Swarm and Evolutionary Computation 87 (2024) 101559

Contents lists available at ScienceDirect

Swarm and Evolutionary Computation

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

An enhanced estimation of distribution algorithm with problem-specific knowledge for distributed no-wait flowshop group scheduling problems

Zi-Qi Zhang [a,b], Yan-Xuan Xu [a], Bin Qian [a,b,*], Rong Hu [a,b], Fang-Chun Wu [a], Ling Wang [c]

[a] School of Information Engineering and Automation, Kunming University of Science and Technology, Kunming 650500, PR China
[b] Yunnan Key Laboratory of Artificial Intelligence, Kunming University of Science and Technology, Kunming 650500, PR China
[c] Department of Automation, Tsinghua University, Beijing 100084, PR China

ARTICLE INFO

Keywords:
Flowshop group scheduling
Distributed no-wait flowshop group scheduling
Estimation of distribution algorithm
Problem-specific knowledge

ABSTRACT

With the trend of economic globalization, distributed manufacturing widely exists in modern manufacturing systems. As an extension of the distributed flowshop scheduling problem, the distributed no-wait flowshop group scheduling problem with sequence-dependent setup times (DNFGSP_SDSTs) is investigated in this article. To address DNFGSP_SDSTs with the criterion of minimizing makespan, this study proposes an enhanced estimation of distribution algorithm·(EEDA) with problem-specific knowledge. First, a mixed integer linear programming (MILP) model of DNFGSP_SDSTs is established. Second, based on the characteristics of DNFGSP_SDSTs, five problem-specific properties about local search operators are derived as prior knowledge to reduce computational cost. Third, two NEH-based two-stage heuristics are presented to construct a high-quality population with diversity. Fourth, a probability model with problem-specific knowledge and a family-based updating mechanism are developed to accumulate valuable pattern information from high-quality solutions, while a sampling strategy is designed to generate new populations with the accumulated information. Fifth, several local search operators are devised to refine the obtained solutions. Furthermore, perturbation and reinitialization methods are developed to avoid premature convergence. Finally, the validity of the MILP model is verified by using the Gurobi solver. The parameters of EEDA are tuned through a design of experiments. The effectiveness of key components in EEDA is confirmed through extensive experiments, and the computational comparisons with the state-of-the-art algorithms indicate the effectiveness of the proposed EEDA for solving DNFGSP_SDSTs.

1. Introduction

Amidst the backdrop of economic globalization and the advent of industrial intelligence, the role of distributed manufacturing has gained increasing importance in efficiently coordinating resources across regions, flexibly adapting to demands, reducing costs, and enhancing efficiency, which has proactively promoted the upgrading of modern supply chains and cell manufacturing systems (CMS) [1,2]. As a crucial challenge within distributed manufacturing, the distributed flowshop group scheduling problem (DFGSP) has attracted considerable attention and dedicated research efforts [3]. Unlike traditional flowshop production, distributed flowshop group production has the advantages of flexible resource allocation and rapid adaptation to requirements, prioritizing cost-efficient group lots and facilitating mass-customized production [4]. Given these significant strengths, it is vital to study distributed scheduling problems (DSPs), specifically those involving collaborative production across multiple factories. Hence, employing emerging techniques to develop effective and efficient approaches for addressing DSPs holds both academic interest and practical significance [5].

Over the past decade, substantial scholarly studies have surfaced, mainly focusing on DSPs and their extensions, with distributed flowshop scheduling problems (DFSPs) emerging as a prominent focus [6]. Pioneering work by Naderi and Ruiz [7] in modeling and optimizing DFSPs have laid the foundation for research on various variants in distributed environments. The concept of CMS has garnered attention for its potential to enhance efficiency and flexibility in production processes by grouping similar jobs into families and processing them in specific cells, thus reducing setup time and inventory cost. This concept, pioneered by Radharamanan [8], has found successful studies and applications across industries, including automotive, smart furniture, semiconductor, and electronics [9]. However, practical production practices necessitate

* Corresponding author.
E-mail address: bin.qian@vip.163.com (B. Qian).

https://doi.org/10.1016/j.swevo.2024.101559
Received 4 February 2024; Received in revised form 13 March 2024; Accepted 26 March 2024
Available online 5 April 2024

scheduling considerations that account for setup times, which play pivotal roles in resource readiness and just-in-time delivery [10]. In general, the durations of operations, such as machine cleaning, tool changeover, and job transfer between machines, often referred to as sequence-dependent setup times (SDSTs), depend on both the current and subsequent families in lines [11]. For FGSP, jobs are grouped based on common characteristics of the production process and processing priorities within cells [12]. Therefore, it becomes essential to consider only setup times between families, enabling to exclude setup time considerations among jobs within the families, enhancing efficiency and reducing waiting times [13]. Given current interests in SDSTs, further in-depth research on FGSPs with SDSTs is of practical significance. In addition, considering real-world applications in industries such as steelmaking, food processing, and chemical and pharmaceutical industries, it is imperative to take into account both SDSTs and no-wait constraints [14]. The no-wait constraints require that job processing cannot be interrupted on consecutive machines [15]. To meet no-wait condition in practical production processes, substantial scholarly studies have emerged in recent years [16–20]. Despite the significance of solving no-wait FGSP_SDSTs (NFGSP_SDSTs) has been stressed, it has not been sufficiently studied in distributed manufacturing. Given the increasing interest of DSPs in CMS, it has become increasingly important to handle the complex constraints of distributed NFGSP_SDSTs (DNFGSP_SDSTs). As a response to this need, this study undertakes the challenge of addressing DNFGSP_SDSTs.

