109.43
5	PHUNTER	81.10
6	EPH	75.25

solution space, the inferior LLHs are excluded and only superior LLHs are applied.

(2) The different sequences of LLHs are generated by CMS-HH during the search process. By generating a different sequence of heuristics for each instance, the changes that occurred during the research can be handled by the CMS-HH algorithms. Furthermore, different areas are explored by the CMS-HH algorithm, which helps it to avoid falling to a local optimum.

5 Conclusion and Future Work

The proposed hyper-heuristic is a general-purpose search method. The control layer and the low-level heuristic are separated by a domain barrier. The operations based on a certain problem domain must be encapsulated in a low-level heuristic. These methods of control layer are applied to the other problems without any modification, which promotes the reusability of the algorithm. The MAB and relay hybridization technology are well combined to decide the selection of the LLHs, which helps to explore the search space for diverse instances from different problem domains. A list-based adaptive threshold move acceptance method is introduced to accept the inferior solution, which helps to avoid the algorithm falling into a local optimum. The multi-point search embedded in a single search effectively promotes the search ability of the algorithm. The proposed framework is demonstrated to generate highly competitive results. The six problem domains are well generalized by the CMS-HH algorithm compared with other top-five hyper-heuristics. The proposed algorithm is promising from the experimental results.

Despite the excellent performance, several issues still need to be considered in future work. (1) Single-point search and multi-point search are switched flexibly to search the solution space independently; (2) To avoid the search stopping for a long time, the various acceptance mechanisms are introduced for the search to enter another area; and (3) CMS-HH can be applied to address the practical problems. For instance, the distributed flow-shop scheduling problem considers minimizing blocking time and total energy consumption as the optimization objectives.

Acknowledgment

This work was supported by the National Key Research and Development Plan (No. 2020YFB1713600), the National Natural Science Foundation of China (No. 62063021), the Lanzhou Science Bureau Project (No. 2018-rc-98), Public Welfare Project of Zhejiang Natural Science Foundation (No. LGJ19E050001), and Project of Zhejiang Natural Science Foundation (No. LQ20F020011).

References

[1] A. Turky, N. R. Sabar, S. Dunstall, and A. Song, Hyper-heuristic local search for combinatorial optimisation problems, *Knowl.-Based Syst.*, vol. 205, p. 106264, 2020.

[2] X. Y. Cai, M. Hu, D. W. Gong, Y. N. Guo, Y. Zhang, Z. Fan, and Y. H. Huang, A decomposition-based coevolutionary multiobjective local search for combinatorial multiobjective optimization, *Swarm Evol. Comput.*, vol. 49, pp. 178–193, 2019.

[3] J. A. Castellanos-Garzon, E. Costa, S. J. L. Jaimes, and J. M. Corchado, An evolutionary framework for machine learning applied to medical data, *Knowl.-Based Syst.*, vol. 185, p. 104982, 2019.

[4] X. W. Cai, H. B. Qiu, L. Gao, C. Jiang, and X. Y. Shao, An efficient surrogate-assisted particle swarm optimization algorithm for high-dimensional expensive problems, *Knowl.-Based Syst.*, vol. 184, p. 104901, 2019.

[5] R. Olivares, F. Muñoz, and F. Riquelme, A multi-objective linear threshold influence spread model solved by swarm intelligence-based methods, *Knowl.-Based Syst.*, vol. 212, p. 106623, 2021.

[6] J. P. Huang, Q. K. Pan, and L. Gao, An effective iterated greedy method for the distributed permutation flowshop scheduling problem with sequence-dependent setup times, *Swarm Evol. Comput.*, vol. 59, p. 100742, 2020.

[7] H. Park, D. Son, B. Koo, and B. Jeong, Waiting strategy for the vehicle routing problem with simultaneous pickup and delivery using genetic algorithm, *Expert Syst. Appl.*, vol. 165, p. 113959, 2021.

[8] G. D'Angelo and F. Palmieri, GGA: A modified genetic algorithm with gradient-based local search for solving constrained optimization problems, *Inf. Sci. (Ny).*, vol. 547, pp. 136–162, 2021.

[9] Y. Hu, Y. Zhang, and D. W. Gong, Multiobjective particle swarm optimization for feature selection with fuzzy cost, *IEEE Trans. Cybern.*, vol. 51, no. 2, pp. 874–888, 2021.

[10] S. C. Gao, Y. R. Wang, J. J. Cheng, Y. Inazumi, and Z. Tang, Ant colony optimization with clustering for solving the dynamic location routing problem, *Appl. Math. Comput.*, vol. 285, pp. 149–173, 2016.

[11] F. Q. Zhao, L. X. Zhang, Y. Zhang, W. M. Ma, C. Zhang, and H. B. Song, A hybrid discrete water wave optimization algorithm for the no-idle flowshop scheduling problem with total tardiness criterion, *Expert Syst. Appl.*, vol. 146, p. 113166, 2020.

