Expert Systems With Applications 260 (2025) 125424

Contents lists available at ScienceDirect

Expert Systems With Applications

journal homepage: www.elsevier.com/locate/eswa

Knowledge-enhanced multidimensional estimation of distribution hyper-heuristic evolutionary algorithm for semiconductor final testing scheduling problem

Zi-Qi Zhang [a,b,c], Xing-Han Qiu [a,b], Bin Qian [a,b,c,*], Rong Hu [a,b], Ling Wang [d], Jian-Bo Yang [e]

[a] *School of Information Engineering and Automation, Kunming University of Science and Technology, Kunming 650500, PR China*
[b] *The Higher Educational Key Laboratory for Industrial Intelligence and Systems of Yunnan Province, Kunming University of Science and Technology, Kunming 650500, PR China*
[c] *Yunnan Key Laboratory of Artificial Intelligence, Kunming University of Science and Technology, Kunming 650500, PR China*
[d] *Department of Automation, Tsinghua University, Beijing 100084, PR China*
[e] *Alliance Manchester Business School, The University of Manchester, Manchester M15 6PB, United Kingdom*

ARTICLE INFO

Keywords:
MEDA
Hyper-heuristic
Semiconductor final testing
High-level strategy
Low-level heuristic

ABSTRACT

The semiconductor final test scheduling problem (SFTSP), recognized as a crucial bottleneck in the semiconductor production process, holds immense significance for improving both quality control and scheduling efficiency within chip and integrated circuit enterprises. This article introduces the knowledge-enhanced multidimensional estimation of distribution hyper-heuristic evolutionary algorithm (KMEDHEA) for addressing the SFTSP with the aim of minimizing the makespan. First, a single-vector encoding scheme is used to represent feasible solutions, and a problem-specific constrained-separable left-shift decoding scheme is devised to transform these solutions into feasible scheduling schedules. Second, eight simple yet effective heuristics with problem-specific knowledge are developed that served as a suite of low-level heuristics (LLHs) for exploring the problem solution space. Third, the multidimensional estimation of distribution algorithm (MEDA) is employed as the high-level strategy to estimate the correlations and connections of the pre-designed LLHs, thereby guiding the search scope towards high-quality individuals. Finally, critical configurations of parameters are systematically analyzed by conducting a design-of-experiment (DOE) approach. Numerical experiments are conducted on well-known benchmark datasets, and the experimental results demonstrate the superiority of the KMEDHEA versus several state-of-the-art approaches. The best-known solutions are updated for nine out of ten benchmark instances, highlighting the effectiveness and efficiency of the proposed KMEDHEA in solving the SFTSP.

1. Introduction

As the cornerstone of the modern electronics industry, the semiconductor sector exerts an essential impact on promoting economic efficiency and technical advances. Semiconductor chips serve as the core components in various electronic devices, with their applications burgeoning across domains such as green energy, satellite communication, robotic devices, smart diagnosis, and autonomous driving (Hao, et al., 2014). The semiconductor production process consists of multiple stages, such as wafer fabrication, wafer probing, packaging, and final testing (FT), each involving various tests to ensure quality and reliability, as shown in Fig. 1. Semiconductor FT (SFT) as the last critical stage requires suitable scheduling strategies to optimize both efficiency and quality. Due to both complexity and flexibility, SFT scheduling poses considerable challenges; however, implementing superior scheduling can significantly strengthen competitiveness by reducing costs and refining efficiency for integrated circuit enterprises. Recent research on the SFT scheduling problem (SFTSP) and solution techniques enables enterprises to fine-tune production processes, enhance product quality, and elevate resource utilization. Although the SFTSP has received widespread attention, there remains a need for further expansion of relevant theories and methodologies. Thus, it is imperative to develop

* Corresponding author.
E-mail addresses: zhangziqi@kust.edu.cn (Z.-Q. Zhang), qxh@stu.kust.edu.cn, 565300983@qq.com (X.-H. Qiu), bin.qian@kust.edu.cn (B. Qian), ronghu_da@kust.edu.cn, ronghu@vip.163.com (R. Hu), wangling@tsinghua.edu.cn (L. Wang), jian-bo.yang@manchester.ac.uk (J.-B. Yang).

https://doi.org/10.1016/j.eswa.2024.125424
Received 27 March 2024; Received in revised form 18 September 2024; Accepted 19 September 2024
Available online 24 September 2024

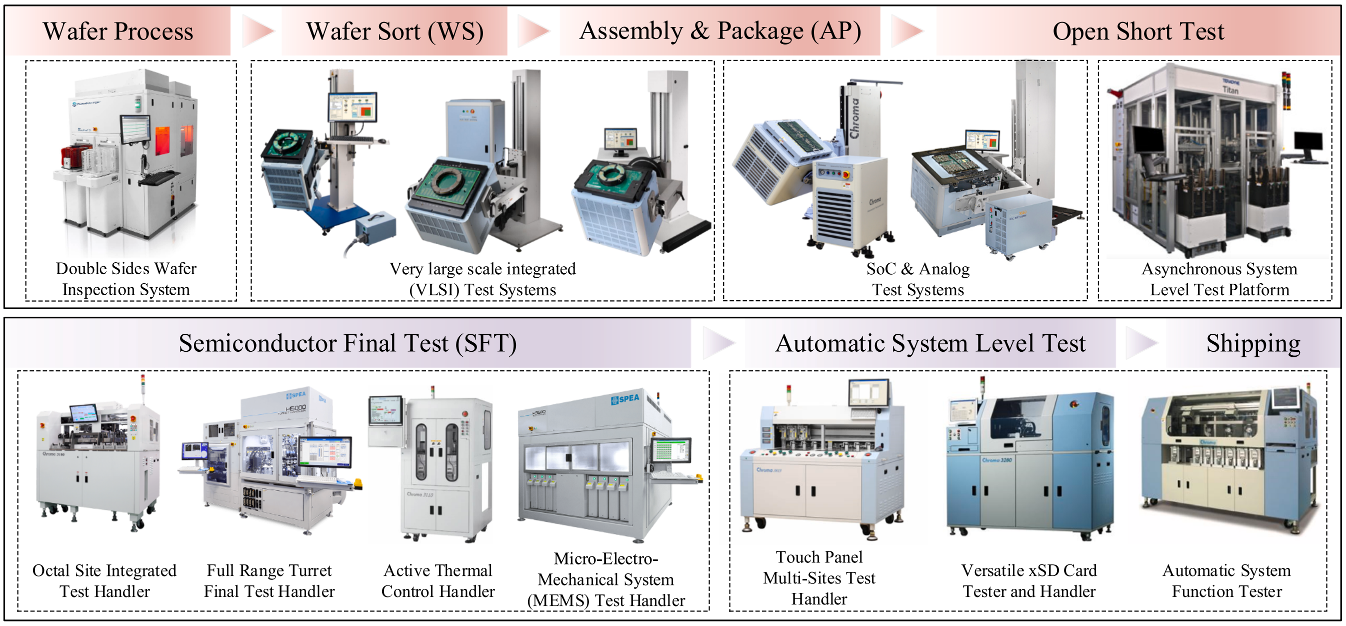

Wafer Process
Wafer Sort (WS)
Assembly & Package (AP)
Open Short Test
Double Sides Wafer Inspection System
Very large scale integrated (VLSI) Test Systems
SoC & Analog Test Systems
Asynchronous System Level Test Platform
Semiconductor Final Test (SFT)
Automatic System Level Test
Shipping
Octal Site Integrated Test Handler
Full Range Turret Final Test Handler
Active Thermal Control Handler
Micro-Electro-Mechanical System (MEMS) Test Handler
Touch Panel Multi-Sites Test Handler
Versatile xSD Card Tester and Handler
Automatic System Function Tester

Fig. 1. Schematic diagram of semiconductor testing stages.

effective and efficient methods for SFTSP, given the significant surge in research interests, to achieve practical engineering applications and economic benefits.

The semiconductor production process can be divided into two main stages: the front-end-of-line (FEOL) and the back-end-of-line (BEOL), each consisting of several critical steps (Dass, et al., 2022; Kim, et al., 2021; Lee, et al., 2023). The FEOL process involves wafer fabrication and inspection, where wafers undergo rigorous testing to detect defective circuits. Following photolithography and etching, wafers are diced to isolate chip circuits, which are encapsulated and wired, resulting in pristine chips free of defects. The BEOL process plays a pivotal role in determining the performance and functionality of the chips, which undergo a series of FTs to ensure the required quality and standard specifications. However, the BEOL process poses significant challenges, as handlers and testers incur substantial costs and face limitations in terms of available accessories or resources, resulting in severe bottlenecks. Hence, SFTSP has emerged as a frontier hotspot with great research interest and potential. Recent studies have revealed that the SFTSP can be reduced to a flexible JSP (FJSP) with multi-resource constraints, thereby proving to be strongly NP-hard (Wu, et al., 2008). To describe the complexity of the SFTSP, the three-field notation $\alpha|\beta|\gamma$ is used, where α represents the shop environment and β and γ represent the constraints and criteria, respectively. The SFTSP with the objective of minimizing makespan is denoted as $FJ|MRs,STs|C_{\max}$. Here, FJ refers to a flexible job shop. MRs and STs represent multi-resource constraints and setup times, and $C_{\max}$ is the makespan criterion. Since $FJ||C_{\max}$ has been proven to be NP-hard problem (Fattahi, et al., 2007), subsequent studies by Mousakhani, (2013) and Lei, et al., (2014) further confirmed the NP-hardness of both $FJ|STs|C_{\max}$ and $FJ|MRs|C_{\max}$. Given that $FJ|MRs,STs|C_{\max}$ is much more complex than $FJ|STs|C_{\max}$ and $FJ|MRs|C_{\max}$, and it can be reduced to either of them, SFTSP is still strongly NP-hard problem (Wu, et al., 2008). This inherent complexity implies the formidable challenge of finding optimal solutions, particularly for tackling large-scale problems, due to the considerable computational complexity (CC). Consequently, the modeling and solving of the SFTSP present a challenging endeavor, holding quite importance in both practical applications and academic research. Over the past few decades, significant scholarly studies have emerged (Cao, et al., 2019; Chen, et al., 1994; Freed, et al., 1999; Hao, et al., 2014; Hu, et al., 2023; Huanxin Henry, et al., 1998; Lin, et al., 2022; Lin, et al., 2004; Ovacik, et al., 1996; Pearn, et al., 2004; Sang, et al., 2018; Uzsoy, et al., 1993; Uzsoy, et al., 1992; Uzsoy, et al., 1991; Wang, et al., 2014, 2015; Wang, et al., 2013; Wu, et al., 2008; Wu, et al., 2012; Zheng, et al., 2014), as summarized in Table 1. However, when dealing with large-scale instances of SFTSP, commonly used mathematical methods such as branch & bound (B&B) (Chen, et al., 1994) become increasingly impractical due to the considerable cost caused by the complexity. Constructive heuristics, while capable of quickly yielding feasible scheduling schemes based on problem properties, often struggle to guarantee the solution's superiority (Huanxin Henry, et al., 1998; Lin, et al., 2004; Pearn, et al., 2004; Uzsoy, et al., 1992). As shown in Table 1, extensive efforts have mainly focused on addressing these challenges through the application of hybrid intelligent optimization algorithms (HIOAs) (Cao, et al., 2019; Hao, et al., 2014; Lin, et al., 2022; Sang, et al., 2018; Wang, et al., 2014, 2015; Wang, et al., 2013; Wu, et al., 2008; Wu, et al., 2012; Zheng, et al., 2014). In contrast to typical mathematical methods and constructive heuristics, HIOAs have the unique advantage of not being constrained by complexity. Through inherent evolutionary mechanisms and search strategies, HIOAs allow the formulation of search operators that are dependent on problem-specific knowledge. This renders their search behaviors versatile, well-suited for tackling complex constraints, and proficient in achieving high-quality solutions within a reasonable timeframe.

As a novel emerging paradigm, hyper-heuristic evolutionary algorithms (HHEAs) have gained significant attention in recent years. HHEAs enable employing high-level strategy (HLS) to modulate low-level heuristics (LLHs), aiming to surpass the limitations of HIOAs and effectively tackle complex problems via interactive interconnections and collaborative searches among various LLHs (Zhang, et al., 2023). The significant distinction between existing HIOAs and HHEAs lies in search behaviors: while HIOAs directly explore the solution space, HHEAs operate in the strategy space of LLHs, guided by HLSs. Well-designed HLSs facilitate the most suitable sequencing of LLHs, while the ordered execution of LLHs enables efficient exploration of the solution space, thereby seeking superior solutions (Branke, et al., 2016; Shang, et al., 2022). Due to their strengths in suitably selecting and adaptively adjusting search strategies, HHEAs have self-learning skills, reducing the reliance on problem-specific knowledge. Consequently, HHEAs have shown success in solving various NP-hard problems, such as engineering optimization and scheduling problems. Recent studies have emerged

Table 1
Review of recent related works for SFTSP.

Author(s)	Objective(s)	Proposed approach(es)
Uzsoy, et al. (1991)	Makespan	Shifting bottleneck approach; Production scheduling algorithms for semiconductor test operations.
Uzsoy, et al. (1992)	Maximum lateness with dynamic arrivals and number of tardy jobs	Constructive heuristics.
Uzsoy, et al. (1993)	Performance of dispatching rules	Several efficient dispatching rules.
Chen, et al. (1994)	Makespan	Lagrangian relaxation; Subgradient direction method.
Ovacik, et al. (1996)	Makespan	Decomposition methods; Specialized strategies.
Huanxin Henry, et al. (1998)	Makespan	Two Petri net-based hybrid heuristic strategies.
Freed, et al. (1999)	Makespan	Designed an enumeration solution method.
Pearn, et al. (2004)	Minimum total machine workload	Three efficient network algorithms.
Lin, et al. (2004)	Maximum committed volume performance	Capacity-constrained approach based on the theory of constraints.
Wu, et al. (2008)	Makespan	Mixed integer linear programming; Genetic algorithm (GA).
Wu, et al. (2012)	Makespan	Bi-vector encoding genetic algorithm (bvGA).
Wang, et al. (2013)	Makespan	Hybrid estimation of distribution algorithm (HEDA).
Hao, et al. (2014)	Makespan	Cooperative estimation of distribution algorithm (CEDA).
Wang, et al. (2014)	Makespan	Compact estimation of distribution algorithm (cEDA).
Zheng, et al. (2014)	Makespan	Novel fruit fly optimization algorithm (nFOA).
Wang, et al. (2015)	Makespan	Knowledge-based multi-agent evolutionary algorithm (KMEA).
Sang, et al. (2018)	Makespan	Cooperative coevolutionary invasive weed optimization (CCIWO) algorithm.
Cao, et al. (2019)	Makespan	Cuckoo search algorithm with reinforcement learning (CSRL).
Lin, et al. (2022)	Makespan	*Q*-learning-based hyper-heuristic (QHH).
Hu, et al. (2023)	Makespan	Greedy-based crow search algorithm (GCSA).

applying HHEAs for shop scheduling problems, including SMSP (Wu, et al., 2021), dynamic JSPs (Fan, et al., 2021; Park, et al., 2018), flexible JSPs (Lim, et al., 2022; Lin, et al., 2019; Zhang, et al., 2023), RCPSPs (Chen, et al., 2021; Chen, et al., 2022; Zhu, et al., 2021), DAPFSPs (Lin, et al., 2017; Song, et al., 2021; Song, et al., 2023), DABFSPs (Zhang, et al., 2023; Zhao, et al., 2022), and SFTSP (Lin, et al., 2022). Although HHEAs are expected to be promising paradigms for solving complex, large-scale problems, their effectiveness is directly dependent on the design of HLSs, which enable the interaction and coordination of LLHs to achieve excellent search efficacy. However, the design of HLS confronts certain challenges. Firstly, typical HLSs may exhibit insensitivity to the importance of LLHs or may struggle with the huge and complex characteristics of the strategy space, thus hindering the development of effective learning mechanisms. Additionally, there is a risk that potential patterns in the sequences of LLHs may be destroyed or that critical characteristics may not be accurately captured, especially for GP-based HHEAs. Furthermore, the adaptability and versatility of HLSs require consideration. Therefore, research efforts have been dedicated to designing efficient HLSs that are capable of effectively guiding the generation of LLH sequences while inheriting their inherent patterns.

Estimation of distribution algorithms (EDAs) as learning-based paradigms have been a prominent research hotspot in the fields of statistical learning and evolutionary computation. EDAs aim to estimate the distribution characteristics and correlations of solutions from the macro perspective by establishing probabilistic models and extrapolating prospective patterns of high-quality solutions, driving the search behavior through the collaborative interactions of sampling strategies and updating mechanisms to seek superior solutions. Recent decades have witnessed widespread applications of EDAs in solving scheduling problems, including PFSP (Jarboui, et al., 2009), two-stage FSP (Liu, et al., 2018), multi-objective PFSP (Tiwari, et al., 2015), lot-streaming FSP (Pan, et al., 2012), FJSP (Wang, et al., 2012), and quay crane scheduling problem (QCSP) (Expósito-Izquierdo, et al., 2013). To the best of our knowledge, relevant research on applications of EDAs to HHEAs is still scarce. Existing EDA-based HHEAs commonly attempt to employ one or more two-dimensional (2-D) probabilistic models to represent the pattern characteristics of LLHs (Song, et al., 2023; Zhao, et al., 2023). Here, the patterns refer to the block structures and block distributions, where each block consists of any two consecutive LLH pairs in sequences of heuristics. However, these 2-D probabilistic models encounter challenges in accurately capturing the positional details and distributional tendencies of LLHs. As a result, critical blocks may be misplaced in the newly generated heuristic sequences during sampling, thus constraining the effectiveness of EDA-based HHEAs. In contrast, multidimensional probabilistic models provide richer feature extraction and much more accurate correlations, which can easily recognize and represent potential patterns, showing significant strengths in characterizing the positions and interconnections of blocks (Zhang, et al., 2022; Zhang, et al., 2021; Zhang, et al., 2022). Motivated by these insights, a knowledge-enhanced multidimensional estimation of distribution hyper-heuristic evolutionary algorithm (KMEDHEA) is developed for addressing the SFTSP. In KMEDHEA, eight simple yet efficient heuristics are crafted to create pools of LLHs, while the multidimensional EDA (MEDA) serves as the learning-based paradigm to manipulate problem-specific LLHs. Specifically, MEDA-based HLS employs the 3-D probabilistic model to generate heuristic sequences through special sampling strategies (see Section 4.4), and executes specific sorted LLHs to steer search scopes toward promising regions within the search space of problem solutions. This 3-D probabilistic model is updated by an efficient update mechanism (see Subsection 4.2.3) to reasonably reserve valuable information extracted from superior sequences of LLHs.

The innovative contributions of this article are as follows:

- A novel operation-based sequence model is formulated for SFTSP, problem properties are analyzed, and a knowledge-enhanced multidimensional estimation of distribution hyper-heuristic evolutionary algorithm (KMEDHEA) is proposed aiming at minimizing the makespan.
- A problem-dependent hybrid initialization method (HIM) based on three problem-specific heuristic rules is developed to improve the quality and diversity of the initial population.
- A single-vector encoding scheme and an effective constrained-separable left-shift decoding scheme are designed based on multi-resource constraints to represent feasible scheduling solutions and transform these solutions into suitable scheduling schemes and schedules.
- A suite of problem-specific heuristics is developed that act as pools of LLHs dedicated to exploring the solution space of the problem and executing fine-grained searches. In addition, a simulated annealing (SA)-based acceptance mechanism is introduced to enhance the ability to jump out of local optima.
- As a prospective learning-based paradigm, the multidimensional EDA (MEDA) is applied as the HLS in EDA-based HHEAs, which manipulates a set of easy-to-implement LLHs and accurately records the block structure and block distribution of LLHs through the use of 3-D probabilistic models with specific sampling strategies and updating mechanisms.
- Numerical experiments are conducted on well-known public datasets, and ablation studies show the effectiveness of the critical

components of the KMEDHEA; in particular, the MEDA-based HLS is effective due to its strengths in learning ordinal correlation and positional connection of the LLHs, while computational results demonstrate the superiority of the KMEDHEA, with the best-known scheduling solutions updated for 9 out of 10 benchmark instances.

The subsequent sections of this article are organized as follows. Section 2 provides a comprehensive review of pertinent literature. Section 3 introduces the SFTSP and formulates an operation-based sequence model for it. Section 4 details the implementation of KMEDHEA, and Section 5 provides and discusses the computational evaluation and comparison results. Finally, Section 6 presents concluding remarks and outlines future research directions.

2. Literature review

2.1. Related work on SFTSP

Over the past decades, SFTSP has attracted significant attention in both the fields of smart manufacturing and evolutionary computation. Pioneering work was studied by Uzsoy, et al., (1991), in which a foundational framework for modeling semiconductor test operations was introduced using the shifting bottleneck approach. They divided the facility or shop into distinct work centers and adopted a disjunctive graph representation to capture inherent interactions between work centers. Uzsoy, et al., (1992) treated SFTSP as a variant of SMSP with setup times. They devised dynamic programming (DP) aimed at minimizing maximum tardiness, taking into account dynamic arrivals and the number of tardy jobs. Subsequently, Uzsoy, et al., (1993) evaluated the effectiveness of several scheduling strategies for SFT processes, considering the impacts of job arrival patterns and uncertainties. Experimental results revealed findings that a single strategy hardly performed well across metrics simultaneously. Chen, et al., (1994) modeled the SFTSP as JSPs with simultaneous resources. They introduced a Lagrangian relaxation (LR) approach to address both resource and precedence constraints. The results indicated that the LR method outperformed DP-based heuristics proposed by Uzsoy, et al., (1992). Ovacik, et al., (1996) developed decomposition methods that segmented SFT facilities into various types of work centers. They devised specific procedures for multiple work centers, and the results confirmed the viability of these decomposition procedures. Huanxin Henry, et al., (1998) adopted PetriNets to represent lot sizes of jobs, resource and precedence constraints, concurrent activities, and flexible routes. In Petri nets, they proposed two hybrid search strategies—a best-first heuristic strategy and a controlled backtracking strategy—to find optimal or near-optimal schedules. Freed, et al., (1999) generalized the SFTSP as a multi-head tester scheduling problem. To address this extended problem, they designed an enumerative procedure that considered the efficiency of utilizing various numbers of test heads. Pearn, et al., (2004) further investigated the multistage SFTSP with reentry and attempted to develop three efficient heuristics aimed at minimizing machine workloads. Lin, et al., (2004) developed discrete event simulation models based on e-M$-$Plant$^{\text{TM}}$ to implement suitable strategies via capacity-constrained rules. The results indicated that capacity-constrained rules outperformed other rules in terms of committed volume performance.