DNFGSP_SDSTs commonly comprises coupled subproblems, including the family assignment subproblem and the job processing order subproblem. The crucial challenge lies in determining the allocation of families to cellulars, the arrangement of families within each cellular, and the order of jobs within each family to minimize the makespan. Both of subproblems inherently involve SDSTs and no-wait constraints, rendering them highly complex and challenging. Notably, the FGSP with SDSTs is NP-hard in a strong sense [21], whereas the FGSP_SDSTs is a generalization of the DNFGSP_SDSTs, which is also NP-hard. To facilitate clear representation, Graham et al. [22] introduced the three-field notation $\alpha|\beta|\gamma$, where α is the production environment, and β and γ denote the process constraints and criteria, respectively. For DNFGSP_SDSTs aiming to minimize the makespan, it can be denoted as $F_m,\ldots,F_m|fmls,nwt,SDSTs|C_{\max}$ [23]. Here, F_m represents a flowshop with m machines, $F_m, \ldots, F_m$ represents distributed flowshops with m machines, *fmls*, *nwt*, *SDSTs* represent group processing, no-wait constraint, and SDSTs, respectively, and $C_{\max}$ refers to the makespan. DFGSP_SDSTs can be represented as $F_m, \ldots, F_m|fmls, SDSTs|C_{\max}$ and NFGSP_SDSTs as $F_m, \ldots, F_m|fmls, nwt, SDSTs|C_{\max}$. Since $F_m, \ldots, F_m|fmls, SDSTs|C_{\max}$ and $F_m|fmls, nwt, SDSTs|C_{\max}$ have been proven to be NP-hard [18,24], it follows that the strongly constrained variant $F_m, \ldots, F_m|fmls, nwt, SDSTs|C_{\max}$ of both is also NP-hard. To address the NP-hardness challenges effectively, there are both exact and approximate approaches available. Exact approaches, such as mixed integer linear programming (MILP), constraint programming (CP), column generation (CG) and benders decomposition (BD), are available but may cause considerable computational costs, especially when dealing with large-sized FGSPs, limiting their practical utility. Conversely, approximate approaches, including constructive heuristics and metaheuristics, provide practical insights for solving FGSPs and their extensions in distributed environments. Constructive heuristics can quickly generate feasible solutions based on problem characteristics and constraints, although the quality of these solutions may not always be desirable. However, extracting features or formulating effective heuristic rules for addressing complex FGSPs may be challenging. As typical approaches in metaheuristics, hybrid intelligent optimization algorithms (HIOAs) have garnered attention for their exceptional efficacy in solving scheduling problems. HIOAs commonly combine problem-specific global search frameworks and local search strategies based on feasible encodings. Specifically, the potential of HIOAs in solving strongly coupled scheduling problems is stimulated by well-integrated use of constructive heuristics and local search operators, along with mechanisms like diversity maintenance to effectively balance global and local search, thus yielding high-quality solutions within a limited timeframe [23]. Various studies have employed HIOAs, such as GA [13,16,25-27], TS [28,29], PSO [17,30], SA [19,31,32], and ACO [33], to tackle FGSPs, DFGSPs, and their variants. A review of recent related research on FGSPs is summarized in Table 1.

Among HIOAs, the estimation of distribution algorithm (EDA) stands out as holding significant research value and promising prospects, as probability modeling and implicit parallelism provide novel insights and methodologies for addressing various types of complex and large-scale problems [20,34]. By utilizing probability models to estimate the structural and distributional characteristics of high-quality solutions, EDA has been demonstrated to explore complex search spaces effectively, efficiently guiding the search scope towards promising regions for seeking superior solutions. When addressing DNFGSP_SDSTs, family constraints constrain the allocation of jobs, no-wait constraints tighten the connections between jobs, and SDSTs stresses the significance of family sequencing for the quality of solutions, all of which cause challenges. Although existing EDAs excelled at learning relationships between jobs, they struggled to effectively capture correlations of families, resulting in the oversight of specific features crucial to superior solutions. Inspired by multiple probability model-based EDAs [14,35] and multidimensional probability model-based EDAs (MEDAs) [20,36-39], this study presents a knowledge-enhanced EDA tailored for DNFGSP_SDSTs. This EEDA focuses on extracting and estimating information between adjacent jobs and patterns across families. To capture the connections between families and between jobs, the probability model is divided into multiple partition matrices based on the relationships of intra- and inter-family jobs, respectively. A family-based updating mechanism is used to extract potential patterns from high-quality solutions, and promising patterns are converted into probabilities to be accumulated in these partition matrices. Next, a specific sampling strategy is used to yield new solutions with certain quality by extracting valuable information from this probability model. Compared to previous EDAs, an enhanced EDA with problem-specific knowledge has been developed for DNFGSP_SDSTs, aiming to retain EDA's strengths while adapting to problem's characteristics for seeking superior solutions. The prospect of this knowledge-enhanced learning paradigm in tackling the challenges posed by DNFGSP_SDSTs seems promising.

As mentioned above, the main contributions of this article are highlighted below.

- The MILP model for DNFGSP_SDSTs is developed, which considers family assignment and job arrangement subproblems with no-wait constraints and sequence-dependent setup times.
- Five problem properties are derived as problem-specific knowledge, and several knowledge-based speedup strategies are designed to save computational efforts in evaluating solutions.
- To efficiently solve DNFGSP_SDSTs, EEDA with problem-specific knowledge is presented. In EEDA, the probability model is designed by partitioning matrices used to record relations of intra- and inter-family jobs, which captures the connections between families and between jobs to effectively guide the global search towards promising regions.
- To yield high-quality initial solutions with certain diversity, an effective hybrid initialization method based on two problem-specific heuristics is proposed, which can reasonably refine population quality and reduce the processing times for both job and family sequences.
- A family-based updating mechanism and a novel sampling strategy are embedded in EEDA to estimate the distribution of crucial characteristics and execute effective exploration in the solution space, so as to efficiently direct search scopes toward promising regions for seeking superior solutions, while both perturbation and

Table 1
Summarization of papers on FGSPs.

Source	Problem characteristics							Year	Approaches	Problem
	STs	*nwt*	*block*	*tst*	*rd*	flexible	*DM*			
Ying et al. [31]	√							2010	SA	$F_m\|fmls, SDSTs\|C_{\max}$
Karimi et al. [25]	√					√		2010	MOGA	$FF_s\|fmls, SDSTs\|C_{\max}$, TWT
Solimanpur et al. [28]			√					2011	TS	$F_m\|fmls, block\|C_{\max}$
Lin et al. [32]	√					√		2011	SA	$FF_s\|fmls, SDSTs\|C_{\max}$
Hajinejad et al. [30]	√							2011	PSO	$F_m\|fmls, SDSTs\|$TCT
Naderi et al. [40]	√							2012	GSA	$F_m\|fmls, SDSTs\|$TCT
Ying et al. [16]	√	√						2012	MM, GA, SA, IG	$F_m\|fmls, SDSTs, nwt\|C_{\max}$
Keshavarz et al. [41]	√					√		2013	MA, LBD	$FF_s\|fmls, SDSTs\|C_{\max}$
Liou et al. [42]	√		√	√				2013	LBD, GA, PSO	$F_2\|fmls, SDSTs, tst\|C_{\max}$
Lu et al. [29]	√				√			2013	MM, TS	$F_m\|fmls, SDSTs, rd\|$TWT, TWCT
Ebrahimi et al. [13]	√							2014	NSGA-II, MOGA	$F_m\|fmls, SDSTs\|C_{\max}$, TT
Liou et al. [43]	√			√				2015	LBD, GA, PSO	$F_m\|fmls, SDSTs, tst\|C_{\max}$
Keshavarz et al. [44]	√					√		2015	LBD, MA	$FF_s\|fmls, SDSTs\|C_{\max}$
Li et al. [45]	√							2015	HHS	$F_m\|fmls, SDSTs\|$TCT, TT
Behjat et al. [17]	√	√						2017	PSO, VNS	$F_m\|fmls, SDSTs, nwt\|$TCT
Yazdani Sabouni et al. [46]	√							2018	LBD, MM, B&P	$F_m\|fmls, SDSTs\|$TWT, TWCT
Lin et al. [18]	√	√						2019	MM, SA, LKH	$F_m\|fmls, SDSTs, nwt\|C_{\max}$
Huang et al. [47]							√	2020	HHS	$F_m, ..., F_m\|fmls, SDSTs\|C_{\max}$
Yuan et al. [21]	√			√				2020	LBD, DDE	$F_m\|fmls, SDSTs, tst\|C_{\max}$
Cheng et al. [19]	√	√						2021	RMSA	$F_m\|fmls, SDSTs, nwt\|$TCT
Goli et al. [48]	√							2022	BBO, VNS	$F_m\|fmls, SDSTs\|$TWET
Pan et al. [3]	√						√	2022	CCEA	$F_m, ..., F_m\|fmls, SDSTs\|C_{\max}$
Wang et al. [49]	√						√	2022	IG	$F_m, ..., F_m\|fmls, SDSTs\|$TT
Wang et al. [24]	√						√	2023	IG	$F_m, ..., F_m\|fmls, SDSTs\|C_{\max}$