[12] F. Q. Zhao, X. He, and L. Wang, A two-stage cooperative evolutionary algorithm with problem-specific knowledge for energy-efficient scheduling of no-wait flow-shop problem, *IEEE Trans. Cybern.*, doi: 10.1109/TCYB.2020.3025662.

[13] F. Q. Zhao, L. X. Zhao, L. Wang, and H. B. Song, An ensemble discrete differential evolution for the distributed blocking flowshop scheduling with minimizing makespan criterion, *Expert Syst. Appl.*, vol. 160, p. 113678, 2020.

[14] Z. S. Shao, D. C. Pi, and W. S. Shao, A novel discrete water wave optimization algorithm for blocking flow-shop scheduling problem with sequence-dependent setup times, *Swarm Evol. Comput.*, vol. 40, pp. 53–75, 2018.

[15] Z. S. Shao, D. C. Pi, and W. S. Shao, Hybrid enhanced discrete fruit fly optimization algorithm for scheduling blocking flow-shop in distributed environment, *Expert Syst. Appl.*, vol. 145, p. 113147, 2020.

[16] Z. S. Shao, D. C. Pi, and W. S. Shao, A multi-objective discrete invasive weed optimization for multi-objective blocking flow-shop scheduling problem, *Expert Syst. Appl.*, vol. 113, pp. 77–99, 2018.

[17] Y. W. Zhong, L. J. Wang, M. Lin, and H. Zhang, Discrete pigeon-inspired optimization algorithm with Metropolis acceptance criterion for large-scale traveling salesman problem, *Swarm Evol. Comput.*, vol. 48, pp. 134–144, 2019.

[18] G. Y. Zhu, C. Ding, and W. B. Zhang, Optimal foraging algorithm that incorporates fuzzy relative entropy for solving many-objective permutation flow shop scheduling problems, *IEEE Trans. Fuzzy Syst.*, vol. 28, no. 11, pp. 2738–2746, 2020.

[19] J. M. Chen, B. Dan, and J. Shi, A variable neighborhood search approach for the multi-compartment vehicle routing problem with time windows considering carbon emission, *J. Clean. Prod.*, vol. 277, p. 123932, 2020.

[20] J. Q. Li, Y.Q. Han, P. Y. Duan, Y. Y. Han, B. Niu, C. D. Li, Z. X. Zheng, and Y. P. Liu, Meta-heuristic algorithm for solving vehicle routing problems with time windows and synchronized visit constraints in prefabricated systems, *J. Clean. Prod.*, vol. 250, p. 119464, 2020.

[21] G. Mweshi and N. Pillay, An improved grammatical evolution approach for generating perturbative heuristics to solve combinatorial optimization problems, *Expert Syst. Appl.*, vol. 165, p. 113853, 2021.

[22] J. Lin, Z. J. Wang, and X. Li, A backtracking search hyper-heuristic for the distributed assembly flow-shop scheduling problem, *Swarm Evol. Comput.*, vol. 36, pp. 124–135, 2017.

[23] S. Y. Zhang, Z. L. Ren, C. X. Li, and J. F. Xuan, A perturbation adaptive pursuit strategy based hyper-heuristic for multi-objective optimization problems, *Swarm Evol. Comput.*, vol. 54, p. 100647, 2020.

[24] L. Ahmed, C. Mumford, and A. Kheiri, Solving urban transit route design problem using selection hyper-heuristics, *Eur. J. Oper. Res.*, vol. 274, no. 2, pp. 545–559, 2019.

[25] E. Kieffer, G. Danoy, M. R. Brust, P. Bouvry, and A. Nagih, Tackling large-scale and combinatorial bi-level problems with a genetic programming hyper-heuristic, *IEEE Trans. Evol. Comput.*, vol. 24, no. 1, pp. 44–56, 2020.

[26] A. Aslan, I. Bakir, and I. F. A. Vis, A dynamic thompson sampling hyper-heuristic framework for learning activity planning in personalized learning, *Eur. J. Oper. Res.*, vol. 286, no. 2, pp. 673–688, 2020.

[27] S. N. Chaurasia and J. H. Kim, An evolutionary algorithm based hyper-heuristic framework for the set packing problem, *Inf. Sci. (Ny).*, vol. 505, pp. 1–31, 2019.

[28] N. R. Sabar and G. Kendall, Population based Monte Carlo tree search hyper-heuristic for combinatorial optimization problems, *Inf. Sci. (Ny).*, vol. 314, pp. 225–239, 2015.

[29] N. R. Sabar, M. Ayob, G. Kendall, and R. Qu, Automatic design of a hyper-heuristic framework with gene expression programming for combinatorial optimization problems, *IEEE Trans. Evol. Comput.*, vol. 19, no. 3, pp. 309–325, 2015.

[30] A. Kheiri and E. Özcan, An iterated multi-stage selection hyper-heuristic, *Eur. J. Oper. Res.*, vol. 250, no. 1, pp. 77–90, 2016.