Recent research has predominantly focused on developing HIOAs to seek superior solutions for SFTSPs. Wu, et al., (2008) pioneered formulating the MILP model based on crucial characteristics, employing an assignment rule to determine machine configurations and allocate resources, and proposing a problem-specific GA. Wu, et al., (2012) delved deeper into SFTSP, presenting a bi-vector coding-based GA (bvGA) capable of capturing crucial characteristics of dynamic configurations of machines across multiple resources and setup times between operations. Subsequent contributions by Wang, et al., (2013) introduced a hybrid EDA (HEDA), combining both sampling strategies and updating mechanisms of a 2-D probabilistic model with local searches to stress global exploration and local intensification. Hao, et al., (2014) extended a co-evolutionary framework with a cooperative EDA (CEDA), incorporating a divide-and-conquer strategy. Experimental results verified that the CEDA outperformed both GA (Wu, et al., 2008) and bvGA (Wu, et al., 2012), albeit with discernible trade-offs in computational costs. Seeking superior solutions while striking a balance between search efficiency and cost prompted further improvements. Wang, et al., (2014) introduced a compact EDA (cEDA) that used operation-based sequence encoding and decoding schemes and a 2-D probabilistic model to characterize the distribution of the blocks in superior solutions. Zheng, et al., (2014) designed a novel fruit fly optimization algorithm (nFOA), emphasizing exploitation via a cooperative search mechanism by smell- and vision-based operators. Computational evaluation demonstrated the superiority of nFOA with less cost. Wang, et al., (2015) proposed a knowledge-based multi-agent evolutionary algorithm (KMEA). In KMEA, feasible solutions act as agents engaged in mutual learning and competition through problem-specific search operators. Furthermore, the cooperative co-evolutionary invasive weed optimization (CCIWO) was proposed by Sang, et al. (2018). CCIWO utilized two coupled colonies: one dedicated to handling the machine allocation problem and the other focused on the operation sequence problem. The results demonstrated that CCIWO outperformed several state-of-the-art algorithms, showcasing the potential of cooperative strategies. However, it is noteworthy that HIOAs for SFTSP are highly sensitive to parameters, which makes the selection of proper parameters a non-trivial task. To address this concern, Cao, et al., (2019) introduced a cuckoo search with reinforcement learning (CSRL). CSRL used an RL-based parameter control scheme to enhance efficiency, balance diversification and intensification, and evaluate solution rankings through the surrogate model, effectively reducing computational costs. Lin, et al., (2022) designed a Q-learning-based hyper-heuristic (QHH) that incorporated a left-shift strategy to improve resource utilization, marking the first report of Q-learning-based HHEA in solving SFTSP. Further innovations continued with Hu, et al., (2023) who introduced a greedy-based crow search algorithm (GCSA), enhancing its adaptability to SFTSP through revamped crow position updating strategies called "track" and "hover." Upon comprehensive efforts review, it is evident that existing approaches to addressing SFTSP primarily revolve around HIOAs, with limited emphasis on mathematical methods and constructive heuristics. Mathematical methods, despite their theoretical soundness, suffer from considerable computational costs, limiting their practical applicability. Constructive heuristics can provide solutions rapidly but may fall short in terms of solution quality. In contrast, HIOAs exhibit significant superiority by virtue of their efficient evolutionary mechanisms, specific search strategies, and effective neighborhood operators. Therefore, the emergence of HHEAs presents promising insights to address the complexity and uncertainty with limited resources. This flexibility and adaptivity make them indispensable in tackling the challenges posed by SFTSP.

2.2. Related work on HHEAs

As promising paradigms in EAs, HHEAs have emerged as efficient and automatic search methodologies for generating or selecting heuristics to address real-world problems. HHEAs showcase significant strengths in dynamically creating and adaptively coordinating LLHs through HLSs, thereby fully utilizing the potential of heuristics. This autonomy and adaptability allow HHEAs to effectively explore the search space for problem solutions, facilitating the search for superior solutions. A systematic review conducted by Burke, et al., (2013) has categorized HHEAs into two main methodologies: heuristic generation and heuristic selection. The former involves the development of LLHs tailored to problem characteristics, while the latter focuses on the design of HLSs that adaptively select and apply LLHs in specific scenarios, providing guidance on search directions via search states and search

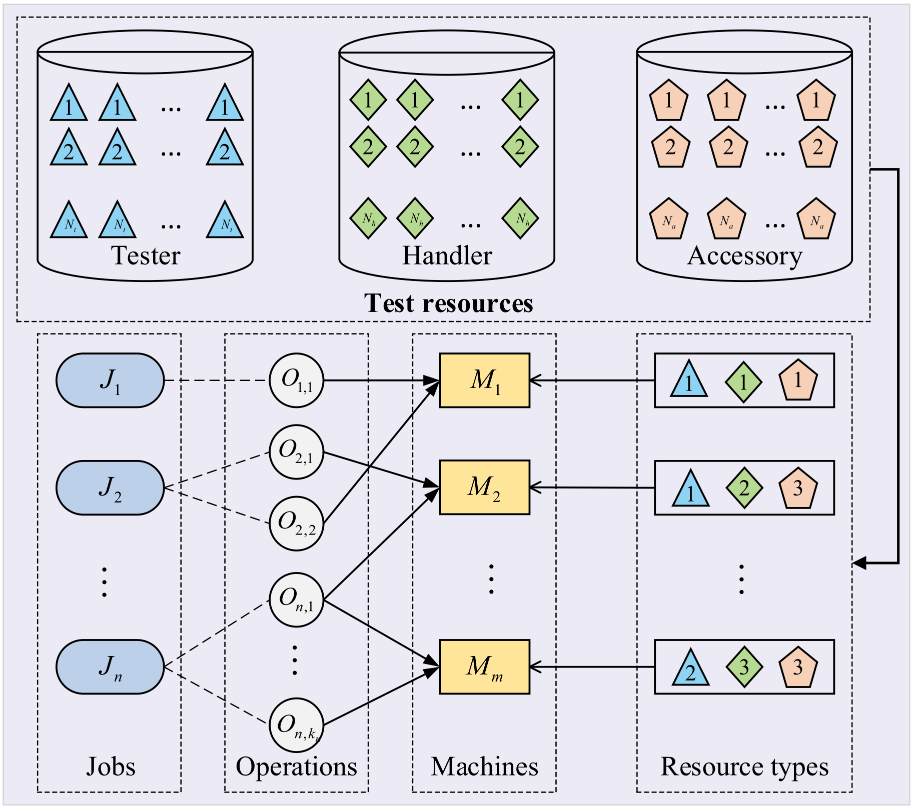


Fig. 2. Illustration of the SFTSP.

experience (Zhang, et al., 2023). Among the methodologies used in HHEAs, GP-based HLSs are the most commonly adopted for heuristic generation, with extensive efforts applied to various problems (Kieffer, et al., 2020; Sabar, et al., 2013; Song, et al., 2021). Park, et al., (2018) proposed an ensemble GP-based hyper-heuristic (GP-HH) for addressing dynamic JSP, aiming at minimizing mean tardiness while considering dynamic arrivals. This study showed that weighted combination schemes performed poorly when evolving ensembles of scheduling schemes. Chen, et al., (2022) proposed a hyper-heuristic-based two-stage GP (TGP-HH) to solve the stochastic resource-constrained multi-project scheduling problem. TGP-HH decomposed the evolutionary process into generation and selection stages, using the strengths of GP-based HLSs to develop multiple suitable scheduling strategies. Furthermore, Kieffer, et al., (2020) extended the application of GP-HH to train efficient greedy heuristics for optimizing cloud pricing problems, aiming at achieving the best deal between service providers and customers. Song, et al., (2021) employed GP-HH to address DAPFSP with sequence-dependent setup times. They developed ten efficient heuristics and used the GP-based HLS to produce heuristic sequences. However, it is worth noting that existing GP-based HLSs suffer from distinct drawbacks: exponential increases in CC as the search space expands, both in terms of the number of rule components and the quantity of rules created. As a result, heuristic selection-based HHEAs have witnessed applications due to both efficiency and effectiveness in various scheduling problems. Sabar, et al., (2015) developed a Monte Carlo tree search-based HHEA, which modeled the search space of LLHs as a tree structure and exploited Monte Carlo tree search as the HLS for efficient exploration, which exhibited excellent efficacy in solving various problems, including FSPs, TSP, and VRPs with time windows. Asta, et al., (2016) developed a tensor-based HHEA for tackling the nurse rostering problem, combining data science techniques with knowledge extraction by tensor analysis, enabling efficient problem-solving and self-improving methods. Lin, et al., (2017) devised a backtracking search-based hyper-heuristic (BS-HH) to address DAPFSP. In BS-HH, ten effective heuristics were devised to form a set of LLHs, and a backtracking search strategy served as the HLS, orchestrating LLHs in pursuit of seeking the best order of LLHs. Furthermore, Song, et al., (2023) presented a hyper-heuristic-based memetic algorithm (HH-MA) aimed at addressing DAPFSP with the objective of minimizing makespan. In HH-MA, EDA-based HLS was used for global exploration while incorporating a critical-products-based referenced local search to enhance local exploitation. Lim, et al., (2022) provided a simulated-annealing-based hyper-heuristic (SA-HH) for FJSP. SA-HH constructed heuristics comprising machine assignment rules (MARs) and job sequencing rules (JSRs) based on state features. Recent advances include the *Q*-learning-based HHEA (QLHHEA) proposed by Zhang, et al., (2023) to solve DABFSP. In QLHHEA, LLHs were treated as states, and state selections were regarded as actions. *Q*-learning served as HLS to determine the best correlations of LLHs and select suitable sequences of LLSs. Zhang, et al., (2023) further proposed an innovative *Q*-learning-based HHEA (QHHEA) for distributed FJSP with crane transportation, applying *Q*-learning as HLS to assist HHEA in identifying suitable LLH for each state via feedback recorded in *Q* tables. Zhao, et al., (2022) devised a constructive heuristic HHNRa and designed a self-learning hyper-heuristic (SL-HH) to tackle the DABFSP. In SL-HH, a self-learning HLS was devised, utilizing historical success rates of LLHs to dynamically adapt and refine the search behaviors. Mahmud, et al., (2022) introduced a self-adaptive, multi-operator, and multi-objective HHEA (SA(MO)$_2$H) for integrated scheduling, devising four solution-updating heuristics to enhance LLH efficacy, coupled with *Q*-learning-based HLS to guide the search for LLHs. Interested readers are advised to refer to the study of Zhang, et al., (2023) for recent advances and applications of HHEAs.

Despite significant strides in studies of HLSs in HHEAs, challenges persist, including reliance on problem-specific domain knowledge, limited generality in HLSs, and relatively high computational costs. To address these challenges, the recently emerged MEDA is recognized as a promising paradigm for HLS. MEDA holds the potential to overcome existing limitations in HLS design and enhance the efficacy of HHEAs.

Table 2
Symbols and explanations.

Notation	Description
Indices	
i	Index of jobs, *i.e.*, $i = 1, 2, \ldots, n$
j	Index of machines, *i.e.*, $j = 1, 2, \ldots, m$
k	Index of operations for job J_i, *i.e.*, $k = 1, 2, \ldots, k_i$
t	Index of time, *i.e.*, $t \geqslant 0$
Sets	
J	Set of jobs, *i.e.*, $J = \{J_1, J_2, \ldots, J_n\}$
M	Set of machines, *i.e.*, $M = \{M_1, M_2, \ldots, M_m\}$
T	Set of tester resources, *i.e.*, $T = \{T_1, T_2, \ldots, T_{N_t}\}$
H	Set of handle resources, *i.e.*, $H = \{H_1, H_2, \ldots, H_{N_h}\}$
A	Set of accessory resources, *i.e.*, $A = \{A_1, A_2, \ldots, A_{N_a}\}$
$EM_{i,k}$	Set of eligible machines for the processing of $O_{i,k}$
R	Set of total resources, *i.e.*, $R = \{T, H, A\}$
Parameters	
n	Number of jobs
m	Number of machines
N_t	Number of types for testers
N_h	Number of types for handles
N_a	Number of types for accessories
k_i	Number of operations for job J_i
TO	Total number of operations for jobs, *i.e.*, $TO = \sum_{i=1}^{n} k_i$
$O_{i,k}$	The kth operation of job J_i
π_{job}	The sequence of operations, *i.e.*, $\pi_{job} = \{\pi_{job}(1), \ldots, \pi_{job}(TO)\}$
π_{mac}	The sequence of machines, *i.e.*, $\pi_{mac} = \{\pi_{mac}(1), \ldots, \pi_{mac}(TO)\}$
$\boldsymbol{\pi}$	The feasible scheduling solution, *i.e.*, $\boldsymbol{\pi} = (\pi_{job}, \pi_{mac})$
$M_{i,k}$	The machine for the processing of $O_{i,k}$
M_j^R	The machine for occupying resource R
$S_{i,k}$	The start time for the processing of $O_{i,k}$
$C_{i,k}$	The complete time for the processing of $O_{i,k}$
$N_{j,t}$	Number of jobs in process of machine M_j at time t
$N_{EM_{i,k}}$	Number of eligible machines for the processing of $O_{i,k}$
$R_{M_{i,k}}$	Type of resources required for $M_{i,k}$
$N_{i,k,t}$	Number of completed processing of operation $O_{i,k}$ at time t
N_{T_a}	The number of testers T_a, $a = 1, \ldots, N_t$
N_{H_b}	The number of handles H_b, $b = 1, \ldots, N_h$
N_{A_c}	The number of accessories A_c, $c = 1, \ldots, N_a$
$F_{M_{i,k}}$	The finish time of the last operation on $M_{i,k}$
$F_{M_j^R}$	The finish time of the last operation on M_j^R
$RT_{M_{i,k}}$	The ready time for the required resources of $M_{i,k}$
$ST_{M_{i,k-1}, M_{i,k}}$	The setup time for operations of J_i transferring from $M_{i,k-1}$ to $M_{i,k}$
$PT_{O_{i,k}, M_{i,k}}$	The processing time of $O_{i,k}$ on $M_{i,k}$
$W_{M_{i,k}}^R$	The waiting time for resources of $M_{i,k}$

These insights inspire the impetus to design MEDA-based HHEA to solve the SFTSP. Remarkably, to the best of our authors' knowledge, there has been no previous investigation into the problem-specific KMEDHEA for SFTSP, thus making this work a worthwhile contribution to the field of evolutionary computation.

3. Problem description

Previous studies show that the SFTSP is a complex FJSP with multi-resource constraints and setup times. This problem entails efficient scheduling of operations and effective allocation of test resources and machines to minimize manufacturing periods, thereby maximizing test efficiency while satisfying specific requirements and multiple constraints (Cao, et al., 2019; Lin, et al., 2022; Sang, et al., 2018; Wang, et al., 2015). Unlike mathematical models, which typically involve decision variables, sequence models include a set of equations that enable the calculation of start and end times for operations on machines, with optimization variables represented as operation sequences. Recent research on HIOAs and HHEAs for SFTSPs has focused on sequence models since their evolutionary mechanisms and search strategies are specific to the operation sequences, which are suitable for solving sequence models (Wang, et al., 2015; Hu, et al., 2023). It is crucial to note that in sequence models, individuals or solutions correspond to variables with undetermined sequencing rather than the decision variables used in mathematical models. Essentially, HIOAs are designed to optimize the operation sequences of sequence models, while mathematical solvers are suited for handling decision variables in mathematical models.

For ease of understanding, this section provides a detailed description of SFTSP and presents the operation-based sequence model for SFTSP. The schematic of the SFTSP is illustrated in Fig. 2, and an explanation of the relevant notations is provided in Table 2. There are n jobs that need to undergo testing on m machines. Each job J_i consists of one or more operations, such as electric testing, temperature testing, burn-in, scanning, baking, tape reeling, packaging, and loading, each of which can be tested on one or more machines. Effective SFT scheduling necessitates the collaboration of multiple resources, including testers, handlers, and accessories, each of which can be categorized into different types. Testers are responsible for performing software tests and stimulating equipment to identify defects. Handlers maintain the device at the required temperature for specific tests. Accessories, particularly adapters, provide the interface between the tester and the device. The allocation of testing resources to each machine and the processing time for each job are predetermined. Since devices undergo multiple functional tests on machines at different temperatures, specific setup times are required to prepare for test machines to perform these tests. Note that the setup time is independent of the testing process, which can only begin once all required resources are available, the setup process is done, and the job arrives at the test machine. Furthermore, several predefined constraints must be satisfied. (1) All jobs, machines, and resources are available from the outset. (2) Testing processes for operations cannot be interrupted or preempted. (3) Machines and all types of available resources must be exclusive during the testing process. (4) Operations belonging to the same job must be executed sequentially with specific priorities. (5) Each machine can process only one operation at a time, and each operation can be tested on only one machine at a time. (6) Transportation times are considered negligible. The aim of addressing the SFTSP is to determine the allocation of operations to machines and the sequence of operations on machines so as to minimize the makespan without exceeding the total available resources. Based on this description, the operation-based sequence model of the SFTSP can be formulated as follows:

$$C_{\max}(\boldsymbol{\pi}) = \max\{C_{1,k_1}, C_{2,k_2}, \ldots, C_{n,k_n}\} \tag{1}$$

$$C_{i,k} = S_{i,k} + PT_{O_{i,k}, M_{i,k}}, \forall i \tag{2}$$

$$S_{i,k} = \max\{RT_{M_{i,k}}, F_{M_{i,k}}\}, \forall i, k = 1 \tag{3}$$

$$S_{i,k} = \max\{RT_{M_{i,k}}, F_{M_{i,k}}, C_{i,k-1} + ST_{M_{i,k-1}, M_{i,k}}\}, \forall i, k > 1 \tag{4}$$

$$RT_{M_{i,k}} = \max\{W_{M_{i,k}}^t, W_{M_{i,k}}^h, W_{M_{i,k}}^a\}, \forall i, k \tag{5}$$

$$W_{M_{i,k}}^T = \min\{F_{M_1^{T_a}}, F_{M_2^{T_a}}, \ldots, F_{M_m^{T_a}}\}, \forall a, i, k; N_{T_a} = 0 \tag{6}$$

$$W_{M_{i,k}}^H = \min\{F_{M_1^{H_b}}, F_{M_2^{H_b}}, \ldots, F_{M_m^{H_b}}\}, \forall b, i, k; N_{H_b} = 0 \tag{7}$$

$$W_{M_{i,k}}^A = \min\{F_{M_1^{A_c}}, F_{M_2^{A_c}}, \ldots, F_{M_m^{A_c}}\}, \forall c, i, k; N_{A_c} = 0 \tag{8}$$

Table 3
Processing times and setup times. ('-' indicates that the machine does not meet the operating conditions).

Machine	Processing time									Setup time			
	$O_{1,1}$	$O_{1,2}$	$O_{2,1}$	$O_{2,2}$	$O_{3,1}$	$O_{3,2}$	$O_{4,1}$	$O_{5,1}$	$O_{5,2}$	M_1	M_2	M_3	M_4
M_1	3	–	3	2	4	5	2	2	4	0	1	3	2
M_2	2	3	–	2	–	3	–	2	2	1	0	2	4
M_3	4	2	2	4	4	–	3	–	–	3	2	0	1
M_4	2	3	3	–	6	3	3	4	2	2	4	2	0

Table 4
Number of available resources.

Type	Tester	Handler	Accessory
Type 1	1	1	1
Type 2	1	1	1
Type 3	1	1	1
Type 4	1	1	0

Table 5
Resource configuration of machines.

Machine	Tester	Handler	Accessory
M_1	Type 1	Type 1	Type 3
M_2	Type 1	Type 2	Type 2
M_3	Type 2	Type 4	Type 1
M_4	Type 4	Type 3	Type 3

$$N_{j,t} \in \{0,1\}, \forall j, t \tag{9}$$

$$N_{i,k,t} = 1, \forall i, k; t = C_{\max} \tag{10}$$

$$C_{\max}(\boldsymbol{\pi}^*) = \min_{\boldsymbol{\pi} \subset \prod} C_{\max}(\boldsymbol{\pi}) \tag{11}$$

$$\boldsymbol{\pi}^* = \arg\{C_{\max}(\boldsymbol{\pi})\} \rightarrow \min, \forall \boldsymbol{\pi} \subset \prod \tag{12}$$

Eqs. (1) to (5) are formulas for calculating the makespan $C_{\max}$. Eqs. (6) to (8) are formulas for calculating the ready time of the required resources for each machine. Eqs. (9) and (10) ensure that each machine can process at most one job at a time and each job should be processed once during the test process, respectively. Eqs. (11) to (12) indicate that the optimization goal is to find the best scheduling solution $\boldsymbol{\pi}^*$ among the set of all feasible solutions $\prod$ aiming to minimize the makespan $C_{\max}$.

To illustrate the problem under consideration, an example with 5 jobs, 9 operations, and 4 machines is provided here. The processing time for each operation and the setup time for each machine are given in Table 3. Each machine requires a tester, a handler, and an accessory to perform test tasks. The number and combination of available resources for machines are shown in Table 4 and Table 5, respectively. For ease of explanation, let the operation vector π_{job} and machine vector π_{mac} in $\boldsymbol{\pi}$ be $\pi_{job} = \{2,1,3,5,4,1,3,2,5\}$ and $\pi_{mac} = \{1,2,3,4,3,2,4,3,1\}$. Fig. 3($a$) shows a Gantt chart of the feasible solution $\boldsymbol{\pi} = (\pi_{job}, \pi_{mac})$. Fig. 3($b$)-3($d$) report the number of available resources corresponding to the test process.

From Fig. 3, it is clear that $O_{2,1}$ and $O_{3,1}$ are processed on M_1 and M_3 at time 0, respectively, due to the testers, handlers, and accessories are available. $O_{1,1}$ cannot be started on M_2 at time 0 because the resources required for M_2 (the first type of tester T_1) are occupied by M_1 at time 0, and the quantity for T_1 is only one. Thus, it has to be processed at time 3 after M_1 releases the occupied tester T_1. The same situation occurs for $O_{5,1}$, which must be started at time 3 after M_1 releases the occupied accessory A_3. In addition, although the required resource for M_3 is released at time 7, $O_{2,2}$ started at time 10 because of the setup time when job J_2 is transferred from M_1 to M_3 (*i.e.*, $O_{2,1}$ and $O_{2,2}$ are processed on M_1 and M_3, respectively). Thus, the makespan $C_{\max}(\boldsymbol{\pi})$ of the feasible solution $\boldsymbol{\pi}$ is 14.