Note: *STs*: SDSTs, *nwt*: no-wait, *block*: a blockage of the machine due to limited buffer space of the next machine, *tst*: transportation time,*rd*: release date,*DM*: distributed manufacturing, F_m: flowshop with m machines, FF_s: Flexible flowshop with s stages, *fmls*: part families: M families with n_h jobs are assigned to a flowshop cell to be processed; **LBD**: Lower-bound development, **GA**: Genetic algorithm, **TS**: Tabu search, **PSO**: Particle swarm optimization, **SA**: Simulated annealing, **VNS**: Variable neighborhood search, **IG**: Iterated greedy, **RMSA**: Revised multi-start simulated annealing, **DDE**: Discrete differential evolution, **B&P**: Branch-and-price, **HHS**: Hybrid harmony search, **BBO**: Biogeography-based optimization, **CCEA**: Cooperative co-evolutionary algorithm, **MA**: Memetic algorithm, **MM**: Mathematical model, **MOGA**: Multi objective genetic algorithm, **NSGA-II**: Non-dominated sorting genetic algorithm; $C_{\max}$: Makespan, TCT: Total completion time, TWT: Total weighted tardiness, TWET: Total weighted earliness and tardiness, TT: Total tardiness, TWCT: Total weighted completion time.

reinitialization methods are also designed to sidestep search stagnation and stimulate search vitality.
- To enhance the local exploitation of EEDA and refine the obtained solutions, three problem-specific local search strategies are presented, where the family-based and job-based swap or insertion operators are properly utilized to further reduce the makespan.
- To verify the effectiveness of the MILP model, the Gurobi solver is employed and validated on small and medium sized instances. Furthermore, the proposed knowledge-enhanced EDA is compared with several state-of-the-art algorithms on instances of different sizes and the statistical results demonstrate the superiority of EEDA for DNFGSP_SDSTs.

The subsequent sections are organized as follows. In Section 3, we provide a detailed description of the MILP formulation for DNFGSP_SDSTs and explain speed-up evaluation methods. Moreover, we provide an in-depth analysis of problem properties. Section 4 describes the knowledge-enhanced EDA, including coding schemes, partitioned probability model, family-based updating mechanism, sampling strategy, and perturbation and reinitialization. Furthermore, problem-specific local search strategies for DNFGSP_SDSTs are introduced. Section 5 is dedicated to extensive experiments and comprehensive comparisons. In Section 6, we discuss some findings, research limitations, and make suggestions for future research.

2. Literature review

In this study, we propose an effective enhanced EDA with problem-specific knowledge to tackle the DNFGSP_SDSTs. It is worth noting that, to the best of our knowledge, research on this problem has rarely been reported in the literature. Thus, we conduct a comprehensive review of the related works on FGSPs, DFSPs, and the applications of EDAs to scheduling problems.

2.1. Flowshop group scheduling problems

FGSPs are widely recognized as considerable challenges in flexible manufacturing systems, and in recent years, studies on FGSPs have garnered increasing attention due to their potential to shorten manufacturing cycles, reduce inventory levels, and decrease production costs [9]. The core concept behind group scheduling aims to enhance efficiency by grouping similar jobs into families, thereby reducing adaptation and preparation times. Radharamanan [8] first introduced this concept, providing a heuristic method for determining both the optimal group and job order in a batch-type production process with a functional layout. It is further confirmed that enterprises implementing CMS receive multiple benefits, including reduced setup and process times, work-in-process inventories, simplified flow of jobs, and improved production efficiency [26]. To ensure that machines are running on time, machine downtime is reduced, and delivery dates are met, it is inevitable that setup times need to be emphasized [10,11]. Thus, the study scope has been extended by further considering the SDSTs as a key factor in optimizing the makespan of FGSPs [50]. For minimizing the makespan of FGSP_SDSTs, Ying et al. [31] proposed an effective SA approach that showed effectiveness in yielding high-quality results. They recommended the utilization of non-permutation schedules to enhance efficiency within reasonable computational costs. Expanding the scope to consider other criteria for FGSP_SDSTs and its variants, numerous endeavors have emerged. Karimi et al. [25] dealt with flexible FGSP_SDSTs which considers minimizing the criteria of both makepan and TWT by implementing a multi-phase bi-objective GA. Yuan et al. [21] investigated FGSP_SDSTs further taking into account the round-trip transportation time between machines and proposed a co-evolutionary DDE algorithm (CDDEA) for minimizing the makespan.

Subsequently, Naderi et al. [40] proposed a metaheuristic hybridizing genetic and simulated annealing (GSA) algorithm for minimizing the total completion time, while Hajinejad et al. [30] introduced a hybrid PSO (HPSO) algorithm for optimizing FGSP_SDSTs with the goal of minimizing TCT. Both GSA and HPSO outperformed the hybrid ACO (HACO) algorithm proposed by Salmasi et al. [33]. Li et al. [45] presented a hybrid harmony search (HHS) for solving the flowline manufacturing cell scheduling problem (FMCSP) with SDSTs to minimize both total tardiness and mean total flowtime. Further contributions by Costa et al. [26,27] introduced a hybrid GA (HGA) and a parallel self-adaptive genetic algorithm (PSAGA), both of which outperformed GSA [40] and HPSO [30], among them, PSAGA achieved the best results, delivering the best results in terms of optimizing the makespan of FGSP_SDSTs. In addition, the practical industrial applications of CMS, spanning diverse sectors like hot rolling, chemical or pharmaceutical processes, and food industries, add additional constraints such as no-wait constraints, escalating the complexity of group scheduling challenges. Considering FGSP_SDSTs with no-wait constraints, Ying et al. [16] proposed three heuristics, *i.e.*, GA-based, SA-based, and IG-based heuristics, for minimizing the makespan of NFGSP_SDSTs. Experimental results revealed that although the SA-based heuristic performed the best among the three, there was no significant difference between them. To find more efficient algorithms for optimizing the TWET of NFGSP_SDSTs, Arabameri and Salmasi [51] developed several HIOAs based on TS and PSO. The results indicated a slight superiority of PSO-based algorithms, especially when handling medium- and large-sized problems. For minimizing the TCT of NFGSP_SDSTs, Behjat and Salmasi [17] proposed several versions of PSO- and VNS-based algorithms, incorporating multiple neighborhood search structures for enhanced problem-solving capabilities. Recently, Cheng et al. [19] also proposed two types of HIOAs, *i.e.*, a revised multi-start SA (RMSA) and a local search-based variant ($RMSA_{LS}$), both of which yielded better solutions than PSO- and VNS-based algorithms [17]. Remarkably, the efficacy of $RMSA_{LS}$ was better than that of RMSA, particularly in real-life applications demanding high-quality solutions for industrial scenarios. According to above review, critical observation shows that the study of NFGSP_SDSTs is still scarce. This gap is motivated by the need to improve productivity, reduce costs and optimize production cycles, emphasizing the significance of investigating FGSPs to tackle the multifaceted challenges posed by complex constraints and setup time dependencies inherent in CMS.