[31] V. A. de S. Júnior, E. Özcan, and V. R. de Carvalho, Hyper-heuristics based on reinforcement learning, balanced heuristic selection and group decision acceptance, *Appl. Soft Comput.*, vol. 97, p. 106760, 2020.

[32] J. H. Drake, A. Kheiri, E. Özcan, and E. K. Burke, Recent advances in selection hyper-heuristics, *Eur. J. Oper. Res.*, vol. 285, no. 2. pp. 405–428, 2020.

[33] M. M. Ferdaus, F. Zaman, and R. K. Chakrabortty, Performance improvement of a parsimonious learning machine using metaheuristic approaches, *IEEE Trans. Cybern.*, doi: 10.1109/TCYB.2021.3051242.

[34] W. B. Yates and E. C. Keedwell, Offline learning with a selection hyper-heuristic: An application to water distribution network optimisation, *Evol. Comput.*, vol. 29, no. 2, pp. 187–210, 2021.

[35] S. W. Cai and Z. D. Lei, Old techniques in new ways: Clause weighting, unit propagation and hybridization for maximum satisfiability, *Artif. Intell.*, vol. 287, p. 103354, 2020.

[36] W. H. El-Ashmawi and D. S. Abd Elminaam, A modified squirrel search algorithm based on improved best fit heuristic and operator strategy for bin packing problem, *Appl. Soft Comput.*, vol. 82, p. 105565, 2019.

[37] G. C. Wang, L. Gao, X. Y. Li, P. G. Li, and M. F. Tasgetiren, Energy-efficient distributed permutation flow shop scheduling problem using a multi-objective whale swarm algorithm, *Swarm Evol. Comput.*, vol. 57, p. 100716, 2020.

[38] J. Y. Shiau, M. K. Huang, and C. Y. Huang, A hybrid personnel scheduling model for staff rostering problems, *Mathematics*, vol. 8, no. 10, p. 1702, 2020.

[39] A. Gharehgozli, C. Xu, and W. D. Zhang, High multiplicity asymmetric traveling salesman problem with feedback vertex set and its application to storage/retrieval system, *Eur. J. Oper. Res.*, vol. 289, no. 2, pp. 495–507, 2021.

[40] B. B. Pan, Z. Z. Zhang, and A. Lim, Multi-trip time-

dependent vehicle routing problem with time windows, *Eur. J. Oper. Res.*, vol. 291, no. 1, pp. 218–231, 2021.

[41] C. P. Almeida, R. A. Goncalves, S. Venske, R. Lüders, and M. Delgado, Hyper-heuristics using multi-armed bandit models for multi-objective optimization, *Appl. Soft Comput.*, vol. 95, p. 106520, 2020.

[42] J. Y. Xu, L. J. Chen, and O. Tang, An online algorithm for the risk-aware restless bandit, *Eur. J. Oper. Res.*, vol. 290, no. 2, pp. 622–639, 2021.

Fuqing Zhao received the BS and PhD degrees from Lanzhou University of Technology, Lanzhou, China in 1994 and 2006, respectively. Since 1998, he has been at the School of Computer and Communication, Lanzhou University of Technology, where he became a full professor in 2012. He was a postdoctoral fellow at the State Key Laboratory of Manufacturing System Engineering, Xi'an Jiaotong University, Xi'an, China from 2009 to 2011. He was a visiting scholar at Exeter Manufacturing Enterprise Center, Exeter University, Exeter, UK from 2008 to 2009, and the Georgia Tech Manufacturing Institute, Georgia Institute of Technology, Atlanta, USA from 2014 to 2015. He has authored two academic books and over 50 refereed papers. His current research interests include intelligent optimization and scheduling.

Shilu Di received the BEng degree in computer and communication technology from Lanzhou University of Technology, Lanzhou, China in 2018. He is currently a master student at the School of Computer and Communication, Lanzhou University of Technology. His research interests include intelligent optimization and scheduling algorithms.

Jie Cao received the BEng degree from Lanzhou University of Technology in 1987, and the MS degree from Xi'an Jiaotong University, China in 1994. She is a professor at the School of Computer and Communication, Lanzhou University of Technology. She had published more than 150 refereed papers, and her research interests include information fusion theory and application, intelligent information processing, and intelligent transportation system theory and application.

Jianxin Tang received the MEng degree from Lanzhou University of Technology in 2012, and the PhD degree from Lanzhou University, China in 2019. He is an associate professor at the School of Computer and Communication, Lanzhou University of Technology. He has published more than 10 refereed papers, and his research interests include social network analysis, intelligent computing, and scheduling theory and optimization.

Jonrinaldi received the PhD degree from University of Exeter, Exeter, UK in 2012. He is now the chairman at the Department of Industrial Engineering, Universitas Andalas, Indonesia. He has published more than 40 refereed papers, and his research interests include logistics, inventory management, production/operations, and supply chain optimization.