4. KMEDHEA for SFTSP

This section introduces KMEDHEA, considering the critical characteristics and problem properties, which is developed to minimize the

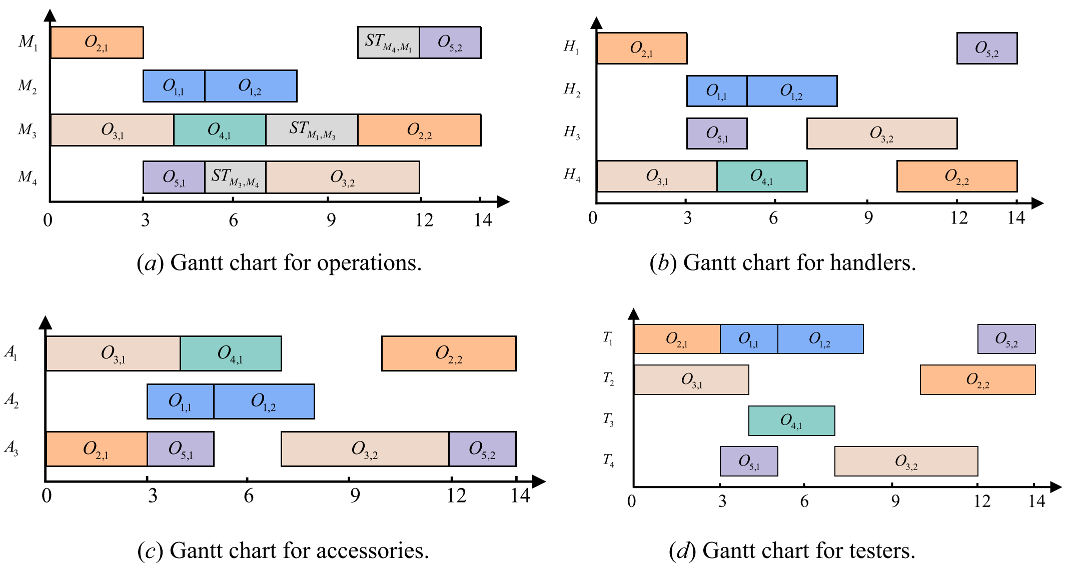


Fig. 3. Illustration of the Gantt chart for a feasible solution.

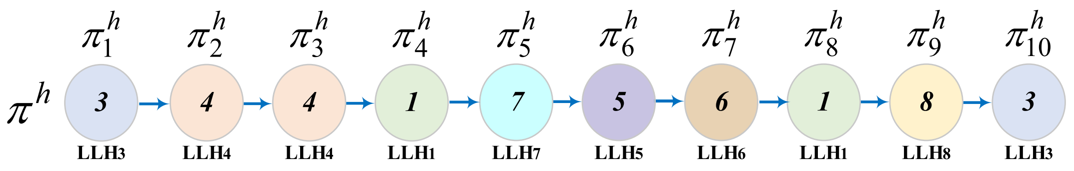


Fig. 4. Illustration of the high-level individual encoding scheme.

makespan of the SFTSP. Firstly, considering the specificity of SFTSP, single-vector encoding and constrained-separable left-shift decoding schemes are developed to represent feasible solutions and transform solutions into scheduling schemes, as described in Section 4.1. Then, a MEDA-based HLS is designed to guide the evolution of the high-level population in the strategy domain detailed in Section 4.2. Furthermore, the problem-dependent hybrid initialization method (HIM) based on three problem-specific heuristic rules is developed in Section 4.3 to produce a high-quality initial population with a certain diversity. Next, eight problem-specific heuristics are developed in Section 4.5 to generate a set of LLHs for forming high-level individuals dedicated to exploring the solution space and executing the fine-grained searches. This innovation ensures effective search behaviors, with each LLH efficiently exploring superior solutions within the specific search scope. Finally, the framework of KMEDHEA and the CC of KMEDHEA are provided and analyzed in Sections 4.6 and 4.7, respectively.

4.1. Encoding and decoding schemes

The efficacy of HIOAs in tackling complex FJSPs with multi-resource constraints depends on the design of the encoding scheme and decoding scheme (Cao, et al., 2019; Lin, et al., 2022; Sang, et al., 2018; Wang, et al., 2014, 2015; Wang, et al., 2013; Wu, et al., 2012; Zheng, et al., 2014). An effective encoding scheme properly reflects the critical characteristics of the problem, enabling efficient exploration of the solution space through specific search strategies. A suitable decoding scheme ensures that the scheduling solutions that are obtained are feasible and high-quality. KMEDHEA presents a promising perspective for HHEAs design, which employs a learning-based paradigm with a bi-level framework, where high-level individuals consist of problem-specific LLHs in the strategy space and low-level individuals consist of sequences of operations in the solution space. Detailed descriptions of the encoding and decoding schemes for the low-level and high-level individuals can be found in Subsections 4.1.1 and 4.1.2, respectively.

4.1.1. Encoding and decoding in problem domain

Encoding and decoding play a crucial role in problem-solving as they enable effective representation and transformation of the specific characteristics of the problem, allowing algorithms to better explore and exploit superior solutions. As shown in previous studies (Cao, et al., 2019; Hu, et al., 2023; Lin, et al., 2022; Sang, et al., 2018; Wang, et al., 2014, 2015; Zheng, et al., 2014), in the solution space of the problem, feasible solutions or low-level individuals are denoted as scheduling sequences, *i.e.*, $\pi_{job} = \{\pi_{job}(1), \pi_{job}(2), ..., \pi_{job}(TO)\}$. The elements in π_{job} denote the job number. The number of occurrences of the job number corresponds to the corresponding operation for that job. The total number of elements in π_{job} is TO, *i.e.*, $TO = \sum_{i=1}^{n} k_i$, and the k_i operations of job J_i are processed only once. It is this simple yet effective coding scheme that allows the execution of search operators on solution sequences to have a direct impact on scheduling schedules, thus avoiding infeasible or redundant scheduling schemes.

Decoding solution sequences into scheduling schedules requires careful consideration of multiple factors. These factors include determining the start and end times of each operation, assigning appropriate machines to perform each operation, and considering the constraints of test resources and operation priorities. However, a challenge arises due to the limited number of resources available. In some cases, most machines may have completed their processing tasks, while a few still have unfinished tasks. This imbalance may significantly increase the production cycle (*i.e.*, the total time required to complete all operations). In addition, due to priority and setup time constraints between operations, idle time may exist between two adjacent operations on the same machine, which further results in large processing time gaps. To address these challenges and minimize idle time, we adopt a constrained-separable left-shift decoding scheme. It is intended to make the processing process of operations as compact as possible. With the strategic shifting of operations within given resources and constraints, this scheme ensures that operations are scheduled to reduce idle time and minimize completion time. The details of this constrained-separable left-shift decoding scheme are outlined below. Firstly, at the start of the decoding decision, it is assumed that the operation $O_{i,k}$ is assigned to the machine $M_{i,k}$ and does not consume resources during processing. Secondly, for each operation $O_{i,k}$, all eligible machines are traversed to determine if there exists unnecessary waiting time (UWT) on the corresponding machine. The calculation of UWT in different cases is given by the following rules (*i.e.*, *UWT Rules* 1–3), respectively. Then, the most suitable machine $M_{i,k}$ and processing priority on $M_{i,k}$ are selected for each operation $O_{i,k}$ with the goal of minimizing UWT per machine, thus obtaining the machine assignment sequence π_{mac} that matches the operation sequence π_{job}. Finally, considering the resource constraints, the makespan is calculated by Eqs. (1)-(12).

UWT Rule 1: If no operations are currently scheduled for processing on the machine, then the is 0.

UWT Rule 2: If at least one operation is scheduled on the current machine and the arrived operation is the first operation $O_{i,1}$ of job J_i, the UWT is determined as follows. Define the period from the completion time of the pth operation to the start time of the $(p+1)$th operation on the same machine as the idle time gap (ITP), where $p = \{1, 2, ..., NP_{M_{i,k}} - 1\}$ and $NP_{M_{i,k}}$ is the number of jobs processed on machine $M_{i,k}$. When $p = 1$, ITP is the time from time 0 to the start of processing for the first operation on the machine. If $ITP > PT_{O_{i,1},M_{i,1}}$, then $UWT = ITP$.

UWT Rule 3: Assume that at least one operation is scheduled for processing on the machine and that the arrived operation $O_{i,k}(k > 1)$ is not the first operation of the job J_i. If $ITP \geqslant PT_{O_{i,k},M_{i,k}} + C_{i,k-1} + ST_{M_{i,k-1},M_{i,k}}$, then $UWT = ITP$.

The pseudo-code for the constrained-separable left-shift decoding scheme is shown in Algorithm 1. For ease of interpretation, the relevant notations are defined. $\varphi_{pos} = \{pos_1, pos_2, .., pos_m\}$: sequence recording the machine processing positions, *e.g.*, $pos_7 = 3$ denotes the 3rd processing position of the machine M_7. $\varphi_c = \{c_1, c_2, ..., c_n\}$: sequence recording the completion time. $\mathbf{\Lambda}_{OP}$: a 2-D matrix used to record the operations at each position on each machine, *e.g.*, $\mathbf{\Lambda}_{OP}[7, 3] = O_{3,2}$ means that $O_{3,2}$ is processed at the third position on the machine M_7.

```
Algorithm 1: Constrained-separable left-shift decoding scheme
Input: $\pi_{job}$
Output: $\pi_{mac}, C_{\max}(\boldsymbol{\pi})$
1:  Initialization: $i,q,l=1, \boldsymbol{\Lambda}_{OP} \leftarrow \varnothing, \boldsymbol{\varphi}_{pos} \leftarrow \varnothing, \pi'_{job} \leftarrow \varnothing, \pi_{mac} \leftarrow \varnothing, \boldsymbol{\varphi}_c \leftarrow \varnothing$.
2:  repeat
3:    Set $Insert\_flag = false$, $j=1$.
4:    repeat
5:      if at least $UWT$ exists on $M_j$ then
6:        Set $Insert\_flag = true$, record $UWT$, $j=j+1$.
7:      else
8:        Record $F_{M_j}$, $j=j+1$.
9:    until $j > N_{EM_{i,k}}$
10:   if $Insert\_flag = true$ then
11:     Select $M_j$ with the minimum $UWT$ as $\pi_{mac}(i)$. Save $O_{i,k}$ in $\boldsymbol{\Lambda}_{OP}$.
12:   else
13:     Select $M_j$ with the minimum $F_{M_j}$ as $\pi_{mac}(i)$. Save $O_{i,k}$ in $\boldsymbol{\Lambda}_{OP}$.
14:   Set $i=i+1$. Update $\pi_{mac}$ to obtain $\boldsymbol{\pi}=(\pi_{job},\pi_{mac})$.
15: until $i > TO$
16: repeat
17:   if $pos_l = 0$ then $pos_l = pos_l + 1$.
18:   if $\Lambda_{OP}[l, pos_l] = 0$ then $l = l+1$.
19:   else
20:     if previous operation of $\Lambda_{OP}[l, pos_l]$ is not in $\pi'_{job}$ then
21:       $l = l+1$.
22:     else
23:       $\pi'_{job}(q) = \Lambda_{OP}[l, pos_l]$, $pos_l = pos_l + 1$, $l = l+1$, $q = q+1$.
24:     if $l > m$ then $l = 1$.
25: until $q > TO$
26: for $i=1$ to $n$ do
27:   for $k=1$ to $k_i$ do
28:     if $N_{T_a} > 0, N_{H_b} > 0, N_{A_c} > 0$ then
29:       $W_1^T = W_2^H = W_3^A = 0$.
30:     else
31:       if $N_{T_a} = 0$ then $W_1^T = \min\{F_{M_1^{T_a}}, F_{M_2^{T_a}}, \ldots, F_{M_m^{T_a}}\}$. //Eq. (6)
32:       if $N_{H_b} = 0$ then $W_2^H = \min\{F_{M_1^{H_b}}, F_{M_2^{H_b}}, \ldots, F_{M_m^{H_b}}\}$. //Eq. (7)
33:       if $N_{A_c} = 0$ then $W_3^A = \min\{F_{M_1^{A_c}}, F_{M_2^{A_c}}, \ldots, F_{M_m^{A_c}}\}$. //Eq. (8)
34:       $C_{i,k} = C_{i,k} + \max\{W_1^T, W_2^H, W_3^A\}$.
35:     if $k=1$ then $c_i = C_{i,k}$.
36:     else
37:       if $C_{i,k} > c_i$ then $c_i = C_{i,k}$.
38:   if $i=1$ then $C_{\max}(\boldsymbol{\pi}) = c_i$.
39:   else
40:     if $c_i > C_{\max}(\boldsymbol{\pi})$ then
41:       $C_{\max}(\boldsymbol{\pi}) = c_i$.
```

.

4.1.2. Encoding and decoding in strategy domain

In HHEAs, high-level individuals represent specific sequences of heuristics, which consist of predefined LLHs in the strategy space. Specifically, the execution order of LLHs determines a specific search strategy for global exploration in the solution space. The search behaviors of HHEAs greatly depend on the design of the HLSs, as they directly influence the generation of high-level individuals. Therefore, the design of efficient HLSs that effectively coordinate and call LLHs is of vital importance. To clarify the description, let H^{gen} represent the high-level population at the *gen*th generation, denoted as $H^{gen} = \{H_1^{gen}, H_2^{gen}, \ldots, H_{Hs}^{gen}\}$, where the size of H^{gen} is *Hs*. Let H_{best}^{gen} be the subset of the superior high-level individuals selected from H^{gen} with size *Is*, *i.e.*,

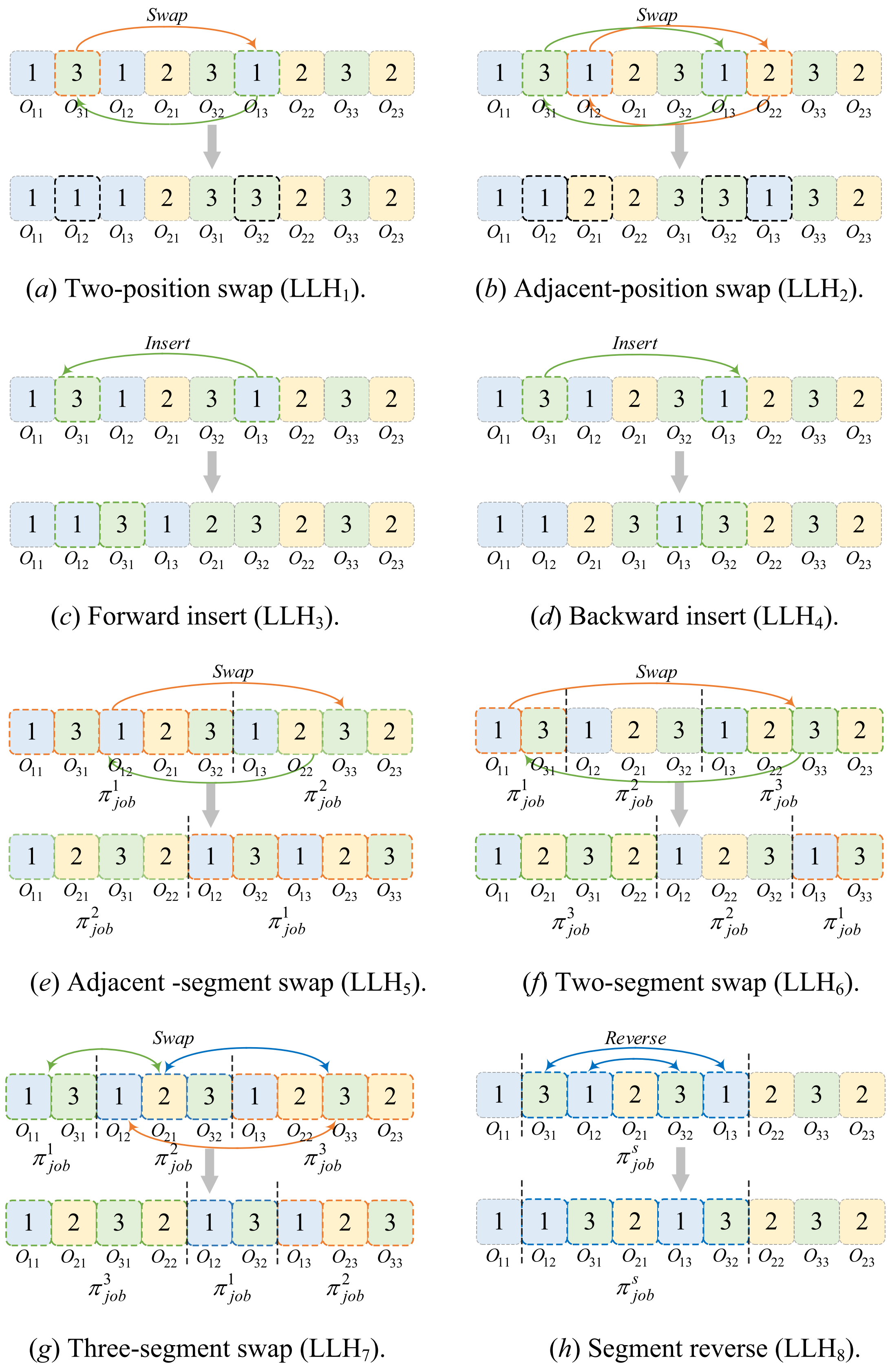

Swap
(a) Two-position swap (LLH$_1$).
Swap
(b) Adjacent-position swap (LLH$_2$).
Insert
(c) Forward insert (LLH$_3$).
Insert
(d) Backward insert (LLH$_4$).
Swap
(e) Adjacent -segment swap (LLH$_5$).
Swap
(f) Two-segment swap (LLH$_6$).
Swap
(g) Three-segment swap (LLH$_7$).
Reverse
(h) Segment reverse (LLH$_8$).

Fig. 5. Illustration of eight low-level heuristics.

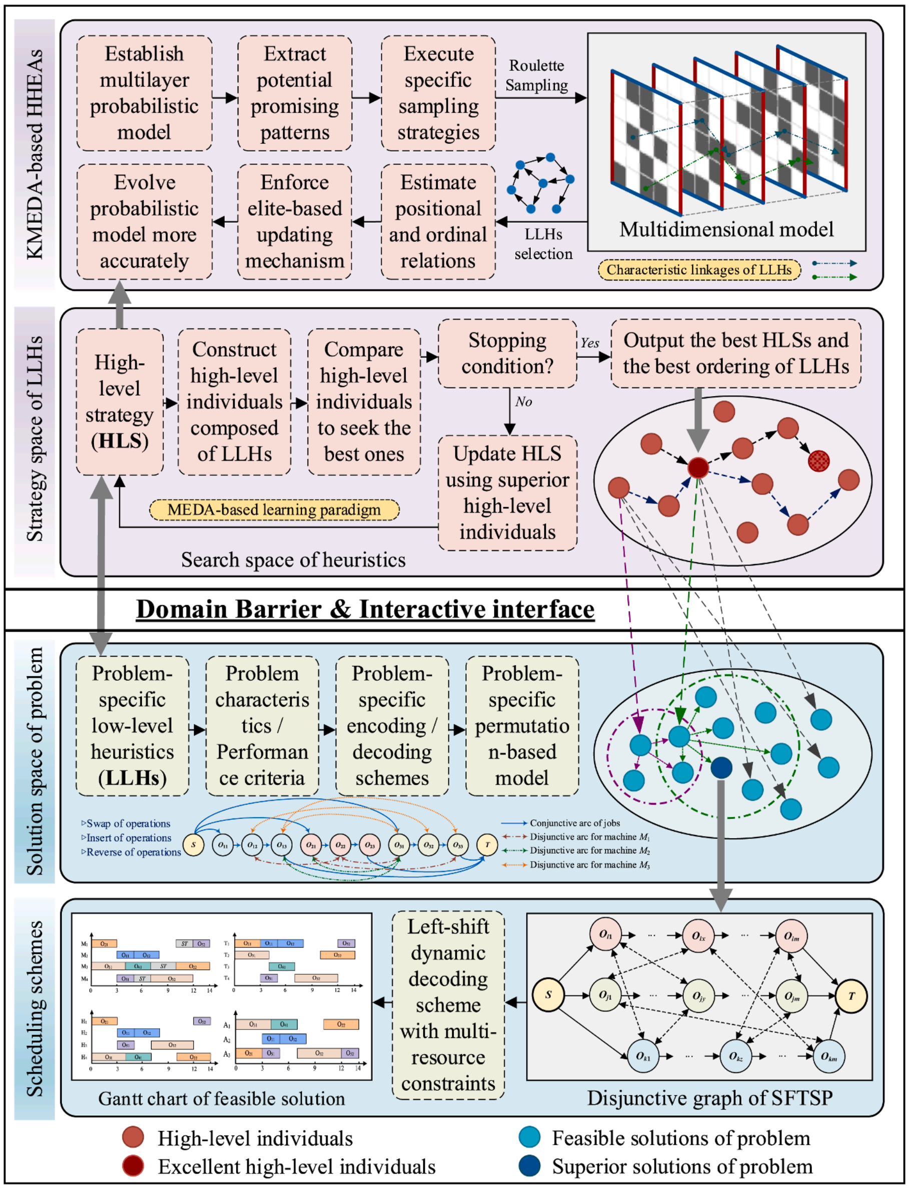

KMEDA-based HHEAs
Establish multilayer probabilistic model
Extract potential promising patterns
Execute specific sampling strategies
Roulette Sampling
Multidimensional model
Evolve probabilistic model more accurately
Enforce elite-based updating mechanism
Estimate positional and ordinal relations
LLHs selection
Characteristic linkages of LLHs
Strategy space of LLHs
High-level strategy (HLS)
Construct high-level individuals composed of LLHs
Compare high-level individuals to seek the best ones
Stopping condition?
Yes
No
Output the best HLSs and the best ordering of LLHs
Update HLS using superior high-level individuals
MEDA-based learning paradigm
Search space of heuristics
Domain Barrier & Interactive interface
Solution space of problem
Problem-specific low-level heuristics (LLHs)
Problem characteristics / Performance criteria
Problem-specific encoding / decoding schemes
Problem-specific permutation-based model
Swap of operations
Insert of operations
Reverse of operations
Conjunctive arc of jobs
Disjunctive arc for machine M1
Disjunctive arc for machine M2
Disjunctive arc for machine M3
Scheduling schemes
Gantt chart of feasible solution
Left-shift dynamic decoding scheme with multi-resource constraints
Disjunctive graph of SFTSP
High-level individuals
Excellent high-level individuals
Feasible solutions of problem
Superior solutions of problem

Fig. 6. The general framework of KMEDHEA.

$H_{best}^{gen} = \{I_1^{gen}, I_2^{gen}, ..., I_{Is}^{gen}\}$ and $Is = Hs \times \rho$, ρ is the percentage of superior high-level individuals. In addition, L^{gen} denotes the low-level population with size *pop* where L^0 is the initialization population. In this subsection, a single-vector encoding scheme is used to represent the high-level individual, denoted by $\pi^h = \{\pi_1^h, \pi_2^h, ..., \pi_{hl}^h\}$. Each element in π^h represents the index number of the LLH, and the sequence of elements denotes the execution order of the LLHs. To ensure efficiency and diversity, the length of the high-level individual is set to $hl = 10$, with no more than three occurrences of the same LLH in each individual. To facilitate understanding, an illustrative example of a high-level individual $\pi^h = \{3, 4, 4, 1, 7, 5, 6, 1, 8, 3\}$ is provided in Fig. 4.

During the decoding process of high-level individuals, the selection of LLHs is based on predetermined priorities, aimed at searching the solution space for seeking superior solutions. If a candidate solution is found to be superior to the original solution, it replaces the original one, and the remaining LLHs are executed. If the candidate solutions do not outperform any of the existing solutions, the next LLH is executed until all the remaining LLHs in the high-level individual are completed. To evaluate the effectiveness of each high-level individual, the improvement rate (IR) is utilized as an evaluating metric. The IR is calculated by determining the average fitness of the best solution obtained after executing each LLH owned by the high-level individual in the population in the solution space. To record fitness values, two sequences φ_h and φ_l are adopted, *i.e.*, $\varphi_h = \{h_1, h_2, ..., h_{pop}\}$ and $\varphi_l = \{l_1, l_2, ..., l_{pop}\}$. The pseudo-code for the calculation of IR is detailed in Algorithm 2, which outlines the steps involved in evaluating the effectiveness of each high-level individual by measuring the fitness improvement achieved by performing the respective LLHs.