2.2. Distributed flowshop scheduling problems

DFGSPs share similar characteristics with DFSPs, with the latter being more extensively explored, and therefore providing valuable insights into the study of the former, which are of great interest. As for the minimization of the makespan for DFSP, Naderi and Ruiz [7] presented a comprehensive characterization of the problem through the formulation of six MILP models. They provided two factory allocation rules, *i.e.*, NR_1 and NR_2, and then devised 14 heuristics incorporating several VND-based local search methods, resulting in excellent effectiveness. After this pioneering effort, Wang et al. [52] also introduced an earliest completion factory (ECF) allocation rule and proposed an effective EDA to address this challenge. Considering no-wait constraints, Shao et al. [53] attempted to address the DNFSP with the goal of minimizing makespan by developing an IG-based heuristic. Comparing the efficacy of EDA [52] and IG [53] with several heuristics [7], both EDA and IG proved to be more effective, especially for large-scale problems. Notably, the effectiveness of the IG based on VND surpassed that of the EDA and other IG-based versions proposed by Shao et al. [53]. For solving DFGSP_SDSTs with the aim of minimizing makespan, Pan et al. [3] proposed a CCEA and Wang et al. [24] presented a two-stage IG (TIG_{v1}). Both CCEA and TIG_{v1} outperformed IG, and TIG_{v1} achieved the best results among them. In addition, Wang et al. [54] proposed a TIG_{v2} to minimize the TT in DFGSP_SDST. Unlike the reconstruction mechanism of TIG_{v1}, which examined all extracted families at all possible positions, that of TIG_{v2} selectively tested potential positions. Li et al. [55] further introduced the destruction-construction mechanism inherent in IGs into the framework of ABC algorithm to address DFSP with peak power consumption, which provided a new perspective on the design of improved ABC (IABC). More recently, He et al. [56] provided a novel insight by developing a greedy CCEA with problem-specific knowledge to solve energy-efficient FGSP with the criteria of minimizing the makespan, total flow time (TFT), and total energy consumption (TEC), simultaneously. Furthermore, Niu et al. [57] designed a two-stage CCEA (TS-CCEA) to minimize both makespan and TEC to address the energy-efficient distributed group blocking flowshop problem with setup carryover in precast systems. Compared with existing multi-objective evolutionary algorithms, TS-CCEA demonstrated superior search performance and stability. Consequently, these research efforts collectively contribute to the development of innovative methodologies and efficient scheduling strategies to address the complex challenges posed by DFGSP. Considering their potential applications in distributed manufacturing systems, the study of DFGSPs is of both academic and engineering significance.

2.3. Estimation of distribution algorithms

As effective and efficient learning-based paradigms within swarm intelligence and evolutionary computation, the recent emergence of EDAs has made crucial contributions to mitigating the building block breakage caused by classical crossover and mutation operators in GAs [58]. The crux of what makes EDAs work is that they can estimate and exploit promising patterns extracted from superior solutions. By formulating probability models and capturing crucial characteristics, EDAs accurately describe and reveal the distribution of building blocks, thus facilitating the detailed analysis and utilization of implicit relations in block structures. The excellent learning ability and good generalization capability of EDAs have triggered substantial studies, suggesting their promising prospects for solving various optimization problems [59]. Notably, recent years have witnessed their successful applications in addressing a variety of different scheduling problems. Jarbouri et al. [60] made the first attempt to address the PFSP by using a two-dimensional (2D) probability model-based EDA that aims to minimize the total flowtime. This 2D probability model focuses on the priority of job order and the presence of similar job blocks in selected superior solutions. Chen et al. [61] further investigated the convergence behavior of EDAs and provided guidelines for designing EDAs that take into account the effects of both intensification and diversification. Their adaptive EA/G demonstrated efficacy in minimizing earliness and tardiness for solving single machine scheduling problems. They claimed that EDAs can retain good structural features and converge faster than that of using genetic operators. Wang et al. [62] put forward an EDA-based memetic algorithm (EDAMA) to solve the distributed assembly permutation flow-shop scheduling problem (DAPFSP). In EDAMA, a problem-specific 2D probability model was built to describe the distribution of superior solutions in the search space, and EDA-based global exploration and critical-path-based local exploitation were integrated into the framework of EDAMA. Qian et al. [63] introduced a copula-based hybrid EDA (CHEDA) to solve the reentrant PFSP. In this approach, the copula theory was utilized to build a 2D probability model based on the joint distribution function to extract features from elite solutions. Shao et al. [14] proposed a pareto-based EDA (PEDA) for multi-objective DNFSP with SDSTs aimed at minimizing the criteria of makespan and total weight tardiness. In PEDA, three 2D probability models were developed to capture the characteristics of jobs in different scenarios. Recently, Zhao et al. [64] developed an EDA-based hyper--heuristic (EDA-HH) for minimizing the makespan of distributed assembly mixed no-idle PFSP (DAMNIPFSP). The EDA-HH utilized EDA as a high-level strategy, adjusting low-level heuristics (LLHs) with a 2D probability model. This 2D probability model was used to record

inherent information in the sequences of LLHs, indicating the potential of EDAs in the field of hyper heuristics. Clearly, for EDAs, probability models are ideal carriers for capturing ordinal relations and learning block structures, providing the potential for significant breakthroughs in addressing the bottlenecks faced by traditional HIOAs. Most of the existing studies mainly focus on adopting 2D probability models to learn pattern information. Due to the inherent limitations in the model structure, they only accumulate the frequency of blocks, rather than accurately capturing the distribution of blocks implicit in sequential relationships. Recent efforts have been devoted to the development of more efficient learning-based paradigms and the design of effective probability models to enhance the efficacy of EDAs and extend their application scope in addressing the challenges of complex scheduling problems. Zhang et al. [37] pioneered the adoption of the matrix cube to learn information about blocks and their positions. Building on this innovation, they designed a multidimensional probability model with a specific sampling strategy to exactly estimate the distribution trend of promising patterns and guide the global search for seeking superior solutions. Experimental results showed that matrix-cube-based EDA (MCEDA) can yield favorable results with significant advantages. Following this pioneering work, MCEDA has been successfully applied to solve some shop scheduling problems, such as the EE_DAPFSP [38], the BFSP_SDSTs [39], and the NFSSP_SDSTs_RDs [20], with remarkable results. Motivated by these advancements, the above insights inspire this study to develop a knowledge-based enhanced estimation of distribution algorithm (EEDA) to address the DNFGSP_SDSTs, which has not been investigated to the best of the authors' knowledge.