.

Algorithm 2: IR calculation

```
Input $\pi^h$, $L^{gen}$
Output IR
1:  for $i=1$ to $pop$ do
2:    for $j=1$ to $hl$ do
3:      if $j=1$ then
4:        Perform $\pi_j^h$ on $L_i^{gen}$.
5:        Decode $L_i^{gen}$ to $\pi$. Calculate $C_{\max}(\pi)$. //Algorithm 1
6:        $l_i = C_{\max}(\pi)$.
7:      else if $j>1$ then
8:        Perform $\pi_j^h$ on $L_i^{gen}$.
9:        Decode $L_i^{gen}$ to $\pi$. Calculate $C_{\max}(\pi)$. //Algorithm 1
10:       if $l_i > C_{\max}(\pi)$ then
11:         $l_i = C_{\max}(\pi)$.
12:   $h_i = l_i - l_{i-1}$, $IR = IR + h_i$.
```

4.2. MEDA

4.2.1. Block structure and similar blocks

For an HLS domain individual, we define two adjacent LLHs as a block structure. Thus, each high-level individual can be seen as a composition of various block structures that appear at different positions. Among the individuals in the high-level population, the same block structures that appear in different positions across all individuals are considered as similar blocks. To illustrate this, let's consider a high-level population with four individuals: [1,3,2,2], [3,2,1,2], [1,3,1,2], and [2,3,3,1]. Here, [1,3], [3,2], and [2,2] are block structures. Notably, [1,3] appears not only as the first block in the first individual but also as the first block in the third individual, and thus block [1,3] is considered a similar block. According to the above description, a three-dimensional (3-D) probabilistic model is designed to guide the update of the high-level population based on the distribution characteristics of similar blocks.

4.2.2. Model for the distribution of similar blocks

In this subsection, the multidimensional matrix model used to record the distributional characteristics of the similar blocks of all high-level individuals in H_{best}^{gen} is defined as $SM_{(p-1)\times q\times q}^{gen}$, where p is the length of the individuals in the HLS domain and q is the number of types of LLHs. This multidimensional matrix model $SM_{(p-1)\times q\times q}^{gen}$ is described as shown in Eqs. (13) to (16):

$$\lambda\_SM_{(p-1)\times q\times q}^{gen,v}(x,y,z) = \begin{cases} 1, & y = I_v^{gen}(x), z = I_v^{gen}(x+1), \\ 0, & otherwise, \end{cases} \quad x = 1,\ldots,p-1; y,z = 1, \ldots, q; v = 1,2,\ldots,Is. \tag{13}$$

$$SM_{(p-1)\times q\times q}^{gen,v}(x,y,z) = \sum_{v-1}^{Is} \lambda\_SM_{(p-1)\times q\times q}^{gen,v}(x,y,z), \quad x = 1,\ldots,p-1; y,z = 1, \ldots, q; v = 1,2,\ldots,Is. \tag{14}$$

$$SM_{(p-1)\times q\times q}^{gen}(x) = \begin{bmatrix} SM_{(p-1)\times q\times q}^{gen}(x,1,1) & \cdots & SM_{(p-1)\times q\times q}^{gen}(x,1,q) \\ \vdots & \ddots & \vdots \\ SM_{(p-1)\times q\times q}^{gen}(x,q,1) & \cdots & SM_{(p-1)\times q\times q}^{gen}(x,q,q) \end{bmatrix}_{q\times q}, x = 1,\ldots,p-1. \tag{15}$$

$$SM_{(p-1)\times q\times q}^{gen}(y,z) = \begin{bmatrix} SM_{(p-1)\times q\times q}^{gen}(1,y,z) \\ SM_{(p-1)\times q\times q}^{gen}(2,y,z) \\ \vdots \\ SM_{(p-1)\times q\times q}^{gen}(p-1,y,z) \end{bmatrix}_{1\times q}, \quad x = 1,\ldots,p-1; y,z = 1, \ldots, q. \tag{16}$$

where $\lambda\_SM_{(p-1)\times q\times q}^{gen,v}(x,y,z)$ in Eq. (13) is the indicator function, which is used to record not only the type and order of LLHs of the v th individual I_v^{gen} in H_{best}^{gen}, but also the distribution information of the similar blocks at each position of I_v^{gen}. Eq. (14) is used to count the positional information of similar blocks of all high-level individuals in H_{best}^{gen}. $SM_{(p-1)\times q\times q}^{gen}(x,y,z)$ is an element in $SM_{(p-1)\times q\times q}^{gen}$ indicating the number of times the block structure [y,z] occurs at the x th position of all the individuals in H_{best}^{gen}. Eqs. (15) and (16) describe the structural features of the 3-D model.

4.2.3. Three-dimensional (3-D) probabilistic model

In order to effectively enhance the efficiency of learning and accumulating information about types of LLHs and characteristics of distributions in $SM_{(p-1)\times q\times q}^{gen}$, a 3-D probabilistic model $PM_{(p-1)\times q\times q}^{gen}$ based on $SM_{(p-1)\times q\times q}^{gen}$ is introduced. The element $PM_{(p-1)\times q\times q}^{gen}(x,y,z)$ in $PM_{(p-1)\times q\times q}^{gen}$ denotes the definition of the probability distribution of the similar block [y,z] at the position x of all individuals in H_{best}^{gen}, which is described in Eqs. (17)-(22):

$$PM_{(p-1)\times q\times q}^{gen}(x) = \begin{bmatrix} PM_{(p-1)\times q\times q}^{gen}(x,1,1) & \cdots & PM_{(p-1)\times q\times q}^{gen}(x,1,q) \\ \vdots & \ddots & \vdots \\ PM_{(p-1)\times q\times q}^{gen}(x,q,1) & \cdots & PM_{(p-1)\times q\times q}^{gen}(x,q,q) \end{bmatrix}_{q\times q}, x = 1,\ldots,p-1. \tag{17}$$

To efficiently sample $PM_{(p-1)\times q\times q}^{gen}$ (see Section 4.4), let the sum of similar blocks [y,z] occurring at the x th position among all individuals in H_{best}^{gen} be denoted as $sSM^{gen}(x)$, which can be calculated by Eq. (18).

$$sSM^{gen}(x) = \sum_{y=1}^{q}\sum_{z=1}^{q} SM_{(p-1)\times q\times q}^{gen}(x,y,z) \tag{18}$$

Similarly, the sum of the probability values of the similar block [y,z] arising at the x th position among all individuals in H_{best}^{gen} is defined as $sPM^{gen}(x)$.

$$sPM^{gen}(x) = \sum_{y=1}^{q}\sum_{z=1}^{q} PM_{(p-1)\times q\times q}^{gen}(x,y,z) \tag{19}$$

The specific update steps of the multidimensional probability model $PM_{(p-1)\times q\times q}^{gen}$ are detailed in Eqs. (20)-(22):

Step 1: If $gen = 0$, generate the initial 3-D probabilistic model $PM_{(p-1)\times q\times q}^{0}$ according to Eq. (20).

$$PM_{(p-1)\times q\times q}^{0}(x,y,z) = \begin{cases} 1/q, x = 1, y,z = 1,\ldots,q, \\ 1/q^2, x = 2,3,\ldots,p-1; y,z = 1,\ldots,q. \end{cases} \tag{20}$$

Step 2: If $gen = 1$, obtain the multidimensional matrix model $SM_{(p-1)\times q\times q}^{0}$ according to Eqs. (13)-(16) and update the 3-D probabilistic model $PM_{(p-1)\times q\times q}^{0}$ to $PM_{(p-1)\times q\times q}^{1}$ according to Eq. (21).

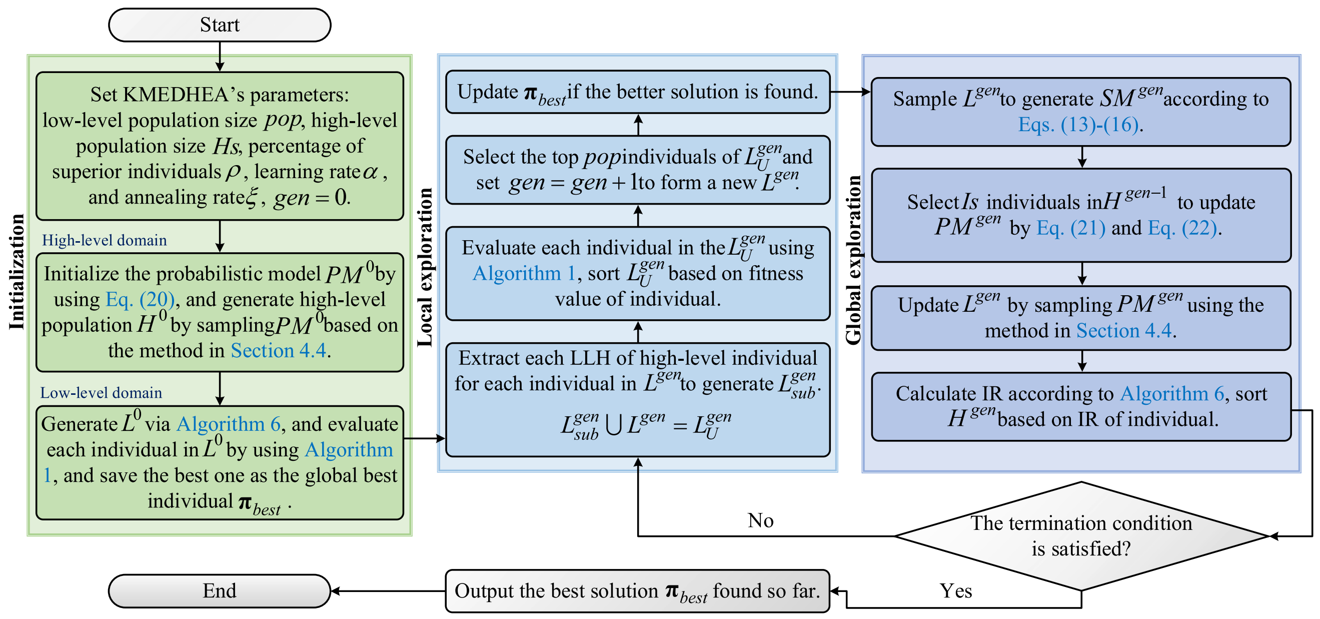

Start
Initialization
Set KMEDHEA's parameters: low-level population size pop, high-level population size Hs, percentage of superior individuals ρ, learning rate α, and annealing rate ξ, gen = 0.
High-level domain
Initialize the probabilistic model PM⁰ by using Eq. (20), and generate high-level population H⁰ by sampling PM⁰ based on the method in Section 4.4.
Low-level domain
Generate L⁰ via Algorithm 6, and evaluate each individual in L⁰ by using Algorithm 1, and save the best one as the global best individual π_best.
Local exploration
Update π_best if the better solution is found.
Select the top pop individuals of L_U^gen and set gen = gen + 1 to form a new L^gen.
Evaluate each individual in the L_U^gen using Algorithm 1, sort L_U^gen based on fitness value of individual.
Extract each LLH of high-level individual for each individual in L^gen to generate L_sub^gen.
L_sub^gen ∪ L^gen = L_U^gen
Global exploration
Sample L^gen to generate SM^gen according to Eqs. (13)-(16).
Select Is individuals in H^gen−1 to update PM^gen by Eq. (21) and Eq. (22).
Update L^gen by sampling PM^gen using the method in Section 4.4.
Calculate IR according to Algorithm 6, sort H^gen based on IR of individual.
The termination condition is satisfied?
No
Yes
Output the best solution π_best found so far.
End

Fig. 7. The flowchart for KMEDHEA.

Table 6
Computational complexity of critical components in KMEDHEA.

Crucial components	CC	Analyzes
Initialization	$O(pop \times TO)$	① The CC for generating the population by MPT rule: $O(0.4 \times pop \times TO)$ ② The CC for generating the population by APT rule: $O(0.3 \times pop \times TO)$ ③ The CC for generating the population by SSR rule: $O(0.2 \times pop \times TO)$ ④ The CC for generating the population by random generation: $O(0.1 \times pop \times TO)$
Local exploration	$O(TO) + O(poplogpop)$	① The CC for fitness value calculation: $O(TO)$ ② The CC for sorting L^{gen}: $O(poplogpop)$ ③ The CC for LLH$_1$ ~ LLH$_8$: $O(1)$
Global exploration	$O(hl^3 \times Is) + O(hl^2 \times Hs)$ $+O(0.1 \times pop \times TO)$ $+O(HslogHs)$	① The CC for updating the 3-D probabilistic model: $O(hl^3 \times Is)$ ② The CC for generating H^{gen}: $O(hl^2 \times Hs)$ ③ The CC for IR calculation: $O(hl \times pop \times TO)$ ④ The CC for sorting H^{gen}: $O(HslogHs)$

$$PM^1_{(p-1)\times q\times q}(x,y,z) = \begin{cases} SM^0_{(p-1)\times q\times q}(x,y,z)/T\_SM^0(x),\ x=1; y,z=1,\ldots,q, \\ [PM^0_{(p-1)\times q\times q}(x,y,z) + SM^0_{(p-1)\times q\times q}(x,y,z)] \\ \quad /[T\_PM^0(x) + T\_SM^0(x)], \\ \quad x=2,3,\ldots,p-1; y,z=1,\ldots,q. \end{cases} \tag{21}$$

Step 3: If $gen > 1$, calculate the multidimensional matrix model $SM^{gen-1}_{(p-1)\times q\times q}$ according to Eqs. (13)-(16) and then update the 3-D probabilistic model $PM^{gen}_{(p-1)\times q\times q}$ according to Eq. (22).

$$\begin{aligned} &PM^{gen}_{(p-1)\times q\times q}(x,y,z) = \alpha \times SM^{gen-1}_{(p-1)\times q\times q}(x,y,z) \\ &+(1-\alpha) \times PM^{gen-1}_{(p-1)\times q\times q}(x,y,z)/T\_SM^{gen-1}(x), \\ &\quad x = 1,2,\ldots,p-1; y,z = 1,\ldots,q. \end{aligned} \tag{22}$$

where α in Eq. (22) is the learning factor, which is used to control the learning rate when updating the 3-D probabilistic model.

Step 4: Set $gen = gen + 1$; if $gen < MaxGen$, skip to step 3; otherwise, terminate the loop.

4.3. Population initialization

The population initialization plays a pivotal role in directly determining the initial search scope, holding crucial significance in accelerating convergence, averting entrapment in local optima, and bolstering the stability and adaptability of HIOAs. As stated in previous studies, constructive heuristics, especially effective for tackling scheduling problems with complex constraints, are commonly employed to produce adequate and acceptable initial solutions or provide affordable available solutions for real-time requirements. As described in Section 3, the study of SFTSP involves addressing two decision-making aspects: determining the order of operations and allocating these operations to machines. In light of the insights in Subsection 4.1.1, it is clear that the order of assigning operations to machines is inherently linked to the order of operations. Thus, it becomes essential to establish heuristic rules that determine the order of operations to provide a proper pre-scheduling scheme for the initialization of HIOAs. Moreover, previous research has established that the choice of initialization strategies significantly impacts the effectiveness and efficiency of HIOAs. Random generation of initial populations, lacking consideration for problem characteristics, may compromise the quality and diversity of initial solutions. To enhance the quality of the initial population, a problem-dependent

Table 7
Parameter values for each factor level.

Parameter	Factor level			
	1	2	3	4
pop	20	40	60	80
Hs	20	30	40	50
ρ	0.3	0.35	0.4	0.5
α	0.05	0.1	0.2	0.3
ξ	0.7	0.75	0.8	0.85

hybrid initialization method (HIM) is introduced. Inspired by these insights, three heuristic rules are devised, considering the optimization objective and the problem-specific characteristics, as follows:

(1) ***MPT Rule***: Aimed at achieving a compact processing process, all jobs in π_{job} are sorted in ascending order based on the minimum processing time (MPT), and then selecting the operation with the shortest MPT. The purpose of this rule is to reduce processing time as much as possible. The pseudo-code of the MPT rule is outlined in Algorithm 3.

(2) ***APT Rule***: Aligned with the goal of the MPT rule, this rule arranges all jobs in ascending order of average processing time (APT). It selects the job with the shortest APT. The pseudo-code of the APT rule is detailed in Algorithm 4.

(3) ***SSR Rule***: Appropriate allocation of resources during scheduling greatly affects the completion time. To reasonably balance the resource allocation of machines, we introduce a separate similar resources (SSR) rule, aiming to shorten the completion time by strategically segregating operations that are processable on machines with similar resource combinations while adhering to the constraints. The pseudo-code of the SSR rule is shown in Algorithm 5.

To ensure the quality of the initial population, 90 % of the individuals in the initial population are generated by the heuristic rules described above. Specifically, 40 % of the individuals are generated by the MPT rule, 30 % by the APT rule, and 20 % by the SSR rule. The remaining 10 % of individuals are randomly generated to increase the diversity of the initial population. To provide clarity in discussing the proposed heuristic rules, relevant notations are defined. $\boldsymbol{\varphi}_{MO} = \{MO_1, MO_2, ..., MO_{TO}\}$: Sequence for recording MPT values. $\boldsymbol{\varphi}_{A} = \{AO_1, AO_2, ..., AO_{TO}\}$: Sequence for recording APT values. $\boldsymbol{\varphi}_{RO} = \{RO_1, RO_2, ..., RO_{TO}\}$: Sequence for recording a combination of resources. $\boldsymbol{\Lambda}_{MR}$: 2-D vector for recording MPT values. $\boldsymbol{\Lambda}_{AR}$: 2-D vector for recording APT values. The pseudo-code of the proposed HIM is given in Algorithm 6.

.

.

Algorithm 3: MPT rule

```
Output: $\pi_{job}$
1:  Initialize $\pi_{job}$, $\boldsymbol{\Lambda}_{MR} \leftarrow \varnothing$, $\boldsymbol{\varphi}_{MO} \leftarrow \varnothing$, $p = 1$.
2:  for $i = 1$ to $n$ do
3:      for $k = 1$ to $k_i$ do
4:          for $j = 1$ to $N_{EM_{i,k}}$ do
5:              $M_{i,k} = EM_{i,k}(j)$.
6:              if $j = 1$ then
7:                  $\Lambda_{MR}[i,k] = PT_{O_{i,k},M_{i,k}}$.
8:              else
9:                  if $PT_{O_{i,k},M_{i,k}} < \Lambda_{MR}[i,k]$ then
10:                     $\Lambda_{MR}[i,k] = PT_{O_{i,k},M_{i,k}}$. //determine MPT for $O_{i,k}$
11: repeat
12:     if $p = 1$ then
13:         for $i = 1$ to $n$ do
14:             if $i = 1$ then
15:                 for $k = 1$ to $k_i$ do
16:                     $\pi_{job}(p) = i$, $MO_p = \Lambda_{MR}[i,k]$.
17:             else
18:                 for $k = 1$ to $k_i$ do
19:                     if $\Lambda_{MR}[i,k] < MO_p$ then
20:                         $\pi_{job}(p) = i$, $MO_p = \Lambda_{MR}[i,k]$.
21:         Set $p = p + 1$. Remove $\Lambda_{MR}[i,k]$ from $\boldsymbol{\Lambda}_{MR}$.
22:     else
23:         for $i = 1$ to $n$ do
24:             for $k = 1$ to $k_i$ do
25:                 if $\Lambda_{MR}[i,k] \times 0.7 < MO_{p-1}$ then
26:                     $\pi_{job}(p) = i$, $MO_p = \Lambda_{MR}[i,k]$, $p = p + 1$. Remove $\Lambda_{MR}[i,k]$ from $\boldsymbol{\Lambda}_{MR}$.
27: until $p > TO$
```

```
Algorithm 4: APT rule
Output: $\pi_{job}$
1:  Initialize $\pi_{job}$, $\mathbf{\Lambda}_{AR} \leftarrow \emptyset$, $\boldsymbol{\varphi}_{AO} \leftarrow \emptyset$, $p = 1$.
2:  for $i = 1$ to $n$ do
3:      for $k = 1$ to $k_i$ do
4:          for $j = 1$ to $N_{EM_{i,k}}$ do
5:              $M_{i,k} = EM_{i,k}(j)$, $\Lambda_{AR}[i,k] = \Lambda_{AR}[i,k] + PT_{O_{i,k},M_{i,k}}$.
6:          $\Lambda_{AR}[i,k] = \Lambda_{AR}[i,k] / N_{EM_{i,k}}$.
7:  repeat
8:      if $p = 1$ then
9:          for $i = 1$ to $n$ do
10:             if $i = 1$ then
11:                 for $k = 1$ to $k_i$ do
12:                     $\pi_{job}(p) = i$, $AO_p = \Lambda_{AR}[i,k]$.
13:             else
14:                 for $k = 1$ to $k_i$ do
15:                     if $\Lambda_{AR}[i,k] < AO_p$ then
16:                         $\pi_{job}(p) = i$, $AO_p = \Lambda_{AR}[i,k]$.
17:         Set $p = p + 1$. Remove $\Lambda_{AR}[i,k]$ from $\mathbf{\Lambda}_{AR}$.
18:     else
19:         for $i = 1$ to $n$ do
20:             for $k = 1$ to $k_i$ do
21:                 if $\Lambda_{AR}[i,k] \times 0.7 < AO_{p-1}$ then
22:                     $\pi_{job}(p) = i$, $AO_p = \Lambda_{AR}[i,k]$, $p = p + 1$. Remove $\Lambda_{AR}[i,k]$ from $\mathbf{\Lambda}_{AR}$.
23: until $p > TO$
```

Table 8
ANOVA results of KMEDHEA's parameters.

Source	*Sum of Squares*	*Df*	*Mean Square*	*F-Ratio*	*P-Value*
pop	27601.4	3	9200.46	2436.32	**0.0000**
Hs	5372.1	3	1790.7	474.19	**0.0000**
α	4379.56	3	1459.85	386.58	**0.0000**
ρ	7500.17	3	2500.06	662.03	**0.0000**
ξ	1036.34	3	345.447	91.48	**0.0000**
$pop * Hs$	3737.85	9	415.317	109.98	**0.0000**
$pop * \alpha$	287.645	9	31.9605	8.46	**0.0000**
$pop * \rho$	595.698	9	66.1886	17.53	**0.0000**
$pop * \xi$	349.686	9	38.854	10.29	**0.0000**
$Hs * \alpha$	55.8442	9	6.20491	1.64	0.0987
$Hs * \rho$	115.295	9	12.8106	3.39	**0.0004**
$Hs * \xi$	106.407	9	11.823	3.13	**0.0010**
$\alpha * \rho$	1100.55	9	122.283	32.38	**0.0000**
$\alpha * \xi$	614.594	9	68.2882	18.08	**0.0000**
$\rho * \xi$	96.9888	9	10.7765	2.85	**0.0025**
Residual	3466.71	918	3.77637		
Total	56416.8	1023			

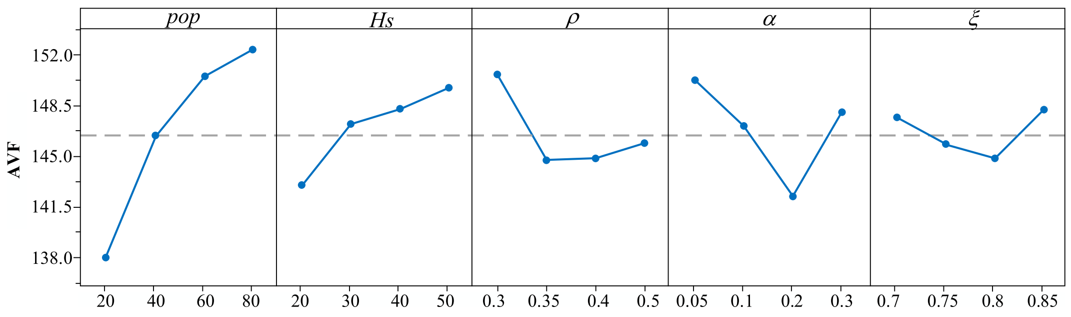


Fig. 8. Main effect plots of parameters.