3. Problem statement

3.1. Distributed no-wait flowshop sequence-dependent group scheduling problems

The DNFGSP_SDSTs can be briefly described as follows. There is a set of F identical cellulars, each with a flowshop layout consisting of m machines. A set of S families is considered, where the family set is defined as $\Gamma = \{\Gamma_1, \Gamma_2, ..., \Gamma_S\}$ and each one includes n_h jobs. Each family can be processed at any cellular and can be assigned to only one cellular. All jobs from the same family must be processed continuously without interruption and preemption. There is no wait time between successive operations of the same job. The sequence-dependent family setup time $s_{h,h',j}$ are considered when the changeover from family Γ_h to the family $\Gamma_{h'}$ takes place on the machine M_j. If Γ_h is the first family on M_j, the initial setup time $s_{0,h,j}$ is required. Suppose that all machines are available at time 0. Each machine can process at most one operation for each job at a time, and each job can be processed at most on one machine. (Tables 2 and 3)

3.1.1. MILP model of DNFGSP_SDSTs

$$\min(C_{\max}(\boldsymbol{\pi})), \tag{1}$$

$$C_{\max}(\boldsymbol{\pi}) = \max\{C_{i,m}\}, \forall i, \tag{2}$$

$$\sum_{i=0, i\neq i'}^{n} x_{i,i'} = 1, \forall i', \tag{3}$$

$$\sum_{i'=0, i'\neq i}^{n} x_{i,i'} = 1, \forall i, \tag{4}$$

$$\sum_{i=1}^{n} x_{i,0} = \begin{cases} F, F \leq S \\ S, F > S \end{cases}, \tag{5}$$

Table 2
Notations used in the model of DNFGSP_SDSTs.

Indices	
i, i'	Index for jobs, $i, i' = 1, 2, ..., n$
j, j'	Index for machines, $j, j' = 1, 2, ..., m$
h, h'	Index for families, $h, h' = 1, 2, ..., S$
f, f'	Index for cellulars, $f, f' = 1, 2, ..., F$
Parameters	
n	Total number of jobs
m	Total number of machines
F	Total number of cellulars
S	Total number of families
$p_{i,j}$	The processing time of $O_{i,j}$
$s_{h',h,j}$	The sequence-dependent setup time between Γ_{h-1} and Γ_h on M_j
n_h	Total number of jobs in Γ_h, $\sum_{h=1}^{S} n_h = n$
n^f	Total number of jobs assigned to cellular f, $\sum_{f=1}^{F} n^f = n$
S_f	Total number of families assigned to cellular f, $\sum_{f=1}^{F} S_f = S$
z_h	Total number of jobs before Γ_h, $z_h = \sum_{h=0}^{h} n_{h'}$
$\boldsymbol{\pi}$	A feasible scheduling solution
$C_{i,j}$	The completion time of $O_{i,j}$
X	Sufficiently large positive number
Sets	
J	Set of jobs, *i.e.*, $J = \{J_1, J_2, ..., J_n\}$
M	Set of machines, *i.e.*, $M = \{M_1, M_2, ..., M_m\}$, $\lvert M_j\rvert \geq 2$
Γ	Set of families, *i.e.*, $\Gamma = \{\Gamma_1, \Gamma_2, ..., \Gamma_S\}$
U	Set of cellulars, *i.e.*, $U = \{U_1, U_2, ..., U_F\}$
O	Set of operations, *i.e.*, $O = \{O_{i,1}, O_{i,2}, ..., O_{i,m}\}$
Ψ	Subset of J with at least two jobs
Decision Variables	
$x_{i,i'}$	If J_i is an immediate successor of J_i, then $x_{i,i'} = 1$, otherwise $x_{i,i'} = 0$
$y_{h',h}$	If Γ_h is an immediate successor of $\Gamma_{h'}$, then $y_{h',h} = 1$, otherwise $y_{h',h} = 0$
$I_{i,k}$	If J_i is the kth job in solution $\boldsymbol{\pi}$, then $I_{i,k} = 1$, otherwise $I_{i,k} = 0$

Table 3
Notations used in the calculation of DNFGSP_SDSTs.

Notation	Description
$ML_{h',h,j}$	The completion time difference between the first job of Γ_h and the family $\Gamma_{h'}$ on M_j
$D_{h,i,i+1}$	The completion time difference between J_i and J_{i+1} in Γ_h
sp_i	The total processing time of J_i on all machines, $sp_i = \sum_{j=1}^{m} p_{i,j}$
$C^f_{\max}$	The makespan of cellular f
T_h	The completion time difference between Γ_h and its previous family
Γ^f	Set of families in cellular f, $\Gamma^f = \{\Gamma^f_1, \Gamma^f_2, ..., \Gamma^f_{S_f}\}$
$\Gamma_h(i)$	The ith job in the job sequence $\boldsymbol{\pi}^F$ of Γ_h

$$\sum_{i=1}^{n} x_{0,i'} = \begin{cases} F, F \leq S \\ S, F > S \end{cases}, \tag{6}$$

$$\sum_{J_i \in \Psi} \sum_{J_{i'} \in \Psi} x_{i,i'} \leq |\Psi| - 1, \forall \Psi \subseteq J, i \neq i', \tag{7}$$

$$\sum_{i=0}^{z_{h-1}} \sum_{i'=z_{h-1}+1}^{z_h} x_{i,i'} + \sum_{i=z_h+1}^{z_S} \sum_{i'=z_{h-1}+1}^{z_h} x_{i,i'} = 1, \forall h, \tag{8}$$

$$y_{h',h} = \begin{cases} \sum_{i'=z_{h'_{-1}}+1}^{z_{h'}} \sum_{i=z_{h-1}+1}^{z_h} x_{i',i}, \forall h, h' \wedge h \neq h' \\ \sum_{i=z_{h-1}+1}^{z_h} x_{0,i}, h' = 0, \forall h \\ \sum_{i'=z_{h'_{-1}}+1}^{z_{h'}} x_{i',0}, \forall h', h = 0 \end{cases}, \tag{9}$$

$$C_{i,j} \geq C_{i,j-1} + p_{i,j}, \forall i,j, \tag{10}$$

$$C_{i,j} \geq C_{i',j} + p_{i,j} + \left(x_{i',i} - 1\right)\cdot X, \forall j, \\ \forall i, i' \in \{z_{h-1}+1, ..., z_h\} \wedge i \neq i', \forall h \tag{11}$$

$$C_{i,j} \geq C_{i',j} + p_{i,j} + s_{h',h,j} + \left(y_{i',i} - 1\right)\cdot X, \forall j, \\ i \in \{z_{h-1}+1, ..., z_h\} i' \in \{z_{h'-1}+1, ..., z_{h'}\}, \forall h, h' \wedge h \neq h' \tag{12}$$

$$C_{i,j} \geq C_{0,j} + s_{0,h,j} + p_{i,j} + \left(y_{0,h} - 1\right)\cdot X, \forall j \\ j = z_{h-1}+1, ..., z_h, \forall h \tag{13}$$

$$C_{i,j} = C_{i,j-1} + \sum_{k=1}^{n_h} I_{k,i}\cdot p_{i,j}, \forall i,j. \tag{14}$$