Algorithm 5: SSR rule

```
Output  $\pi_{job}$
1:  Initialize $\pi_{job}, \varphi_{RO} \leftarrow \varnothing, p = 1$.
2:  repeat
3:      if $p = 1$ then
4:          Randomly choose $O_{i,k}$.
5:          $\pi_{job}(p) = i, RO_p = R_{M_{i,k}}, p = p + 1$.
6:      else
7:          Randomly choose $O_{i,k}$.
8:          for $j = 1$ to $N_{EM_{i,k}}$ do
9:              $M_{i,k} = EM_{i,k}(j)$.
10:             If $R_{M_{i,k}} \neq RO_{p-1}$ then
11:                 $\pi_{job}(p) = i, RO_p = R_{M_{i,k}}, p = p + 1$.
12: until $p > TO$
```

Algorithm 6: HIM

```
Output: initial population $L^0$.
1:  for $i = 1$ to $pop$ do
2:      if $i \leq pop \times 0.4$ then
3:          $\pi_{job} \leftarrow MPT\ rule$.   //Algorithm 3
4:      else if $pop \times 0.4 < i \leq pop \times 0.7$ then
5:          $\pi_{job} \leftarrow APT\ rule$.   //Algorithm 4
6:      else if $pop \times 0.7 < i \leq pop \times 0.9$ then
7:          $\pi_{job} \leftarrow SSR\ rule$.   //Algorithm 5
8:      else
9:          $\pi_{job} \leftarrow Random\ inilization$.
10:     Save $\pi_{job}$ into $L^0$.
```

4.4. Sampling 3-D probabilistic model to generate high-level populations

Since the probability value of a similar block $[I_v^{gen}(u-1), I_v^{gen}(u)]$ is stored in $PM_{(p-1)\times q\times q}^{gen}(u-1)$, we obtain $I_v^{gen}(u)$ at the uth position of I_v^{gen} by sampling $PM_{(p-1)\times q\times q}^{gen}(u-1)$. However, by the definition of the similar block, when $u = 1$, there is no similar block in the first position of I_v^{gen}. Therefore, a special sampling strategy is designed for the selection of the first LLH in the high-level individual, and we sample $PM_{(p-1)\times q\times q}^{gen}(u-1)$ to select the LLH for other positions. The update process of the high-level population H^{gen} is as follows:

Step 1: Set $v = 1$.

Step 2: Set $u = 1$. Calculate $RW^{gen-1}(y)$ according to Eq. (23).

$$RW^{gen-1}(y) = \sum_{z=1}^{p} sPM^{gen-1}(1, y, z), y = 1, ..., p \tag{23}$$

Step 3: Select the LLH of $I_v^{gen}(u)$ by roulette wheel, randomly generate the probability value r_1, $r_1 \in \left[0, \sum_{j=1}^{p} RW^{gen-1}(j)\right]$; if $r_1 \in \left[0, RW^{gen-1}(1)\right]$, then set $x = 1$, and set $I_v^{gen}(1)$ as LLH$_x$. If $r_1 \in \left[\begin{array}{l}\sum_{j=1}^{\omega} sPM_{(p-1)\times q\times q}^{gen-1}(i-1, I_v^{gen}(u-1), j), \\ \sum_{j=1}^{\omega+1} sPM_{(p-1)\times q\times q}^{gen-1}(i-1, I_v^{gen}(u-1), j)\end{array}\right]$, $\omega \in \{1, ..., q-1\}$, then set $x = \omega + 1$, and set $I_v^{gen}(1)$ as LLH$_x$.

Step 4: Set $u = u + 1$.

Step 5: Select LLH$_x$ of $I_v^{gen}(u)$ by the roulette wheel, and randomly generate the probability value r, $r \in \left[0, \sum_{j=1}^{q} sPM_{(p-1)\times q\times q}^{gen-1}(u-1, I_v^{gen}(u-1), j)\right]$.
If $r \in \left[0, sPM_{(p-1)\times q\times q}^{gen-1}(u-1, I_v^{gen}(u-1), j)\right]$, then set $x = 1$, and set $I_v^{gen}(u)$ as LLH$_x$. If $r_1 \in \left[\begin{array}{l}\sum_{j=1}^{\omega} sPM_{(p-1)\times q\times q}^{gen-1}(u-1, I_v^{gen}(u-1), j), \\ \sum_{j=1}^{\omega+1} sPM_{(p-1)\times q\times q}^{gen-1}(u-1, I_v^{gen}(u-1), j)\end{array}\right]$, $\omega \in \{1, ..., q-1\}$, then set $x = \omega + 1$, and let $I_v^{gen}(u)$ be LLH$_x$.

Step 6: Set $u = u + 1$; if $u \leqslant q$, then go to step 5; otherwise, go to step 7.

Step 7: Set $v = v + 1$; if $v \leqslant Hs$, then go to step 2; otherwise, go to step 8.

Step 8: Output H_{gen}.

4.5. Low-level heuristics

The low-level heuristics (LLHs) are the underlying foundational structures in HHEAs, which determine in detail some specific search strategies for performing local searches and optimization operations within the search space. These diverse LLHs are chosen and called through the HLS and thus compose the execution sequences of high-level individuals. Since specific LLHs are responsible for local search operations, both their quality and the way they are selected affect the algorithm's ability for local exploitation (Zhang, et al., 2023). Typically, LLHs are designed on the basis of the domain-specific knowledge, but they can also be developed with respect to the neighborhood structure of specific problems (Zhang, et al., 2023). These neighborhood structures are defined by the description of how some specific operators change the sequences of current solutions to create candidate solutions. Since different neighborhood structures have different search behaviors, the selection scheme of the suitable structure will have an important impact on both the effectiveness and efficiency of the proposed algorithm (Zhang, et al., 2021). Based on three types of easy-implement operators (*i.e.*, *Swap*, *Insert, and Reverse*), this section provides eight simple and effective heuristics for generating an LLH pool. For a more intuitive description, the illustrations of LLH$_1$-LLH$_8$ are shown in Fig. 5(*a*)-(*h*), respectively. The details of these LLHs are described as follows.

(1) ***Two-position swap*** (LLH$_1$): Randomly select two positions from the sequence π_{job}, and then swap the operations on the positions.

(2) ***Adjacent-position swap*** (LLH$_2$): Randomly select two positions p and p' from the sequence π_{job}, and then swap the operation on the position p with the operation on the position p', and swap the operation on the position $p+1$ with the operation on the position $p' + 1$.

(3) ***Forward insert*** (LLH$_4$): Randomly select two positions p and p' $(p < p')$ from the sequence π_{job}, and then insert the operation on the position p' before the operation on the position p.

(4) ***Backward insert*** (LLH$_4$): Randomly select two positions p and p' $(p < p')$ from the sequence π_{job}, and then insert the operation on the position p behind the operation on the position p'.

(5) ***Adjacent-segment swap*** (LLH$_5$): Randomly separate the sequence π_{job} into two segments (*i.e.*, π_{job}^1 *and* π_{job}^2), and then swap the operations on the subsequence π_{job}^1 with the operations on the subsequence π_{job}^2.

(6) ***Two-segment swap*** (LLH$_6$): Randomly separate the sequence π_{job} into three segments (*i.e.*, π_{job}^1, π_{job}^2, and π_{job}^3), and then swap the operations on the subsequence π_{job}^1 with the operations on the subsequence π_{job}^3.

(7) ***Three-segment exchange*** (LLH$_7$): Randomly separate the sequence π_{job} into three segments (*i.e.*, π_{job}^1, π_{job}^2, and π_{job}^3), and then swap the operations on these three subsequences with each other.

(8) ***Segment reverse*** (LLH$_8$): Randomly select a subsequence π_{job}^s from the sequence π_{job}, and then reverse the operations on the subsequence

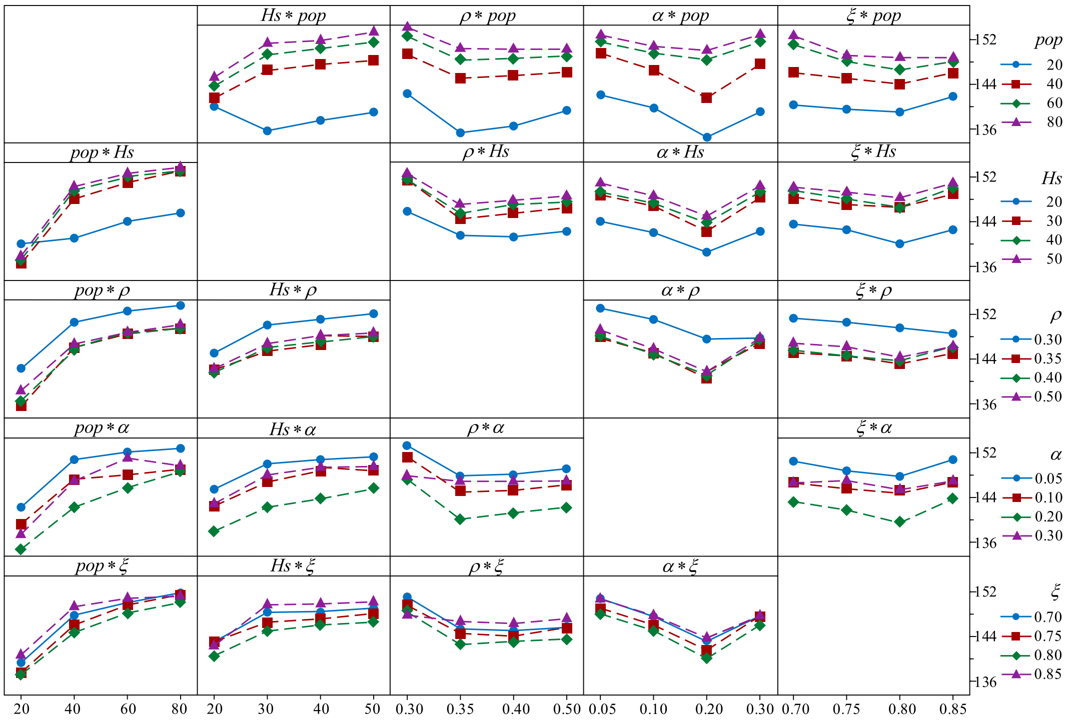


Fig. 9. Interaction effect plots of parameter pairs.

π^s_{job}.

To maintain the search vitality of LLHs to avoid premature convergence, each LLH is embedded with the SA-based acceptance mechanism to accept individuals with relatively poor fitness to improve the performance of the fine local search. These SA-embedded LLHs are executed according to Algorithm 7. The job or operation sequence that needs to be processed is denoted by ϕ. ξ is the annealing rate. The beginning temperature is denoted by T_0 (*i.e.*, $T_0 = 5$) and the terminal temperature is denoted by T_f (*i.e.*, $T_f = 1$).

.

4.6. The framework of KMEDHEA

With the critical components outlined above, this section provides the framework of the MEDA-based HHEA for solving the SFTSP, as illustrated in Fig. 6. Furthermore, the flowchart of the proposed KMEDHEA is shown in Fig. 7. Firstly, the HIM (see Algorithm 6) is employed to produce *pop* potential and promising solutions in the initial

Algorithm 7: SA-based acceptance mechanism

```
Initialization: The initial temperature $T_0$ is set to 5.
1: The LLH is applied on $\pi_{job}$ to generate $\pi'_{job}$.
2: Decode $\pi_{job}$ and $\pi'_{job}$ to $\boldsymbol{\pi}$ and $\boldsymbol{\pi}'$. Calculate $C_{\max}(\boldsymbol{\pi})$ and $C_{\max}(\boldsymbol{\pi}')$. //Algorithm 1
3: While $T_0 > T_f$
4:     The LLH is applied on $\pi'_{job}$ to generate $\pi''_{job}$.
5:     Decode $\pi''_{job}$ to $\boldsymbol{\pi}''$ and calculate $C_{\max}(\boldsymbol{\pi}'')$. //Algorithm 1
6:     If $\Delta T < 0$ then // $\Delta T = C_{\max}(\boldsymbol{\pi}') - C_{\max}(\boldsymbol{\pi}'')$
7:         $\pi'_{job} = \pi''_{job}$.
8:     else
9:         If $rand(0,1) < \exp(-\Delta T / T_0)$ then
10:            $\pi'_{job} = \pi''_{job}$.
11:    $T_0 = \xi \times T_0$.
12: If $C_{\max}(\boldsymbol{\pi}') < C_{\max}(\boldsymbol{\pi})$ then
13:     $\pi_{job} = \pi'_{job}$.
```

Table 9
The parameter settings of SA-HH, BS-HH, and GP-HH.

Algorithm	Parameters setting
SA-HH	$T_0 = 3000$, $coolingrate = 0.9$, $Num_{EVAL} = 3000$
BS-HH	$popsize = 50$, $\lambda = 5$, $r_{mix} = 1$
GP-HH	$popsize = 40$, $D = 4$, $R_m = 0.15$

population L^0 with both quality and diversity. As defined in Section 4.5, eight LLHs are chosen to construct high-level individuals based on the MEDA-based HLS, which are evaluated by a constrained-separable left-shift decoding scheme (see Algorithm 1). As illustrated in Fig. 7, the iterative process of the KMEDHEA consists of two main parts: global exploration and local exploration. In the global exploration, MEDA acts as the HLS to extract and estimate the characteristic of the block structure (*i.e.*, linkage relations of LLHs) and the distribution of similar blocks (*i.e.*, location information of LLH pairs) by the 3-D probabilistic model, which aims at identifying the most suitable sequences of heuristics. These sequences of LLHs are executed for searching the solution space with the expectation of obtaining superior solutions in the local exploration. This entire procedure is iteratively performed until the termination criteria are satisfied.

4.7. *Complexity analysis of KMEDHEA*

Computational complexity (CC) is an important indicator for evaluating the efficiency and feasibility of the algorithm. Analyzing the CC of critical components in KMEDHEA not only provides insights into how the algorithm entirely behaves under different problem sizes but also reveals bottlenecks and boosts in the algorithm, contributing to directing the development and improvement of the algorithm. As illustrated in Fig. 7, KMEDHEA can be classified into three crucial components, including population initialization, local exploration, and global exploration. To analyze the CC of each component in KMEDHEA for solving the SFTSP, Table 6 provides a rough estimation of these CC values. Each generation's total CC of KMEDHEA is denoted as TCC_{MEDHEA}, as shown in Eq. (24).

$$TCC_{KMEDHEA} = O(pop \times TO) + O(TO) + O(poplogpop) \\ +O(hl^3 \times Is) + O(hl^2 \times Hs) + O(0.1 \times pop \times TO) + O(HslogHs) \tag{24}$$

From Eq. (24), it can be seen that the maximum overall CC of KMEDHEA is $O(hl^3 \times Is)$, where hl is a small value (in this study, it is set as 10). This suggests that the KMEDHEA has a relatively low complexity.

5. Numerical results and comparisons

This section provides comprehensive comparisons and analyses of both the effectiveness and efficiency of KMEDHEA in addressing the SFTSP. Firstly, Section 5.1 introduces the basic experimental settings, including details on the test instances and the experimental environments. Subsequently, in Section 5.2, the parameters of KMEDHEA are calibrated, involving validation of the main effects of parameters and the interaction effects between parameters. Following this, the validity of each critical component of KMEDHEA is verified in Section 5.3. Finally, in Section 5.4, extensive experiments and statistical analyses are conducted to demonstrate the KMEDHEA's superiority by evaluating the performance of KMEDHEA against several state-of-the-art algorithms.

5.1. *Experimental settings*

In order to comprehensively check KMEDHEA's capabilities and confirm its effectiveness and efficiency, we employ the publicly available SFTSP benchmark suite of 10 examples introduced by Wu and Chien (2008), which are available at the website: https://dalab.ie.nthu.edu.tw/newsen_content.php?id=0. These instances can be categorized into two subsets:

Table 10
Computational results of all HHEAs and variants of KMEDHEA.

Instance	SA-HH			BS-HH			GP-HH			KMEDHEA_v1			KMEDHEA_v2			KMEDHEA_v3			KMEDHEA		
	BST	AVG	STD	BST	AVG	STD	BST	AVG	STD	BST	AVG	STD	BST	AVG	STD	BST	AVG	STD	BST	AVG	STD
LS1	104	124.60	8.19	105	116.60	5.89	102	107.3	3.73	96	100.20	3.84	95	99.00	4.04	90	95.15	3.84	**89**	**92.35**	**2.29**
LS2	125	139.65	9.55	111	117.80	5.18	106	112.55	3.90	104	107.93	3.01	104	107.60	3.94	100	103.01	3.61	**97**	**100.65**	**1.62**
LS3	112	123.60	7.32	107	111.75	3.03	102	106	2.81	101	106.85	3.31	99	102.75	3.40	94	98.46	3.20	**93**	**97.15**	**2.73**
LS4	129	141.60	8.51	122	126.55	3.57	121	125.1	2.98	116	121.00	3.83	113	117.54	3.91	**110**	113.72	3.15	**110**	**112.95**	**1.83**
LS5	122	140.15	8.44	116	121.25	**2.51**	112	116	3.19	116	126.82	7.51	106	109.45	3.37	102	105.47	3.44	**98**	**102.70**	3.76
WR1	229	255.70	13.15	231	249.50	11.34	242	257.3	7.77	236	247.25	7.46	224	234.00	7.49	**218**	**226.40**	6.88	221	227.65	**3.81**
WR2	175	198.40	9.99	192	219.35	11.59	193	201.6	**5.18**	193	204.21	5.71	183	188.91	5.64	**176**	**182.35**	5.75	177	185.80	5.24
WR3	195	215.05	11.02	216	226.10	7.46	220	224.6	3.81	208	219.54	7.53	204	211.00	5.69	196	207.43	9.89	**192**	**202.45**	**3.22**
WR4	195	203.45	7.76	187	201.65	10.58	189	189.95	0.86	**185**	191.35	6.35	189	193.10	3.77	**185**	190.45	4.07	**185**	**185.02**	**0**
WR5	203	231.00	11.97	222	228.25	5.27	214	222.45	**4.66**	205	228.63	9.50	202	209.35	6.89	**195**	203.77	7.58	**196**	**201.55**	5.81
Average	158.60	177.32	9.59	160.90	171.88	6.64	161.10	166.29	3.89	156	165.38	5.80	151.90	157.27	4.81	146.60	152.59	5.14	**146.10**	**150.82**	**3.03**

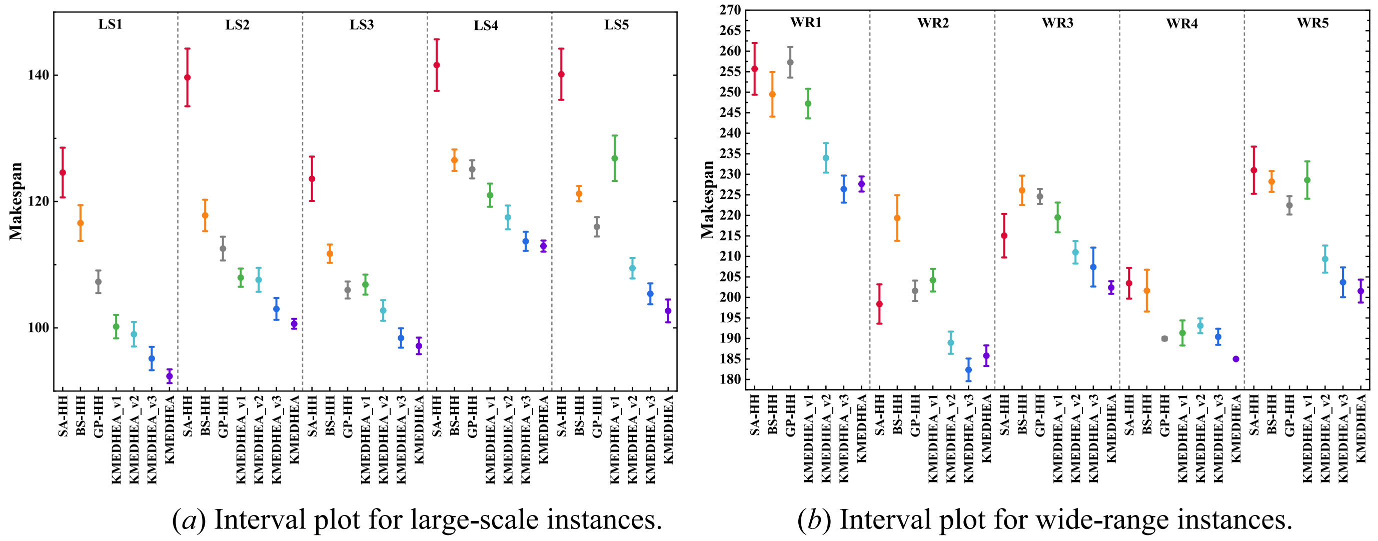


Fig. 10. Means plots and 95% Tukey's HSD confidence intervals for the interaction between the algorithms and instances with different sizes.

- **Large-Scale Instances**: This subset comprises 5 large-scale (LS) instances, each consisting of 100 jobs, 36 machines, and 196–213 operations. The processing times for each operation in these instances range from 1 to 15 h.
- **Wide-Range Instances**: The second subset contains 5 wide-range (WR) instances, each composed of 60 jobs, 36 machines, and 114–134 operations. These instances exhibit a wider range of processing times for each operation, varying from 1 to 50 h.

Considering the resource constraints inherent in SFTSP, the above-mentioned instances include three types of resources, *i.e.*, testers, handlers, and accessories. These three resources collectively contribute to machine configuration and cooperatively complete each job's operations. To ensure fair comparisons and eliminate random errors, all algorithms are executed under the same experimental setup, including identical hardware facilities, programming language, and termination conditions. To be more precise, all algorithms involved in this study have been implemented in Delphi 2010 and independently executed on a PC equipped with an Intel Core™ i7-3050 CPU@ 4 GHz and 16 GB of RAM, operating in the Windows 10 environment. The Pascal code can be found on the web: https://github.com/qxh-zhu/KMEDHEA-for-SFTSP. Each algorithm shares the same termination criteria, which is set to the maximum elapsed CPU time of $n \times m \times \lambda$ seconds, and the runtime factor λ can be adjustable to 1 and 5. Furthermore, to guarantee the stability and reliability of the test results, each algorithm is independently executed 20 times per instance. All the solutions from these runs are combined and chosen as the best solution for the specific instance. Thus, for all test instances, 200 results are generated for each algorithm, providing a fair basis for computational comparisons.