The objective function is shown in Eq. (1) and can be calculated by Eq. (2). Eq. (3) and Eq. (4) enforce that each position has only one immediate predecessor and successor, respectively. Eq. (5) and Eq. (6) ensure that the dummy job J_0 is an immediate successor and an immediate predecessor of the jobs in set J with $\min(F, S)$ times. Subtours are eliminated by Eq. (7). Eq. (8) ensures the jobs from one family are never split up and combined with jobs from other families. The sequence relationship of two families is determined by the relationship of their jobs, which is considered by Eq. (9). The completion time of $O_{i,j}$ must be larger than or equal to the completion time of its upstream $O_{i,j-1}$ plus the processing time $p_{i,j}$ for two successive operations of J_i, as shown in Eq. (10). Note that $C_{i,0} = 0$ and $C_{0,j} = 0$. The completion time of $O_{i,j}$ for J_i and its immediate predecessor $J_{i'}$ on M_j must not be less than the sum of $p_{i,j}$ and the completion time of $O_{i',j}$, shown in Eq. (11); otherwise, if the two are distinct from one another and come from Γ_h and $\Gamma_{h'}$, $s_{h',h,j}$ must be added to the computation of the completion time of $O_{i,j}$, as enforced by Eq. (12). Eq. (13) determines the initial setup time $s_{0,h,j}$ is taken into consideration for the first family of jobs processed on M_j. Eq. (14) ensures that the different processes of the same job must be continuous, which satisfies the no-wait constraint.

3.1.2. Encoding and decoding schemes

The effective encoding and decoding schemes can reasonably reflect the crucial characteristics and critical constraints for complex and coupled problems [65]. The considered DNFGSP_SDSTs is composed of two subproblems: the cellular assignment for each family and the processing sequence for all jobs. Thus, a two-segment encoding scheme is designed to describe feasible solutions, which is denoted as $\boldsymbol{\pi} = [\boldsymbol{\pi}^C, \boldsymbol{\pi}^F]$. The first segment $\boldsymbol{\pi}^C$ represents the family sequence in cellular, *i.e.*, $\boldsymbol{\pi}^C = [(\Gamma_1^1,...,\Gamma_{S_1}^1),...,(\Gamma_1^F,...,\Gamma_{S_F}^F)]$, and the second segment $\boldsymbol{\pi}^F$ represents the job sequence in family, *i.e.*, $\boldsymbol{\pi}^F = [(\Gamma_1(1),...,\Gamma_1(n_1)),...,(\Gamma_S(1),...,\Gamma_S(n_S))]$. The representation of a feasible solution $\boldsymbol{\pi}$ is shown in Fig. 1. It is clear that Γ_1 and Γ_2 are assigned to U_1 and Γ_3 is allocated to U_2. The job sequence of families Γ_1, Γ_2, and Γ_3 is (J_1,J_2,J_3), (J_4), and (J_5,J_6), respectively.

For the decoding scheme of DNFGSP_SDSTs, $ML_{h',h,j}$ can be calculated by Eq. (15). $D_{h,i,i+1}$ is the difference of the completion time between adjacent jobs, which can be calculated by Eq. (16). $C_{\max}(\boldsymbol{\pi})$ can be calculated by Eq. (17). The Gantt chart of an example with $S = 2$, $n = 4$, and $m = 3$ is provided in Fig. 2.

$$ML_{h',h,j} = \begin{cases} \max\left\{s_{h',h,1} + p_{i,1} - p_{i-1,2}, s_{h',h,2}\right\} + p_{i,2}, j = 2 \\ \max\left\{ML_{h',h,j-1} - p_{i,j}, s_{h',l,i}\right\} + p_{i,j}, j = 3, ..., m \end{cases}, \forall i, \forall h, h', \tag{15}$$

$$D_{h,i,i+1} = \max_{k=1,2,...,m}\left\{\sum_{l=k}^{m}\left(p_{i+1,l} - p_{i,l}\right) + p_{i,k}\right\}, i = 1, 2, ..., n_h - 1, h \\ = 1, 2, ..., S, \tag{16}$$

$$C_{\max}(\boldsymbol{\pi}) = \sum_{h=1}^{S} ML_{h-1,h,m} + \sum_{h=1}^{S}\sum_{i=1}^{n_h-1} D_{h,i,i+1}. \tag{17}$$

3.1.3. Example instance

Considering 4 families, 8 jobs, 3 machines, and 2 cellulars, an example is given in this subsection. Four families are denoted as $\Gamma_1 = (J_1,J_2,J_3)$, $\Gamma_2 = (J_4)$, $\Gamma_3 = (J_5,J_6)$, and $\Gamma_4 = (J_7,J_8)$. Two cellulars are represented as $U_1 = [(J_1,J_2,J_3),(J_4)]$ and $U_2 = [(J_5, J_6), (J_7, J_8)]$, respectively. Thus, the feasible solution can be represented as $\boldsymbol{\pi}_0 = [[(J_1, J_2, J_3), (J_4)], [(J_5, J_6), (J_7, J_8)]]$. The job processing time and family sequence-dependence setup time are given in Tables 4 and 5, respectively. The corresponding Gantt chart of a feasible scheduling solution $\boldsymbol{\pi}_0$ is shown in Fig. 3.

3.2. Analysis of problem properties

To reduce the computational complexity (CC) and enhance efficiency, it is of great importance to analyze the problem properties and design effective speed-up evaluation methods to accelerate the calculation while executing some search operators [20]. To the best of our knowledge, the speed-up evaluation methods devised for DNFGSPs are still scarce. Therefore, in this section, we derive five problem-specific properties as knowledge to aid in the design of five speed-up evaluation methods.

Property 1. For a feasible solution $\boldsymbol{\pi}$, randomly select family Γ_a in cellular f and family Γ_b in cellular $f'(f \neq f'$ and each cellular has m machines) from $\boldsymbol{\pi}^C$. Next, swap the positions of two families Γ_a and Γ_b in $\boldsymbol{\pi}^C$. After performing this swap operator, a new solution $\boldsymbol{\pi}'$ is obtained. The CC of calculating $C_{\max}(\boldsymbol{\pi}')$ can be reduced from $O(n \times m)$ to $O(m)$.

Proof. Assuming that there are S_f and $S_{f'}$ families assigned to cellulars f and f', respectively. According to Eqs. (15)-(17), $C_{\max}^f(\boldsymbol{\pi})$ and $C_{\max}^{f'}(\boldsymbol{\pi})$ can be calculated by Eq. (18). When families Γ_a and Γ_b ($a = 1,...,S_f$, $b = 1,...,S_{f'}$) are chosen from cellulars f and f', respectively, $C_{\max}^f(\boldsymbol{\pi})$ and $C_{\max}^{f'}(\boldsymbol{\pi})$ can be further represented as shown in Eq. (19). After swapping two families Γ_a and Γ_b, $C_{\max}^f(\boldsymbol{\pi}')$ and $C_{\max}^{f'}(\boldsymbol{\pi}')$ can be calculated by Eq. (20). Then, the completion time differences $\triangle t_f$ and $\triangle t_{f'}$ in cellulars f and f' can be calculated by using Eq. (21). As shown in Eqs. (19)-(20), $\sum_{h=1}^{a-1} ML_{h-1,h,m}$, $\sum_{h=1}^{a-1}\sum_{i=1}^{n_h-1} D_{h,i,i+1}$ in front of Γ_a and $\sum_{h=a+2}^{S_f} ML_{h-1,h,m}$, $\sum_{h=a+1}^{S_f}\sum_{i=1}^{n_h-1} D_{h,i,i+1}$ behind Γ_a in cellular f are not changed after swapping families Γ_a and Γ_b. Thus, $C_{\max}^f(\boldsymbol{\pi}')$ can be calculated as Eq. (21), and the same derivations can be applied to $C_{\max}^{f'}(\boldsymbol{\pi}')$. Based on Eqs. (19)-(22), the CC of calculating $C_{\max}(\boldsymbol{\pi}')$ can be reduced from $O(n \times m)$ to $O(m)$.

$$\begin{cases} C_{\max}^f(\boldsymbol{\pi}) = \sum_{h=1}^{S_f} ML_{h-1,h,m} + \sum_{h=1}^{S_f}\sum_{i=1}^{n_h-1} D_{h,i,i+1} \\ C_{\max}^{f'}(\boldsymbol{\pi}) = \sum_{h=1}^{S_{f'}} ML_{h-1,h,m} + \sum_{h=1}^{S_{f'}}\sum_{i=1}^{n_h-1} D_{h,i,i+1} \end{cases}, \tag{18}$$

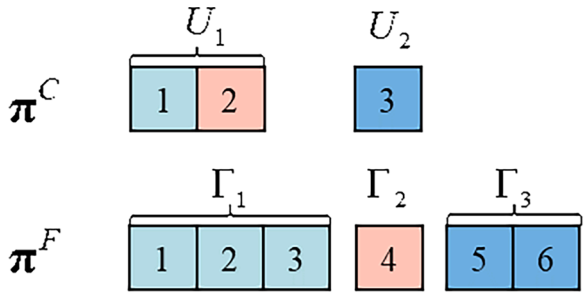


Fig. 1. The representation of a feasible solution $\boldsymbol{\pi}$.

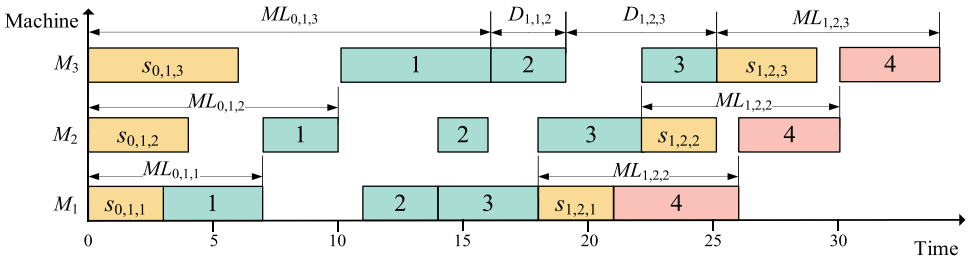


Fig. 2. Gantt chart of an example.

Table 4
Processing time of jobs on machines.

Family	*Job*	*Machine* 1	*Machine* 2	*Machine* 3
1	1	4	3	6
	2	3	2	3
	3	4	4	3
2	4	5	4	4
3	5	3	4	3
	6	2	3	3
4	7	3	4	2
	8	2	4	2

Table 5
Sequence-dependence setup time of families on machines.

	Machine 1				*Machine* 2				*Machine* 3			
Family	1	2	3	4	1	2	3	4	1	2	3	4
1	3	2	4	4	4	2	3	5	5	4	4	3
2	4	4	5	3	4	3	3	3	3	5	4	3
3	2	4	5	2	3	5	4	2	2	3	5	4
4	3	2	5	4	3	4	3	3	3	2	4	3

$$\begin{cases} C^{f}_{\max}(\boldsymbol{\pi}) = \sum_{h=1}^{a-1} ML_{h-1,h,m} + \sum_{h=a+2}^{S_f} ML_{h-1,h,m} + ML_{a-1,a,m} + ML_{a,a+1,m} + \\ \quad \sum_{h=1}^{a} \sum_{i=1}^{n_h-1} D_{h,i,i+1} + \sum_{h=a+1}^{S_f} \sum_{i=1}^{n_h-1} D_{h,i,i+1} + \sum_{i=1}^{n_a-1} D_{a,i,i+1} \\ C^{f'}_{\max}(\boldsymbol{\pi}) = \sum_{h=1}^{b-1} ML_{h-1,h,m} + \sum_{h=b+2}^{S_{f'}} ML_{h-1,h,m} + ML_{b-1,b,m} + ML_{b,b+1,m} + \\ \quad \sum_{h=1}^{b} \sum_{i=1}^{n_h-1} D_{h,i,i+1} + \sum_{h=b+1}^{S_{f'}} \sum_{i=1}^{n_h-1} D_{h,i,i+1} + \sum_{i=1}^{n_b-1} D_{b,i,i+1} \end{cases}, \tag{19}$$

$$\begin{cases} C^{f}_{\max}(\boldsymbol{\pi}') = \sum_{h=1}^{a-1} ML_{h-1,h,m} + \sum_{h=a+2}^{S_f} ML_{h-1,h,m} + ML_{a-1,b,m} + ML_{b,a+1,m} + \\ \quad \sum_{h=1}^{a-1} \sum_{i=1}^{n_h-1} D_{h,i,i+1} + \sum_{h=a+1}^{S_f} \sum_{i=1}^{n_h-1} D_{h,i,i+1} + \sum_{i=1}^{n_b-1} D_{b,i,i+1} \\ C^{f'}_{\max}(\boldsymbol{\pi}') = \sum_{h=1}^{b-1} ML_{h-1,h,m} + \sum_{h=b+2}^{S_{f'}} ML_{h-1,h,m} + ML_{b-1,a,m} + ML_{a,b+1,m} + \\ \quad \sum_{h=1}^{b-1} \sum_{i=1}^{n_h-1} D_{h,i,i+1} + \sum_{h=b+1}^{S_{f'}} \sum_{i=1}^{n_h-1} D_{h,i,i+1} + \sum_{i=1}^{n_a-1} D_{a,i,i+1} \end{cases}, \tag{20}$$

$$\begin{cases} \Delta t_f = C^{f}_{\max}(\boldsymbol{\pi}') - C^{f}_{\max}(\boldsymbol{\pi}) = ML_{a-1,b,m} + ML_{b,a+1,m} + \sum_{i=1}^{n_b-1} D_{b,i,i+1} - \\ \quad ML_{a-1,a,m} - ML_{a,a+1,m} - \sum_{i=1}^{n_a-1} D_{a,i,i+1} \\ \Delta t_{f'} = C^{f'}_{\max}(\boldsymbol{\pi}') - C^{f'}_{\max}(\boldsymbol{\pi}) = ML_{b-1,a,m} + ML_{a,b+1,m} + \sum_{i=1}^{n_a-1} D_{a,i,i+1} - \\ \quad ML_{b-1,b,m} - ML_{b,b+1,m} - \sum_{i=1}^{n_b-1} D_{b,i,i+1} \end{cases}, \tag{21}$$

$$C_{\max}(\boldsymbol{\pi}') = \max\left(C^{f}_{\max}(\boldsymbol{\pi}) + \Delta t_f, C^{f'}_{\max}(\boldsymbol{\pi}) + \Delta t_{f'}\right). \tag{22}$$

For ease of understanding, an intuitive example is provided in Fig. 4. As illustrated in Fig. 4(*a*), families Γ_3 and Γ_2 are assigned to cellular f_2 and f_1, respectively. From Fig. 4(*b*), after swapping the positions of Γ_3 and Γ_2, it can be concluded that the processing time of Γ_1 before Γ_3 in cellular 1 is not affected, while only $ML_{3,4,3}$ is changed to $ML_{2,4,3}$ in the calculation of processing time after Γ_2 in cellular 2.