5.2. Analysis of parameters

Parameter settings have a significant influence on the performance of HIOAs, and selecting appropriate configurations is pivotal for determining the operational efficiency and optimization effectiveness of HHEAs. To determine desirable parameter combinations for the KMEDHEA, following the existing work (Zhang, et al., 2023), the Design of Experiments (DOE) method (Hinkelmann, et al., 2008) was employed in this section. There are five key parameters in KMEDHEA, *i.e.*, low-level population size (*pop*), high-level population size (*Hs*), percentage of superior high-level individuals (ρ), learning rate (α), and annealing rate (ξ). The parameter values and levels were determined by investigating previous studies on solving similar problems and by conducting preliminary experiments. Table 7 details the values of each parameter across different levels, along with all possible parameter combinations, resulting in a total of $4 \times 4 \times 4 \times 4 \times 4 = 1024$ configurations for KMEDHEA. However, it is worth noting that there would be a potential risk of overfitting by adjusting parameters using the same instances; thus, the same instances should not be adopted for parameter calibration and computational comparisons. Therefore, ten new instances were generated using the same method as the benchmark instances for parameter calibration. Each configuration was evaluated on these ten instances, and each algorithm was executed 20 times per instance, resulting in a total of $1024 \times 10 \times 20 = 204800$ treatments in the full factorial experimental design across all configurations. Herein, the termination condition for the algorithm was set to a maximum elapsed CPU time of $n \times m \times 0.5$ seconds. To put this into perspective, if the parameter calibration program were to run as a single-process program, it would require at least 2560 CPU days to complete. Fortunately, multiple PCs with identical multi-core architectures were available to conduct these experiments in parallel, reducing the total time to 35.5 days to complete the entire parameter calibration. To evaluate the effectiveness of the parameter combinations, the average fitness value (AVF) was used as the response variable (RV). Obviously, the lower RV value signifies the superior performance of the algorithm associated with the specific parameter combination. Similar to previous studies, the experimental results were analyzed by multifactor analysis of variance (ANOVA), which is a robust parametric statistical technique applied to determine whether two or more factors significantly affect the average values of samples across different levels. ANOVA is widely applied in analyzing experimental results from HIOAs used to solve various scheduling problems (Zhang, et al., 2021). Before conducting ANOVA, it is imperative to check and confirm that three primary assumptions are met: normality, homoscedasticity, and independence of residuals, to ensure the validity of the analysis. Upon careful checking with experimental results, it was found that all assumptions were satisfied, indicating no significant differences.

The ANOVA results for KMEDHEA's parameters are presented in Table 8. For ANOVA results, when the $P-Value$ is less than 0.05, the $F-Ratio$ will be used as an indicator of significance. In general, a large $F-Ratio$ suggests that the factor has a significant impact on the RV. As can be seen in Table 8, the $P-Values$ for all the parameters are below the confidence level of 0.05, which confirms the significance in the statistical sense. Among these parameters, the low-level population size (*pop*) is the most significant one, as evidenced by its largest $F-Ratio$, indicating its considerable influence on the performance of KMEDHEA.

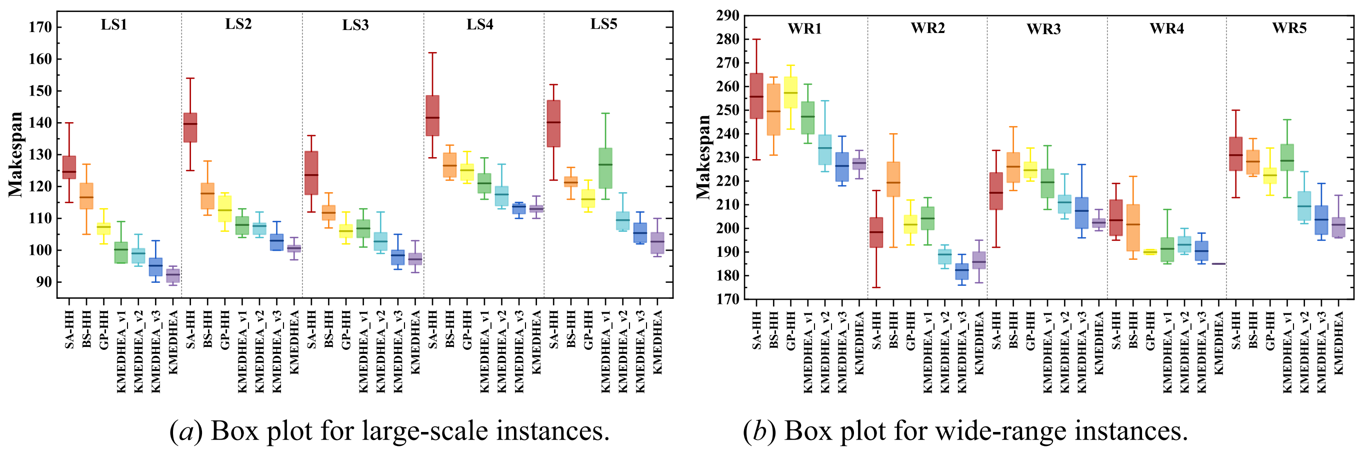


Fig. 11. Box plots and 95% Tukey's HSD confidence intervals for the interaction between the algorithms and instances with different sizes.

According to all $F-Ratio$ values, the impact of the other parameters on the algorithm follows in descending order: *pop*, ρ, *Hs*, α, and ξ. Fig. 8 illustrates the main effects plot for all parameters. It can be seen that performance deteriorates with increasing *pop*, with the choice of $pop = 20$ yielding the best results. The reason for this is that a smaller low-level population size facilitates greater potential for global exploration. If the large size of *pop* is chosen, the algorithm's ability to perform more iterations within the limited CPU time may be compromised. As a result, the population size (*pop*) should be set to a slightly smaller value, and the same reason for the high-level population size (*Hs*). The percentage of superior individuals (ρ) emerges as the second most significant factor. As shown in Fig. 8, the choice $\rho = 0.35$ achieves the best performance. It is clear that smaller ρ may lead to inadequate learning of characteristics and correlation of LLHs from high-quality individuals during iterations, resulting in potentially missing promising patterns. This mismatch in the positioning of LLHs among offspring individuals may result in generating unreasonable LLH sequences, thus failing to produce effective and efficient problem-specific heuristics. Conversely, the larger ρ can cause the allocation of LLHs in the newly generated individuals to be too similar to that of the parent population, thereby making the algorithm easily trapped into local optima. The third most important factor is the learning rate (α), as depicted in Fig. 8, with $\alpha = 0.2$ providing the best performance, whereas $\alpha = 0.05$ results in the worst. The learning rate significantly influences the performance of KMEDHEA by controlling the updating rate of characteristic information during iterations. It is usually used as a trade-off between the convergence speed and stability of the algorithm. A smaller learning rate indicates that KMEDHEA can use existing information from the 3-D probabilistic model, which promotes stable convergence but may slow down the process. Conversely, the larger learning rate will accelerate the acquisition of new characteristic information but risks missing global optima, thus affecting the algorithm's overall search behavior. To strike a balance, it is advisable to start with the larger learning rate in early iterations, favoring large jumps in the search space to gain more experience and slowly decreasing the learning rate during the learning process. In later iterations, a smaller learning rate allows the algorithm to utilize historical experience to perform a fine-grained search across the solution space. In addition, the annealing rate (ξ) exerts the slightest significant impact on the performance of KMEDHEA, as indicated by its smallest $F-Ratio$. It is obvious from Fig. 8 that $\xi = 0.8$ can obtain the best performance, while $\xi = 0.7$ leads to the worst results. Therefore, a slightly larger annealing rate ξ can be beneficial for enhancing the effectiveness of the fine-tuning search.

Although Fig. 9 has provided the preferred choice of the best levels for each parameter, it's important to note that analyzing single parameters (as shown in Fig. 8) may not yield meaningful results when significant interactions exist between these parameters (Zhang, et al., 2021). Therefore, we further conducted a more in-depth investigation into the two-level interactions among these five parameters. The corresponding statistical results are reported in Tables 7 and 8, respectively. As demonstrated in Tables 7–8, the $P-Values$ of almost all pairs of parameters, regarding their interactions, are below 0.05, suggesting that the interactions between parameters are not statistically significant. The interaction effect plots for these parameter pairs are depicted in Fig. 9. It's evident from these plots that the interactions between parameters are quite weak, consistent with the findings from Fig. 8. Furthermore, as indicated in Table 8, the $F-Ratio$ values for the main effects of each parameter are greater than those for the interaction effects between the parameters, implying that the interaction effects can be neglected. According to the parameter experiments and analysis, the suitable values of KMEDHEA's parameters are suggested as follows: $pop = 20$, $Hs = 20$, $\rho = 0.35$, $\alpha = 0.2$, and $\xi = 0.8$.

5.3. Effectiveness of critical components

The efficacy of HIOAs is commonly closely coupled to the correlations among critical components, and exploring the effectiveness of each component is essential not only for analyzing the performance of the algorithm but also for revealing the underlying mechanisms driving search behaviors (Zhang, et al., 2022; Zhang, et al., 2022). As detailed description in Section 4, four critical components and corresponding improvement strategies were developed to enhance the performance of the proposed KMEDHEA, including (1) Constrained-separable left-shift decoding scheme (detailed in Subsection 4.1.1): This innovative decoding strategy was introduced to dynamically and actively decode all scheduling solutions, optimizing the sequences of solutions and parsing and translating them into high-quality scheduling schedules. (2) MEDA-based high-level strategy (designed in Section 4.2): KMEDHEA employed the HLS based on MEDA, which effectively enhances the algorithm's global exploration capabilities. (3) Hybrid initialization method (HIM) (see Section 4.3 for details): KMEDHEA utilizes HIM to generate initial populations of high quality and diversity, thereby accelerating the convergence of the algorithm. (4) SA-embedded low-

Table 11
Parameters of NFOA, HEDA, KMEA, CCIWO, CSRS, and QHH.

Algorithm	Parameter setting
NFOA	$NP = 70$, $S = 1$, $P = 0.7$
HEDA	$P = 100$, $SP = 20$, $\alpha = 0.1$, $\beta = 0.3$
KMEA	$L = 6$, $\alpha = 0.2$, $MAX\_G = 10000$
CCIWO	$\psi_0 = 10$, $s^a_{\min} = 1$, $s^a_{\max} = 2$, $\zeta^a_0 = n/2$, $\zeta^a_f = 1$, $\psi^a_{\max} = 50$, $\rho = 0.9$, $s^s_{\min} = 1$, $s^s_{\max} = 2$, $\zeta^s_0 = 5$, $\zeta^s_f = 2$, $\psi^s_{\max} = 20$
CSRS	$\lambda = 1.5$, $Ti = 150$, $Np = 30$
QHH	$T_0 = 6$, $EP = 10$, $\zeta = 0.7$, $\gamma = 0.7$

Table 12
Comparison results of KMEDHEA and six state-of-the-art algorithms ($n \times m \times 1$ (seconds)).

Instance	NFOA			HEDA			KMEA			CCIWO			CSRS			QHH			KMEDHEA		
	BST	AVG	STD	BST	AVG	STD	BST	AVG	STD	BST	AVG	STD	BST	AVG	STD	BST	AVG	STD	BST	AVG	STD
LS1	118	123.10	2.28	108	113.85	2.50	100	103.95	3.58	93	96.80	4.06	100	106.05	4.51	95	100.40	3.01	**89**	**92.35**	**2.29**
LS2	122	128.40	4.89	115	117.00	1.79	104	107.95	3.47	102	105.15	3.80	116	124.80	5.78	109	112.05	2.59	**97**	**100.65**	**1.62**
LS3	109	113.25	3.11	107	108.95	1.56	99	101.55	2.62	96	100.20	3.56	107	113.95	5.78	106	108.70	2.67	**93**	**97.15**	**2.73**
LS4	133	137.90	3.39	125	129.00	2.47	118	121.05	2.85	111	114.95	3.92	123	134.85	5.11	120	122.00	2.49	**110**	**112.95**	**1.83**
LS5	129	132.75	2.70	122	125.45	**2.33**	109	111.75	2.96	104	106.95	3.31	114	124.65	5.85	110	114.55	3.61	**98**	**102.70**	3.76
WR1	267	281.20	9.58	255	260.75	4.26	232	247.25	7.54	220	229.00	7.43	234	245.30	8.79	242	257.45	9.09	**221**	**227.65**	**3.81**
WR2	218	229.00	8.56	204	214.45	**4.21**	184	191.75	5.19	179	**183.85**	5.69	186	197.15	9.57	184	191.75	5.19	**177**	185.80	5.24
WR3	232	244.55	5.23	229	234.75	3.60	210	214.95	3.73	200	206.35	5.62	215	226.20	9.52	210	214.95	3.73	**195**	**202.40**	**3.22**
WR4	195	204.95	6.12	**185**	193.10	5.48	**185**	185.10	0.44	**185**	188.10	3.59	**185**	201.40	10.91	**185**	185.45	1.56	**185**	**185.00**	**0.00**
WR5	233	239.75	4.58	229	235.95	4.31	205	212.50	**4.30**	196	204.55	7.11	210	221.75	10.38	205	212.50	**4.30**	**196**	**201.55**	5.81
Average	175.60	183.49	5.04	167.90	173.32	3.25	154.60	159.78	3.69	148.60	153.59	4.81	159.00	169.61	7.62	156.60	161.98	3.82	**146.10**	**150.82**	**3.03**

Table 13
Comparison results of KMEDHEA and six state-of-the-art algorithms ($n \times m \times 5$ (seconds)).

Instance	NFOA			HEDA			KMEA			CCIWO			CSRS			QHH			KMEDHEA		
	BST	AVG	STD	BST	AVG	STD	BST	AVG	STD	BST	AVG	STD	BST	AVG	STD	BST	AVG	STD	BST	AVG	STD
LS1	103	115.95	6.52	105	109.30	2.33	95	101.15	2.76	89	94.00	3.95	95	99.40	2.18	94	100.50	2.91	**80**	**87.10**	**1.95**
LS2	110	116.80	5.61	113	115.50	1.60	105	110.80	2.94	98	101.90	3.39	110	118.60	3.68	106	108.95	2.25	**89**	**90.80**	**0.98**
LS3	105	110.60	3.15	105	107.75	1.44	96	99.90	2.51	92	97.00	3.15	100	102.95	2.13	98	102.85	3.50	**82**	**86.20**	**1.81**
LS4	122	125.60	3.22	125	128.10	2.32	116	119.10	2.77	109	112.50	2.71	119	123.20	2.80	117	121.95	3.76	**95**	**96.85**	**1.39**
LS5	116	120.40	2.73	120	124.25	2.14	106	110.00	3.15	98	102.85	2.89	111	121.25	5.41	106	110.65	4.04	**85**	**91.50**	**2.01**
WR1	230	240.65	6.89	253	258.80	3.39	227	238.95	9.46	215	222.00	5.25	233	239.40	4.85	235	243.15	4.59	**187**	**192.45**	**4.58**
WR2	192	210.55	8.95	202	211.75	4.05	187	192.00	4.74	174	180.30	4.65	183	191.40	4.78	183	187.75	2.79	**159**	**165.10**	**2.49**
WR3	216	225.45	7.55	227	233.35	3.80	208	215.75	6.06	193	205.40	9.71	212	216.40	3.37	206	212.40	4.63	**180**	**182.10**	**2.51**
WR4	185	195.75	7.58	185	189.75	3.87	185	186.40	1.71	185	185.00	**0.00**	185	188.15	3.37	185	185.20	0.87	**185**	**185.00**	**0.00**
WR5	214	226.35	6.88	226	232.85	4.23	198	204.60	4.39	193	200.00	5.54	206	216.95	5.95	199	208.80	4.39	**178**	**184.45**	**3.77**
Average	159.30	168.81	5.91	166.10	171.14	2.92	152.30	157.87	4.05	144.60	150.10	4.12	155.40	161.77	3.85	152.90	158.22	3.37	**132**	**136.16**	**2.15**

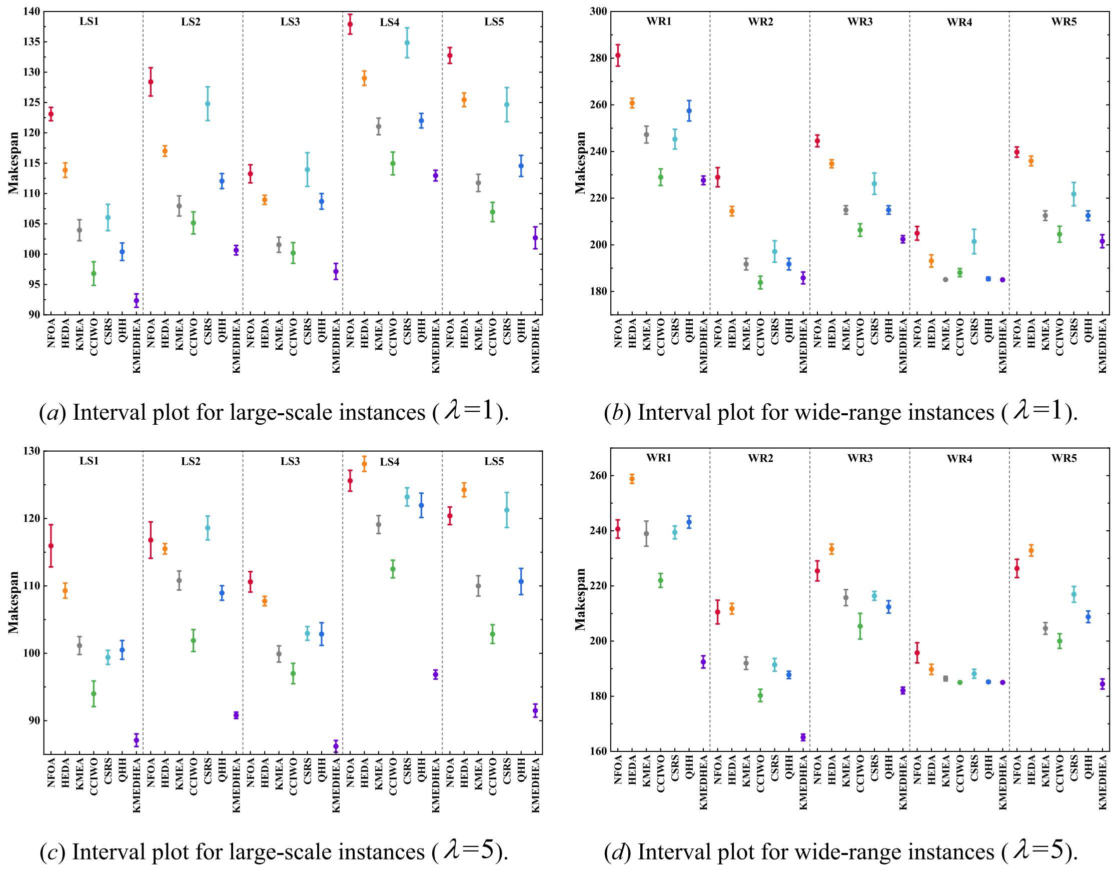


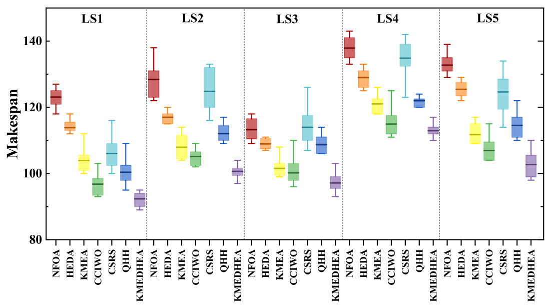
Fig. 12. Means plots and 95% Tukey's HSD confidence intervals for the interaction between the algorithms, runtime factors, and instance sizes.

level heuristics (devised in Section 4.5): LLHs embedded with a simulated annealing mechanism were developed, enriching the search behaviors and promoting more effective local exploitation. In this section, we aim to validate the robustness and effectiveness of these components and strategies by creating and analyzing variants of KMEDHEA, each focusing on one specific component or strategy. Given the critical role of the design of HLSs in HHEAs, we first validated the strengths and limitations of various HLSs. To be specific, we conducted a comprehensive comparative analysis, pitting KMEDHEA against several superior HHEAs, including SA-HH (Lim, et al., 2022), BS-HH (Lin, 2019), and GP-HH (Song, et al., 2021). To ensure fairness and the validity of computational comparisons, all algorithms, including KMEDHEA, employed the same encoding and decoding scheme, hybrid initialization method, and a consistent set of LLHs (*i.e.*, $\text{LLH}_1 \sim \text{LLH}_8$, as introduced in Section 4.5). The primary distinction among the algorithms only lies in their respective HLS designs. Moreover, the constrained-separable left-shift decoding scheme, detailed in Subsection 4.1.1, was uniformly applied across all algorithms. The parameter settings for these algorithms were sourced from their original literature, and the recommended values can be found in Table 9.

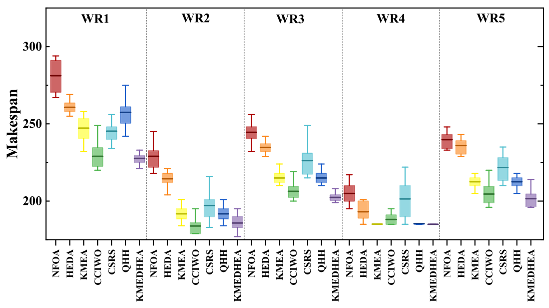
To further validate the effectiveness of the other components or strategies, three variants of KMEDHEA, denoted as KMEDHEA_v1, KMEDHEA_v2, and KMEDHEA_v3, were created, and their performance was compared with that of the original KMEDHEA. Specifically, for the first variant KMEDHEA_v1, we replaced the constrained-separable left-shift decoding scheme with the random dynamic decoding scheme, allowing us to evaluate the effect of the proposed decoding scheme. For the second variant KMEDHEA_v2, we eliminated the hybrid initialization method (HIM) and substituted HIM with a random initialization method for generating the initial population. For the simulated annealing mechanism embedded in LLHs, KMEDHEA_v3 removed the SA-based acceptance condition from each of the LLHs. It is essential to emphasize that the components within KMEDHEA are independent of each other, meaning that modifications to one component do not impact the others. For each variant, only one component was replaced, while the others remained unchanged. All variants were independently executed 20 times on the same set of 10 instances introduced in Section 5.1. The experimental setup was consistent across all tests, with the stopping condition set at a maximum CPU time of $n \times m$ seconds. The test results obtained by all algorithms for each instance are reported in Table 10, including the three metrics of best values (BST), average values (AVG), and standard deviation (STD) in detail, with the dominant values highlighted in **bold**.

As demonstrated in Table 10, for almost all instances, our presented algorithm achieves better results than other algorithms in terms of metrics, BST, AVG, and STD, underscoring the effectiveness and indispensability of all the components in KMEDHEA for addressing the SFTSP. Although these results in Table 10 reflect the strong performance of the proposed algorithm, it is still necessary to confirm whether the observed differences are statistically significant. For this purpose, we further employed the ANOVA technique to analyze these experimental results. Fig. 10 and Fig. 11 show the means plots and the box plots with 95 % Tukey's Honest Significant Difference (HSD) confidence intervals for KMEDHEA and other HHEAs, respectively, for solving instances across various problem scales. Notably, the overlapping intervals indicate statistically insignificant differences between the compared algorithms. As shown in Fig. 10, it is clear that KMEDHEA demonstrates superior stability compared to the other algorithms.

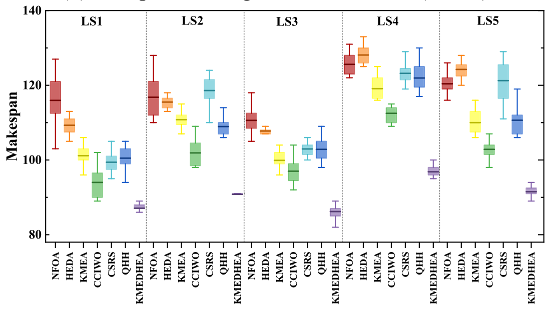
The box plots provide valuable insights into the stability of the algorithms by revealing the dispersion of test results. As depicted in Fig. 11, the interquartile range (IQR), represented by the length of the box, is notably narrower for KMEDHEA compared to all other algorithms, regardless of both small- and large-scale instances. These findings suggest that KMEDHEA yields tightly clustered test results,



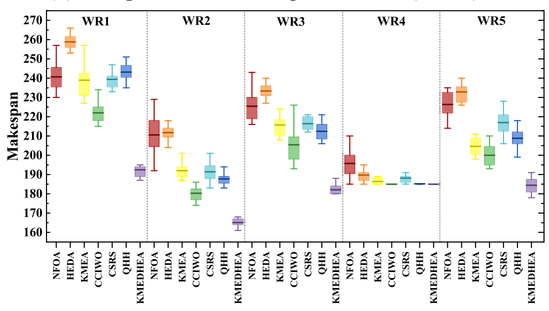
(*a*) Box plot for large-scale instances (λ=1).



(*b*) Box plot for wide-range instances (λ=1).



(*c*) Box plot for large-scale instances (λ=5).



(*d*) Box plot for wide-range instances (λ=5).

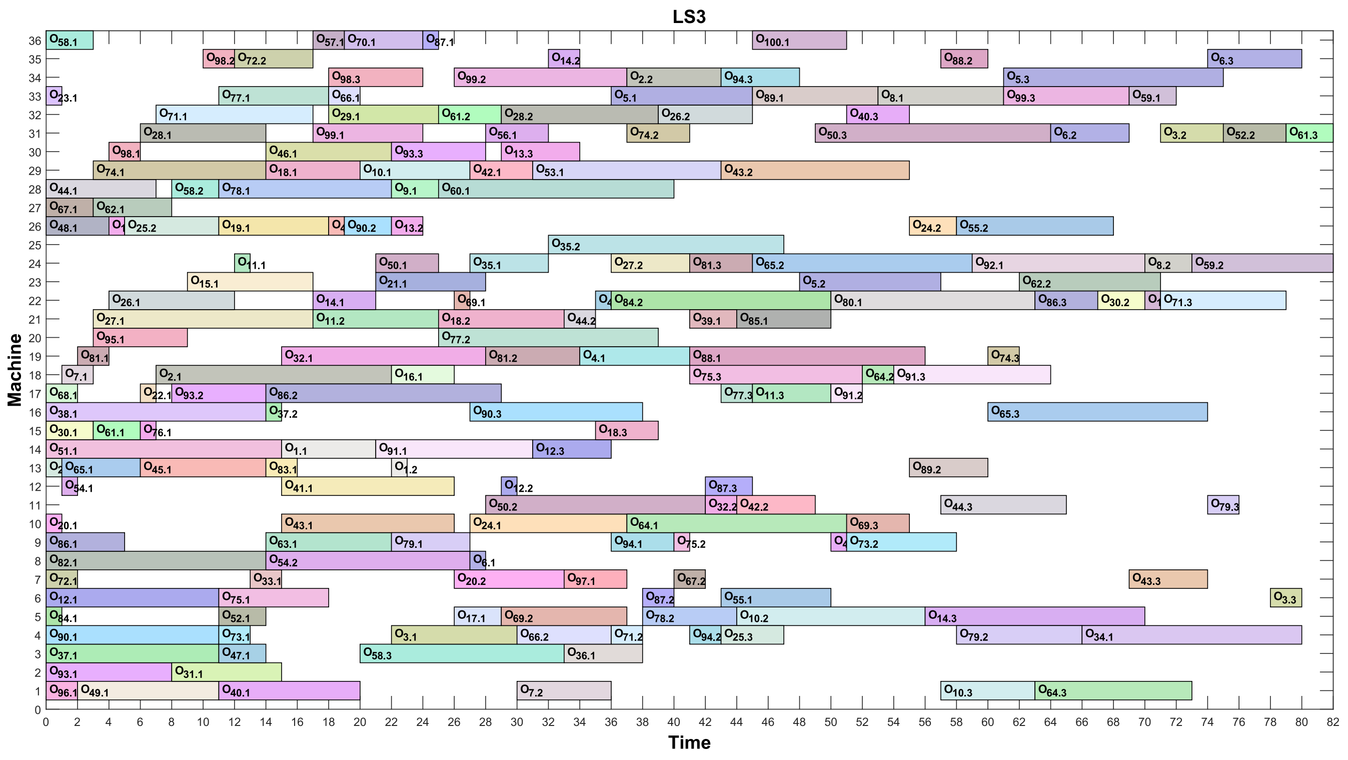
Fig. 13. Box plots and 95% Tukey's HSD confidence intervals for the interaction between the algorithms, runtime factors, and instance sizes.

suggesting its superior stability. Consequently, KMEDHEA exhibits robust search capabilities, outperforming other algorithms when tackling SFTSP across various instance sizes. Through careful analysis of Figs. 10-11 and the results presented in Table 10, it is evident that KMEDHEA demonstrates significant superiority over SA-HH, BS-HH, and GP-HH. On the basis of these observations, two conclusions can be concluded by carefully analyzing these causes. First, the performance of SA-HH is highly dependent on the quality of the initial solution, and it is susceptible to getting trapped in local optima during evolution, making it prone to premature convergence compared to KMEDHEA. Second, core operations like mutation and crossover, which are commonly used in backtracking search-based HHEA and genetic programming-based HHEA (*i.e.*, BS-HH and GP-HH), simply adjust the ordering of LLHs without incorporating effective and efficient learning mechanisms, and these operations may inadvertently destroy promising partial patterns. These limitations and impediments hinder the efficiency and effectiveness of extracting and estimating the positional and ordinal relationships of LLHs. Consequently, these traditional HHEAs may face challenges in capturing critical characteristics from high-quality individuals. In contrast, KMEDHEA's strength lies in MEDA-based HLS, which enables the exploration of high-quality regions within the search space of heuristics by leveraging valuable information stored in the 3-D probabilistic model. This 3-D model retains valuable information about promising patterns and structural features found in excellent high-level individuals, thereby enhancing KMEDHEA's ability to seek superior solutions.

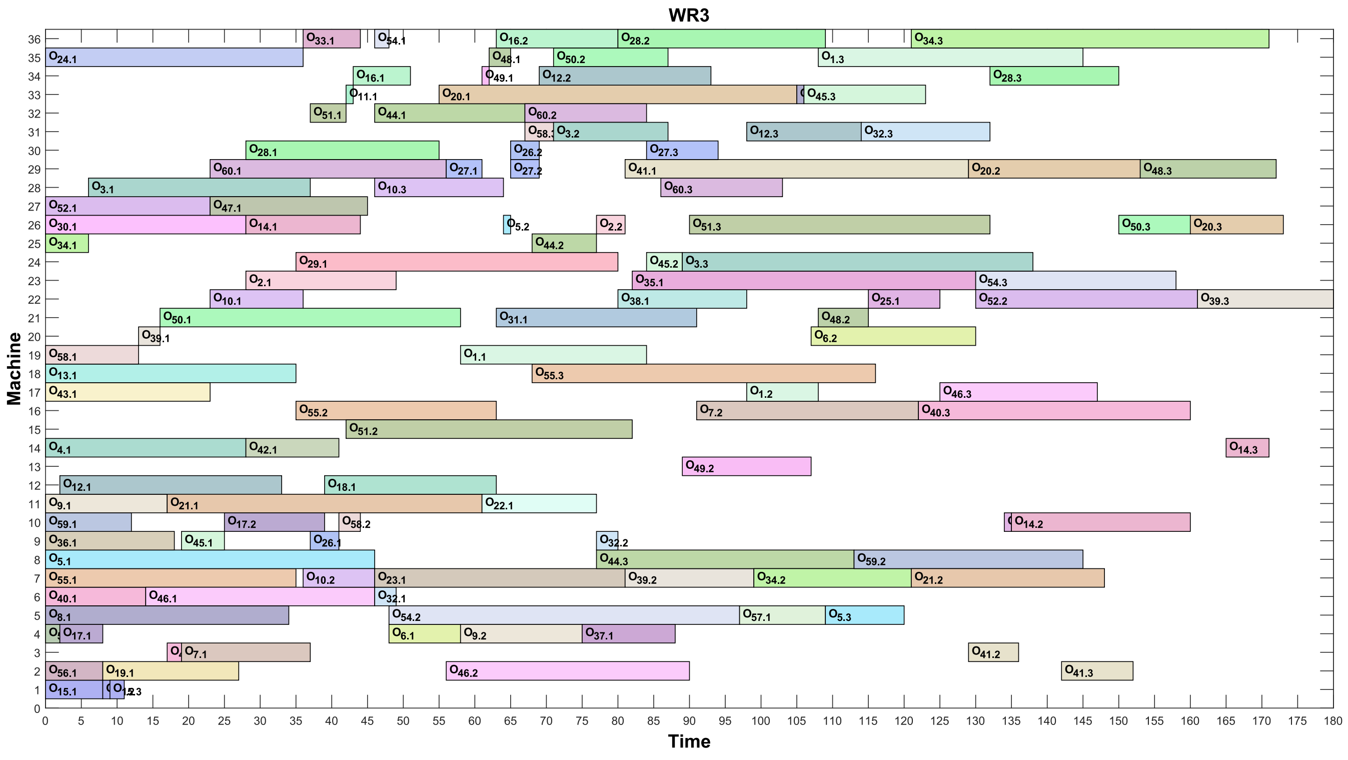
As can be expected, KMEDHEA demonstrated the effectiveness of its critical components by consistently delivering satisfactory results when compared to its three variants. The detailed statistical results reported in Table 10 confirm that KMEDHEA outperforms KMEDHEA_v1, emphasizing the advantages of the proposed problem-specific decoding scheme given in Subsection 4.1.1. An in-depth analysis of the problem's inherent characteristics reveals that both resource utilization and waiting times are crucial factors in determining the makespan of scheduling solutions. Therefore, the design of an efficient decoding strategy becomes especially essential. Such decoding strategy should be tailored to increase resource utilization while minimizing waiting time, thus paving the way for seeking superior solutions that are close to or even consistent with the optimal ones. Moreover, KMEDHEA beats KMEDHEA_v2 across all instances, which implies the pivotal role of HIM in KMEDHEA's success. Since SFTSP has considerable complexity involving additional constraints, which include setup time and multiple resources, it becomes imperative to devise suitable heuristic rules that cater to these specific characteristics. The initial solutions, crafted by these heuristic rules, facilitate the population to swiftly converge towards promising regions during the early iterations and improve the potential to seek and converge upon optimal solutions in subsequent iterations. To enrich the diversity of initial solutions and expand the search scope in the solution space, the proposed HIM combining both heuristic rules and random initialization can further improve the quality of initial solutions. Furthermore, compared to KMEDHEA_v3, the results of BST and AVG are also lower in most cases for KMEDHEA, indicating that embedding the SA-based acceptance mechanism in LLHs has a positive effect. These SA-embedded LLHs allow the acceptance of individuals with relatively poor fitness values. In contrast, when HHEAs always accept only so-called best solutions, they may steer the entire population towards dead ends, resulting in reduced diversity, loss of search vitality, and trapping into local optima.

5.4. Comparisons of KMEDHEA with state-of-the-art algorithms

In this section, we evaluate the effectiveness and efficiency of KMEDHEA by conducting comprehensive comparisons and extensive experiments with six state-of-the-art algorithms for solving SFTSP, *i.e.*, NFOA (Zheng, et al., 2014), HEDA (Wang, et al., 2013), KMEA (Wang, et al., 2015), CCIWO (Sang, et al., 2018), CSRS (Cao, et al., 2019), and QHH (Lin, et al., 2022). To ensure fairness in computational comparisons, we re-implemented these comparison algorithms using the same programming language and executed them in the same experimental environment as KMEDHEA. The parameter values for each compared algorithm were set according to their original literature, as listed in Table 11. As mentioned in Section 5.1, we adopted the maximum elapsed CPU time of $n \times m \times \lambda$ seconds as the stopping criteria for each compared algorithm, with the runtime factor of λ, allowing available values of 1 and 5. Each algorithm was tested 20 times per instance. The computational results are reported in Tables 12–13, including the values at the three metrics of BST, AVG, and STD, with the best results for each metric highlighted in **bold**.



(*a*) New best solution of instance LS3 obtained by KMEDHEA.



(*b*) New best solution of instance WR3 obtained by KMEDHEA.

Fig. 14. Gantt chart of the best solutions found by KMEDHEA on instances LS3 and WR3.

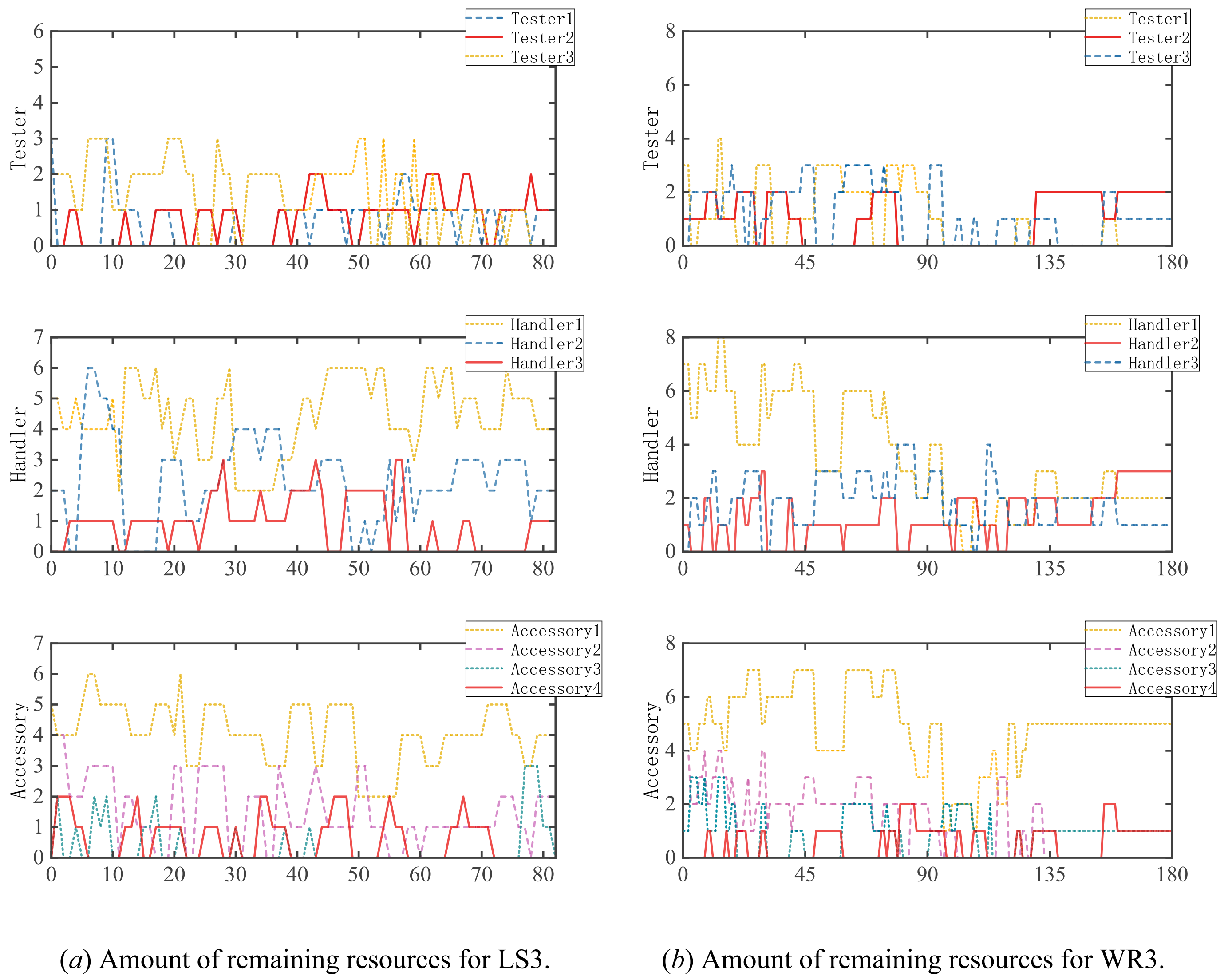


Fig. 15. Curve chart of remaining resources for instances LS3 and WR3.

As illustrated in Tables 12 and 13, KMEDHEA consistently outperforms all other algorithms across all instances regardless of the time factor λ, with the superiority becoming even more pronounced as the runtime factor increases. That is, as for 10 instances (*i.e.*, LS1-LS5, and WR1-WR5), our proposed KMEDHEA always delivers higher-quality scheduling solutions compared to NFOA, HEDA, KMEA, CCIWO, CSRS, and QHH, which confirms that KMEDHEA can efficiently address SFTSP. Specifically, in Table 12, the experimental results obtained by KMEDHEA under $n \times m \times 1$ seconds are overwhelmingly better than those obtained by the other six state-of-the-art algorithms on the BST metric. In terms of the AVG metric, KMEDHEA also demonstrates pretty good performance, with an average AVG value of 150.82, which is superior to the value of 153.59 obtained by CCIWO and the value of 183.49 obtained by NFOA in average senses, and only slightly suffers on the instance of WR2 as compared to CCIWO. With respect to the relative percent deviation of the average AVG, KMEDHEA's results were better than the other six algorithms, ranging from 1.8 % (*i.e.*, (153.59–150.82)/150.82≈0.018) to 21.7 % (*i.e.*, (183.49–150.82)/150.82≈0.217), respectively. Regarding STD metric, MEDEA (3.03) achieved favorable results on average STD versus NFOA (5.04), HEDA (3.25), KMEA (3.69), CCIWO (4.81), CSRS (7.62), and QHH (3.82), further observing that all the algorithms obtain the best values on 0, 2, 1, 0, 0, 0, 1, and 7 instances (see bold notation), which account for 0 %, 20 %, 10 %, 0 %, 0 %, 10 %, and 70 %, respectively, (KMEA and QHH obtained the same results on WR5). Furthermore, from the comparison results on the three metrics in Table 13, it is clear that KMEDHEA bests the other six compared algorithms, *i.e.*, NFOA, HEDA, KMEA, CCIWO, CSRS, and QHH, in terms of BST, AVG, and STD metrics in solving different scale instance when the running termination time is capped at $n \times m \times 5$ seconds. It is observed that KMEDHEA overwhelmingly outperforms the other six algorithms in BST, AVG, and STD metrics across all nine instances except for WR4, with the best results ranging from 7.2 % to 41.2 %, 8.4 % to 35.8 %, and 12.1 % to 461.0 %, respectively. The reason for excluding WR4 is that the optimal solution has almost been found, *i.e.*, the best makespan of WR4 may be 185 h. In addition, it is observed that the average values obtained by KMEDHEA on the STD metric across all instances in Tables 12 and 13 are also the smallest. Interestingly, the superiority of KMEDHEA may become more pronounced as the runtime factor λ increases. According to the above analysis, it can be concluded that the KMEDHEA is capable of minimizing the makespan for the SFTSP with highly efficient reliability and stability.

To provide comprehensive pictures of the effectiveness and stability of all compared algorithms in solving various instances under different termination conditions, the mean plots and box plots with 95 % Tukey's HSD confidence intervals for KMEDHEA and the other algorithms are shown in Fig. 12 and Fig. 13, respectively. As shown in Fig. 12, there are no overlapping intervals between KMEDHEA and the other algorithms. The upper edges of the intervals for KMEDHEA are significantly lower than the lower edges of the intervals for the other algorithms. These details indicate that KMEDHEA exhibits the most outstanding performance in terms of statistical significance. Fig. 13 demonstrates that the box lengths of KMEDHEA are shorter than those of the other compared algorithms in all scenarios except for WR2 under $\lambda = 1$. For the WR2 instance under $\lambda = 1$, the length of the box (IQR) obtained by KMEDHEA is comparable to that of CCIWO, yet still shorter than the other compared algorithms. Therefore, it can be confidently concluded that KMEDHEA has superior stability and robustness.

Based on extensive experiments and comparisons, it can be concluded that the proposed KMEDHEA stands as a newly efficient and effective learning-based paradigm for solving the SFTSP. This conclusion can be attributed to the following aspects: (1) Unlike population-based HIOAs such as NFOA, HDEA, KMEA, CCIWO, and CRCS, which rely on complex local search strategies, KMEDHEA only employs a set of simple LLHs to evolve effective heuristics, which can not only improve search efficiency but also be able to exhibit richer search behaviors. (2) Although QHH is effective, it evolves only for a single individual, which has lower diversity than KMEDHEA and is prone to premature convergence. KMEDHEA integrates three initialization strategies, resulting in producing the initial population with both good quality and diversity, which accelerates the convergence speed of the algorithm. (3) The global exploration of KMEDHEA can more effectively steer the search directions by applying the MEDA-based high-level strategy, while its local exploitation further adopts the most suitable sequences of LLHs to search for the solution space, especially embedding SA-based acceptance mechanism to efficiently jump out of local optimal regions. (4) The multidimensional probabilistic models in KMEDHEA allow for both extracting and estimating promising pattern features of LLH sequences to make proper decisions in the heuristic space, which is surely superior to most of the HHEAs that are based on typical HIOAs used as HLSs searching in the heuristic space. (5) The bi-layer framework of KMEDHEA adapts well to handle the complexity of the problem. It allows the search for superior solutions through the use of proper problem-specific LLHs, which may avoid search blindness compared to some typical operators commonly used in most of the HIOAs. (6) The constrained-separable left-shift decoding scheme developed based on critical constraints further improves the quality of the superior solutions.

Furthermore, Fig. 14 presents Gantt charts illustrating the best solutions found so far by KMEDHEA for two distinct instances: the large-scale LS3 (consisting of 100 jobs, 36 machines, and 196 operations) and the wide-range WR3 (consisting of 60 jobs, 36 machines, and 123 operations). In Fig. 14(*a*) and 14(*b*), it is clear that the makespan for the two feasible scheduling solutions stands at 82 *h* and 180 *h*, respectively. Additionally, Fig. 15 provides a comprehensive view of the remaining resources (*i.e.*, testers, handlers, and accessories), which serves as compelling evidence that the derived scheduling solutions are indeed feasible. As observed in both Figs. 14 and 15, the constrained-separable left-shift decoding scheme employed ensures that operations on each machine are shifted to the earliest possible time slots, thereby greatly reducing machine idle time, effectively improving resource utilization, and ultimately minimizing production cycles. Moreover, the real-time residual resource curves depict the best utilization of each resource while meticulously adhering to multi-resource constraints. These observations, coupled with the extensive experimental results presented, can be confidently concluded that the proposed KMEDHEA emerges as a robust and efficient learning-based paradigm for effectively tackling the considerable challenges associated with SFTSP.

6. Managerial implications

This study introduces an emerging framework of HHEA (*i.e.*, KMEDHEA) that develops a highly effective and efficient MEDA-based learning paradigm capable of autonomously learning promising patterns, capturing critical characteristics, and enabling collaborative interactions to efficiently address complex challenges posed by SFTSP. Through thorough computational comparisons, the excellent efficacy of KMEDHEA is explicitly demonstrated. Furthermore, the detailed case study of ten instances highlights the potential of KMEDHEA for practical applications in semiconductor manufacturing environments. This research contributes significantly to the field of evolutionary computation and provides valuable insights for practitioners and decision-makers. Building upon these insightful findings, this section delves into the managerial implications of implementing KMEDHEA for complex FJSPs with multi-resource constraints, with specific points listed below:

(1) KMEDHEA, as a newly-emerged learning-based paradigm, is capable of effectively extracting critical characteristics and efficiently seeking for superior scheduling solutions within acceptable timeframes, which not only can enhance test efficiency but also accelerate the semiconductor manufacturing process. With the ongoing advances in both HHEAs and SFT domains, various learning-based paradigms are thriving, and enterprises' demand for test scheduling is steadily rising. Therefore, there is substantial value in investigating active learning-based paradigms and knowledge-enhanced scheduling strategies based on problem-specific properties.

(2) SFTSP faces significant challenges due to considerations such as the processing priority of operations, machine assignment, and resource allocation, resulting in the complex and constantly changing search space posing difficulties for specific problem-solving approaches. KMEDHEA's innovative bi-layer framework is well-equipped not only to address these complexities but also to provide superior search strengths that excel in guiding global exploration, facilitating the search for superior solutions through the best sequences of problem-specific LLHs. Unlike typical operators commonly found in most HIOAs, KMEDHEA's framework avoids search futility and flexibly adapts to inter-layer interactions. This adaptability ensures the efficient and effective exploration and exploitation of potential regions in the strategy and solution spaces, which is crucial in confronting the changing challenges and multi-resource constraints commonly encountered in real-world scenarios.

(3) Scheduling for SFT in semiconductor fabrication presents one of the core challenges due to setup time restrictions and resource constraints. To address these challenges, this study introduces a novel constrained-separable left-shift decoding scheme that takes into account constraint preferences. This decoding scheme gives priority to the setup time to ensure that critical preparation steps are promptly executed before jobs arrive and machines become available, eliminating unnecessary waiting time and enabling effective enhancements in test equipment utilization and production efficiency. Once all jobs have arrived and machines are ready, this scheme shifts its focus to adapting to resource constraints. This constrained-separable left-shift decoding scheme enables rapid adaptation of test resources and compact scheduling of test operations, even in response to complex and dynamic constraints. It provides valuable insights and ideas that can be applied to addressing complex constraints in flexible manufacturing systems.

7. Findings, research limitations, and future work

7.1. Findings

In response to the pressing need for enhanced efficiency in the chip and integrated circuit industry, the pursuit of scheduling strategies and solving techniques in semiconductor final testing (SFT) has become a driving force. However, the inherent intricacies of semiconductor production processes present significant challenges for SFT. This insightful study, which pioneered the application of HHEAs in collaboration with MEDA tailored to tackle the SFTSP, is the first attempt to adopt a multidimensional probabilistic model for learning promising patterns from sequences of LLHs. It aims to establish a learning-based paradigm that promotes inter-layer connections and collaboration through the application of HHEA, signifying a significant step forward in tackling the complexity of FJSPs with multiple resources. The findings of this study are summarized below.

(1) Establishing a novel operation-based sequence model for the SFTSP opens fresh insights for solving such complex problems.

(2) Designing a single-vector encoding scheme and a constrained-separable left-shift decoding scheme not only represents feasible solutions effectively but also decodes them into scheduling schemes efficiently.

(3) Developing a hybrid initialization method that consists of three problem-dependent heuristic rules for generating a high-quality initial population, ensuring that the algorithm has a strong start in searching within the solution space.

(4) Creating a set of problem-specific heuristics that serve as an LLH pool for fine-grained exploration of the solution space. The introduction of the SA-based acceptance mechanism further enhances the algorithm's ability to escape from local optima, ensuring a more adaptive and efficient search process.

(5) Applying MEDA as the HLS of HHEA to manipulate a set of easy-to-implement LLHs, and employing a 3-D probabilistic model to accurately record the block structure and block distribution of LLHs.

(6) Demonstrating the superiority and stability of MEDA-based HHEA through extensive computational comparisons. Notably, KMEDHEA outperforms the existing best solutions in 9 out of the 10 benchmark instances, representing a remarkable advance in tackling SFTSP.

7.2. Research limitations

The research limitations are twofold, both in terms of the limitations of the proposed MEDA-based HHEA and the impediments in its practical application. Firstly, despite the demonstrated effectiveness of KMEDHEA, it still showed less stability in almost all test instances when subjected to shorter runtimes. This observation prompts the necessity to explore adaptive knowledge-driven HLSs that can reduce computational resource requirements, thereby enhancing the efficiency and stability of HHEAs. Secondly, fine-tuning the parameters depending on the actual size of the problem is a time-consuming process, inevitably impacting the production efficiency of enterprises and posing challenges to the practical application of MEDA-based HHEA. To address these challenges, advanced forward-looking strategies should be emphasized, such as integrating multi-agent deep reinforcement learning (MADRL) and graph neural networks (GNN) with KMEDHEA, envisioned to facilitate self-tuning of parameters and self-learning of heuristics, thereby potentially mitigating the impact of the adjustment process on production efficiency.

7.3. Future work

In forthcoming research, the proposed MEDA-based HHEA holds significant potential for extensions, particularly in addressing multi-objective SFTSPs. These extensions typically involve integrating optimization objectives, such as electricity cost, energy consumption, and machine preventive maintenance, reflecting the diverse concerns commonly confronted in production practice. The study scope of this work can be further extended to address dynamic SFTSPs, such as scenarios with stochastic processing times and machine breakdowns, considering uncertainties and suddenness in real-life manufacturing environments. Moreover, it is interesting to investigate effective and efficient learning-based techniques by incorporating problem-specific knowledge and promising pattern information into the comprehensive framework of HHEAs. Furthermore, the proposed innovative learning-based hyper-heuristic framework can be extended and applied to address dynamic workflow scheduling in cloud computing environments and multi-task scheduling in cloud-edge collaborative manufacturing scenarios. Meanwhile, it is worthwhile to study the application of deep reinforcement learning-based HHEAs, which can provide valuable insights into dealing with dynamic scheduling problems in real-life manufacturing scenarios, providing promising perspectives on the design of innovative methods in evolutionary computation domains.

8. Conclusions

This paper presents an innovative multidimensional EDA-based hyper-heuristic evolutionary algorithm (HHEA) for solving the SFTSP, primarily focusing on minimizing the makespan. To the best of the authors' knowledge, this article is the first attempt to apply HHEA combined with multidimensional EDA to solve the SFTSP. First, based on the problem characteristics of SFTSP, an operation-based sequence model for SFTSP is formulated that takes into account operations and machine constraints. Second, single-vector encoding and constrained-separable left-shift decoding schemes are designed to represent feasible scheduling solutions and transform them into superior scheduling schemes. Third, eight simple yet effective heuristics are adopted as the LLH set, and a 3-D probabilistic model is applied to learn and accumulate crucial information such as the order of LLHs, the similar block of LLHs, and the positional relationships among LLHs so that MEDA-based HLS enables estimating promising patterns and predicting LLHs for the next position based on the current status and to guide the global search towards potential regions through in-depth insights into the patterns of LLHs. Finally, from extensive computational results, it has been demonstrated that the proposed KMEDHEA can achieve better performance in terms of superiority and stability than the existing state-of-the-art algorithms, especially for large-scale problems.

CRediT authorship contribution statement

Zi-Qi Zhang: Conceptualization, Methodology, Funding acquisition, Writing – original draft. **Xing-Han Qiu:** Investigation, Methodology, Software, Writing – original draft. **Bin Qian:** Methodology, Funding acquisition, Supervision, Writing – review & editing. **Rong Hu:** Funding acquisition, Supervision, Writing – review & editing. **Ling Wang:** Supervision, Project administration. **Jian-Bo Yang:** Supervision.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

The authors sincerely thank the anonymous reviewers for their insightful comments and suggestions, which greatly improve this paper. This work was financially supported by the National Natural Science Foundation of China (Grant Nos. 72201115, 62173169, and 61963022), the Yunnan Fundamental Research Projects (Grant No. 202201BE070001-050 and 202301AU070069), and the Basic Research Key Project of Yunnan Province (Grant No. 202201AS070030), and the Construction Project of Higher Educational Key Laboratory for Industrial Intelligence and Systems of Yunnan Province (Grant No. KKPH202403003).

References

Asta, S., Özcan, E., & Curtois, T. (2016). A tensor based hyper-heuristic for nurse rostering. *Knowledge-Based Systems, 98*, 185–199.

Branke, J., Nguyen, S., Pickardt, C. W., & Zhang, M. (2016). Automated Design of Production Scheduling Heuristics: A Review. *IEEE Transactions on Evolutionary Computation, 20*, 110–124.

Burke, E. K., Gendreau, M., Hyde, M., Kendall, G., Ochoa, G., Özcan, E., & Qu, R. (2013). Hyper-heuristics: A survey of the state of the art. *Journal of the Operational Research Society, 64*, 1695–1724.

Cao, Z., Lin, C., Zhou, M., & Huang, R. (2019). Scheduling Semiconductor Testing Facility by Using Cuckoo Search Algorithm With Reinforcement Learning and Surrogate Modeling. *IEEE Transactions on Automation Science and Engineering, 16*, 825–837.

Chen, H., Ding, G., Qin, S., & Zhang, J. (2021). A hyper-heuristic based ensemble genetic programming approach for stochastic resource constrained project scheduling problem. *Expert Systems with Applications, 167*, Article 114174.

Chen, H., Zhang, J., Li, R., Ding, G., & Qin, S. (2022). A two-stage genetic programming framework for Stochastic Resource Constrained Multi-Project Scheduling Problem under New Project Insertions. *Applied Soft Computing, 124*, Article 109087.

Chen, T.-R., & Hsia, T. C. (1994). *Job shop scheduling with multiple resources and an application to a semiconductor testing facility* (Vol. 2,, 1564–1570.

Dass, S. N., & Feng, C. J. (2022). Change Qualification Framework in Semiconductor Manufacturing. *IEEE Transactions on Semiconductor Manufacturing, 35*, 87–101.

Expósito-Izquierdo, C., González-Velarde, J. L., Melián-Batista, B., & Marcos Moreno-Vega, J. (2013). Hybrid Estimation of Distribution Algorithm for the Quay Crane Scheduling Problem. *Applied Soft Computing, 13*, 4063–4076.

Fan, H., Xiong, H., & Goh, M. (2021). Genetic programming-based hyper-heuristic approach for solving dynamic job shop scheduling problem with extended technical precedence constraints. *Computers & Operations Research, 134*, Article 105401.

Fattahi, P., Mehrabad, M. S., & Jolai, F. (2007). Mathematical modeling and heuristic approaches to flexible job shop scheduling problems. *Journal of Intelligent Manufacturing, 18*, 331–342.

Freed, T., & Leachman, R. C. (1999). Scheduling semiconductor device test operations on multihead testers. *IEEE Transactions on Semiconductor Manufacturing, 12*, 523–530.

Hao, X. C., Wu, J. Z., Chien, C. F., & Gen, M. (2014). The cooperative estimation of distribution algorithm: A novel approach for semiconductor final test scheduling problems. *Journal of Intelligent Manufacturing, 25*.

Hinkelmann, K., & Kempthorne, O. (2008). *Design and Analysis of Experiments Set*: Design and analysis of experiments.

Hu, W., Liu, M., Dong, M., Liu, T., Zhang, Y., & Cheng, G. (2023). A greedy-based crow search algorithm for semiconductor final testing scheduling problem. *Computers & Industrial Engineering, 183*, Article 109423.

Huanxin Henry, X., & Zhou, M. (1998). Scheduling of semiconductor test facility via Petri nets and hybrid heuristic search. *IEEE Transactions on Semiconductor Manufacturing, 11*, 384–393.

Jarboui, B., Eddaly, M., & Siarry, P. (2009). An estimation of distribution algorithm for minimizing the total flowtime in permutation flowshop scheduling problems. *Computers & Operations Research, 36*, 2638–2646.

Kieffer, E., Danoy, G., Brust, M. R., Bouvry, P., & Nagih, A. (2020). Tackling Large-Scale and Combinatorial Bi-Level Problems With a Genetic Programming Hyper-Heuristic. *IEEE Transactions on Evolutionary Computation, 24*, 44–56.

Kim, J., Nam, Y., Kang, M. C., Kim, K., Hong, J., Lee, S., & Kim, D. N. (2021). Adversarial Defect Detection in Semiconductor Manufacturing Process. *IEEE Transactions on Semiconductor Manufacturing, 34*, 365–371.

Lee, G. T., Lim, H. G., & Jang, J. (2023). Sequential Residual Learning for Multistep Processes in Semiconductor Manufacturing. *IEEE Transactions on Semiconductor Manufacturing, 36*, 37–44.

Lei, D., & Guo, X. (2014). Variable neighbourhood search for dual-resource constrained flexible job shop scheduling. *International Journal of Production Research, 52*, 2519–2529.

Lim, K. C. W., Wong, L.-P., & Chin, J. F. (2022). Simulated-annealing-based hyper-heuristic for flexible job-shop scheduling. *Engineering Optimization*, 1–17.

Lin, J. (2019). Backtracking search based hyper-heuristic for the flexible job-shop scheduling problem with fuzzy processing time. *Engineering Applications of Artificial Intelligence, 77*, 186–196.

Lin, J., Li, Y.-Y., & Song, H.-B. (2022). Semiconductor final testing scheduling using Q-learning based hyper-heuristic. *Expert Systems with Applications, 187*, Article 115978.

Lin, J., Wang, Z.-J., & Li, X. (2017). A backtracking search hyper-heuristic for the distributed assembly flow-shop scheduling problem. *Swarm and Evolutionary Computation, 36*, 124–135.

Lin, J. T., Wang, F. K., & Lee, W. T. (2004). Capacity-constrained scheduling for a logic IC final test facility. *International Journal of Production Research, 42*, 79–99.

Liu, C., Chen, H., Xu, R., & Wang, Y. (2018). Minimizing the resource consumption of heterogeneous batch-processing machines using a copula-based estimation of distribution algorithm. *Applied Soft Computing, 73*, 283–305.

Mahmud, S., Abbasi, A., Chakrabortty, R. K., & Ryan, M. J. (2022). A self-adaptive hyper-heuristic based multi-objective optimisation approach for integrated supply chain scheduling problems. *Knowledge-Based Systems, 251*, Article 109190.

Mousakhani, M. (2013). Sequence-dependent setup time flexible job shop scheduling problem to minimise total tardiness. *International Journal of Production Research, 51*, 3476–3487.

Ovacik, I. M., & Uzsoy, R. (1996). Decomposition methods for scheduling semiconductor testing facilities. *International Journal of Flexible Manufacturing Systems, 8*, 357–387.

Pan, Q.-K., & Ruiz, R. (2012). An estimation of distribution algorithm for lot-streaming flow shop problems with setup times. *Omega, 40*, 166–180.

Park, J., Mei, Y., Nguyen, S., Chen, G., & Zhang, M. (2018). An investigation of ensemble combination schemes for genetic programming based hyper-heuristic approaches to dynamic job shop scheduling. *Applied Soft Computing, 63*, 72–86.

Pearn, W. L., Chung, S. H., Chen, A. Y., & Yang, M. H. (2004). A case study on the multistage IC final testing scheduling problem with reentry. *International Journal of Production Economics, 88*, 257–267.

Sabar, N. R., Ayob, M., Kendall, G., & Qu, R. (2013). Grammatical Evolution Hyper-Heuristic for Combinatorial Optimization Problems. *IEEE Transactions on Evolutionary Computation, 17*, 840–861.

Sabar, N. R., & Kendall, G. (2015). Population based Monte Carlo tree search hyper-heuristic for combinatorial optimization problems. *Information Sciences, 314*, 225–239.

Sang, H.-Y., Duan, P.-Y., & Li, J.-Q. (2018). An effective invasive weed optimization algorithm for scheduling semiconductor final testing problem. *Swarm and Evolutionary Computation, 38*, 42–53.

Shang, C., Ma, L., Liu, Y., & Sun, S. (2022). The sorted-waste capacitated location routing problem with queuing time: A cross-entropy and simulated-annealing-based hyper-heuristic algorithm. *Expert Systems with Applications, 201*, Article 117077.

Song, H.-B., & Lin, J. (2021). A genetic programming hyper-heuristic for the distributed assembly permutation flow-shop scheduling problem with sequence dependent setup times. *Swarm and Evolutionary Computation, 60*, Article 100807.

Song, H.-B., Yang, Y.-H., Lin, J., & Ye, J.-X. (2023). An effective hyper heuristic-based memetic algorithm for the distributed assembly permutation flow-shop scheduling problem. *Applied Soft Computing, 135*, Article 110022.

Tiwari, A., Chang, P.-C., Tiwari, M. K., & Kollanoor, N. J. (2015). A Pareto block-based estimation and distribution algorithm for multi-objective permutation flow shop scheduling problem. *International Journal of Production Research, 53*, 793–834.

Uzsoy, R., Church, L. K., Ovacik, I. M., & Hinchman, J. I. M. (1993). Performance evaluation of dispatching rules for semiconductor testing operations. *Journal of Electronics Manufacturing, 3*, 95–105.

Uzsoy, R., Lee, C.-Y., & Martin-Vega, L. A. (1992). Scheduling semiconductor test operations: Minimizing maximum lateness and number of tardy jobs on a single machine. *Naval Research Logistics, 39*, 369–388.

Uzsoy, R., Martin-Vega, L. A., Lee, C. Y., & Leonard, P. A. (1991). Production scheduling algorithms for a semiconductor test facility. *IEEE Transactions on Semiconductor Manufacturing, 4*, 270–280.

Wang, L., Wang, S., Xu, Y., Zhou, G., & Liu, M. (2012). A bi-population based estimation of distribution algorithm for the flexible job-shop scheduling problem. *Computers & Industrial Engineering, 62*, 917–926.

Wang, S., & Wang, L. (2014). Compact estimation of distribution algorithm for semiconductor final testing scheduling problem. In *2014 IEEE International Conference on Automation Science and Engineering (CASE)* (pp. 113–118). IEEE: Taipei, Taiwan.

Wang, S., & Wang, L. (2015). A knowledge-based multi-agent evolutionary algorithm for semiconductor final testing scheduling problem. *Knowledge-Based Systems, 84*, 1–9.

Wang, S., Wang, L., Liu, M., & Xu, Y. (2013). A hybrid estimation of distribution algorithm for the semiconductor final testing scheduling problem. *Journal of Intelligent Manufacturing, 26*, 861–871.

Wu, C.-C., Bai, D., Chen, J.-H., Lin, W.-C., Xing, L., Lin, J.-C., & Cheng, S.-R. (2021). Several variants of simulated annealing hyper-heuristic for a single-machine scheduling with two-scenario-based dependent processing times. *Swarm and Evolutionary Computation, 60*, Article 100765.

Wu, J.-Z., & Chien, C.-F. (2008). Modeling semiconductor testing job scheduling and dynamic testing machine configuration. *Expert Systems with Applications, 35*, 485–496.

Wu, J. Z., Hao, X. C., Chien, C. F., & Gen, M. (2012). A novel bi-vector encoding genetic algorithm for the simultaneous multiple resources scheduling problem. *Journal of Intelligent Manufacturing, 23*, 2255–2270.

Zhang, Z.-Q., Hu, R., Qian, B., Jin, H.-P., Wang, L., & Yang, J.-B. (2022). A matrix cube-based estimation of distribution algorithm for the energy-efficient distributed assembly permutation flow-shop scheduling problem. *Expert Systems with Applications, 194*, Article 116484.

Zhang, Z.-Q., Qian, B., Hu, R., Jin, H.-P., & Wang, L. (2021). A matrix-cube-based estimation of distribution algorithm for the distributed assembly permutation flow-shop scheduling problem. *Swarm and Evolutionary Computation, 60*, Article 100785.

Zhang, Z.-Q., Qian, B., Hu, R., Jin, H.-P., Wang, L., & Yang, J.-B. (2022). A matrix-cube-based estimation of distribution algorithm for blocking flow-shop scheduling problem with sequence-dependent setup times. *Expert Systems with Applications, 205*, Article 117602.

Zhang, Z. Q., Qian, B., Hu, R., & Yang, J. B. (2023). Q-learning-based hyper-heuristic evolutionary algorithm for the distributed assembly blocking flowshop scheduling problem. *Applied Soft Computing, 146*, Article 110695.

Zhang, Z. Q., Wu, F. C., Qian, B., Hu, R., Wang, L., & Jin, H. P. (2023). A Q-learning-based hyper-heuristic evolutionary algorithm for the distributed flexible job-shop scheduling problem with crane transportation. *Expert Systems with Applications, 234*, Article 121050.

Zhao, F., Di, S., Wang, L., Xu, T., Zhu, N., & Jonrinaldi. (2022). A self-learning hyper-heuristic for the distributed assembly blocking flow shop scheduling problem with total flowtime criterion. *Engineering Applications of Artificial Intelligence, 116*, 105418.

Zhao, F., Zhu, B., & Wang, L. (2023). An Estimation of Distribution Algorithm-Based Hyper-Heuristic for the Distributed Assembly Mixed No-Idle Permutation Flowshop Scheduling Problem. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 1–12.

Zheng, X.-L., Wang, L., & Wang, S.-Y. (2014). A novel fruit fly optimization algorithm for the semiconductor final testing scheduling problem. *Knowledge-Based Systems, 57*, 95–103.

Zhu, L., Lin, J., Li, Y.-Y., & Wang, Z.-J. (2021). A decomposition-based multi-objective genetic programming hyper-heuristic approach for the multi-skill resource constrained project scheduling problem. *Knowledge-Based Systems, 225*, Article 107099.