Property 2. For a feasible solution $\boldsymbol{\pi}$, randomly select family Γ_a in

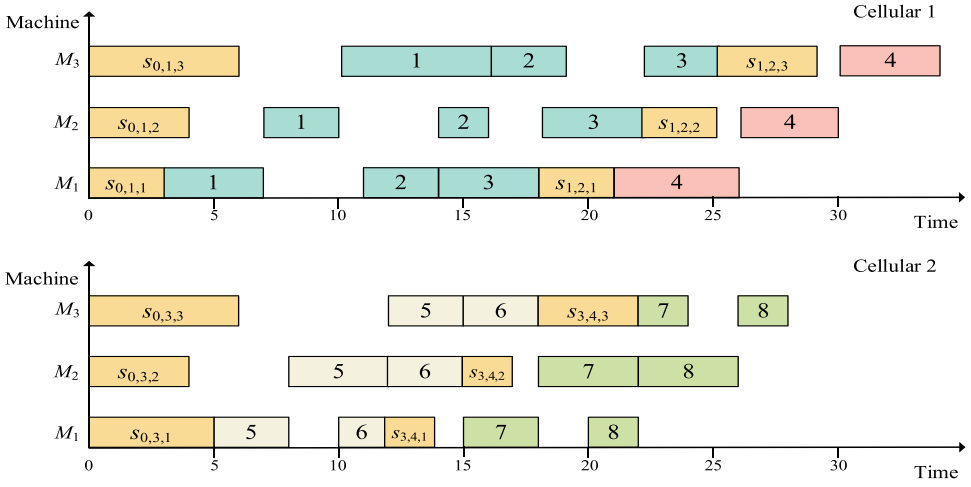
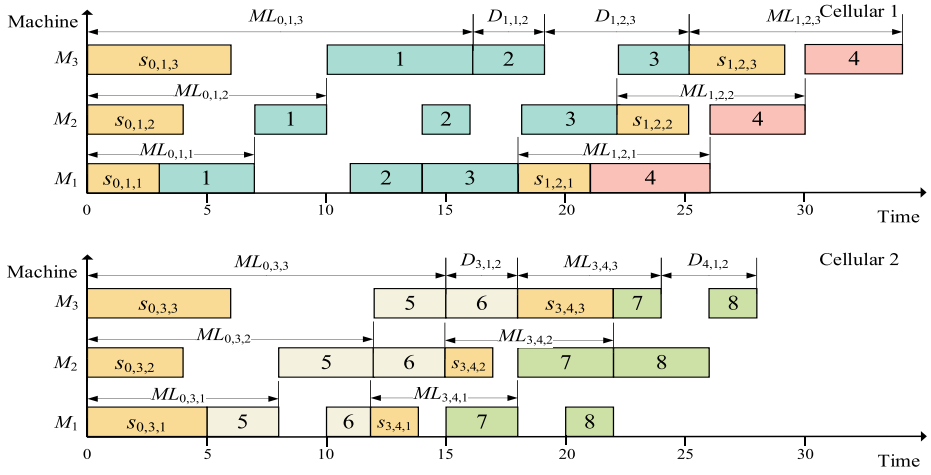
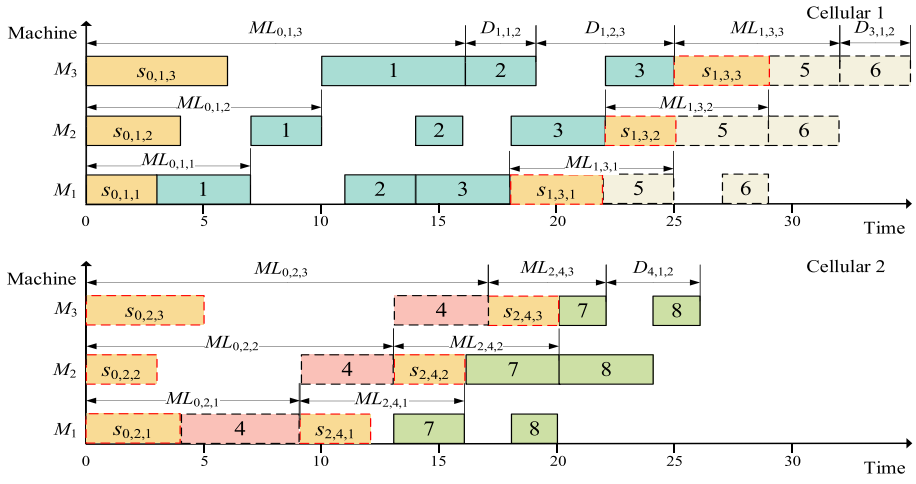


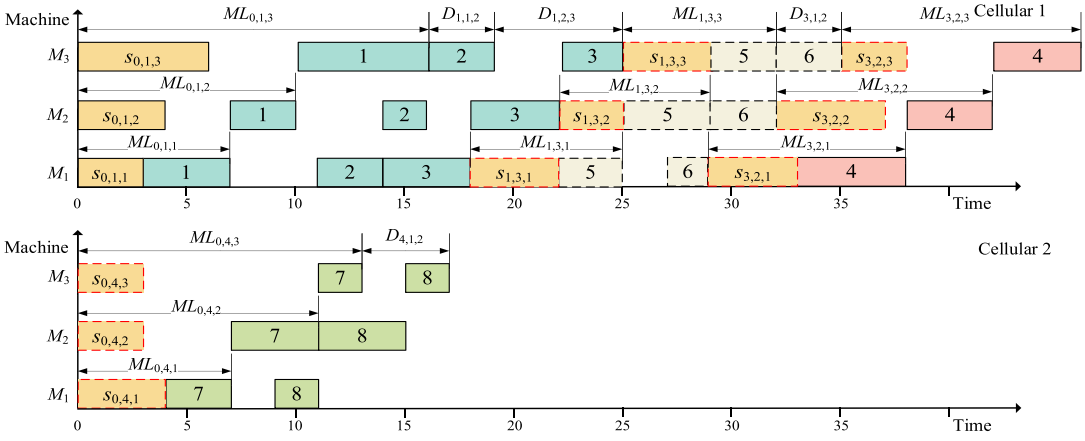
Fig. 3. Gantt chart of a feasible scheduling solution $\boldsymbol{\pi}_0$.



(*a*) Gantt chart of $\boldsymbol{\pi}$.

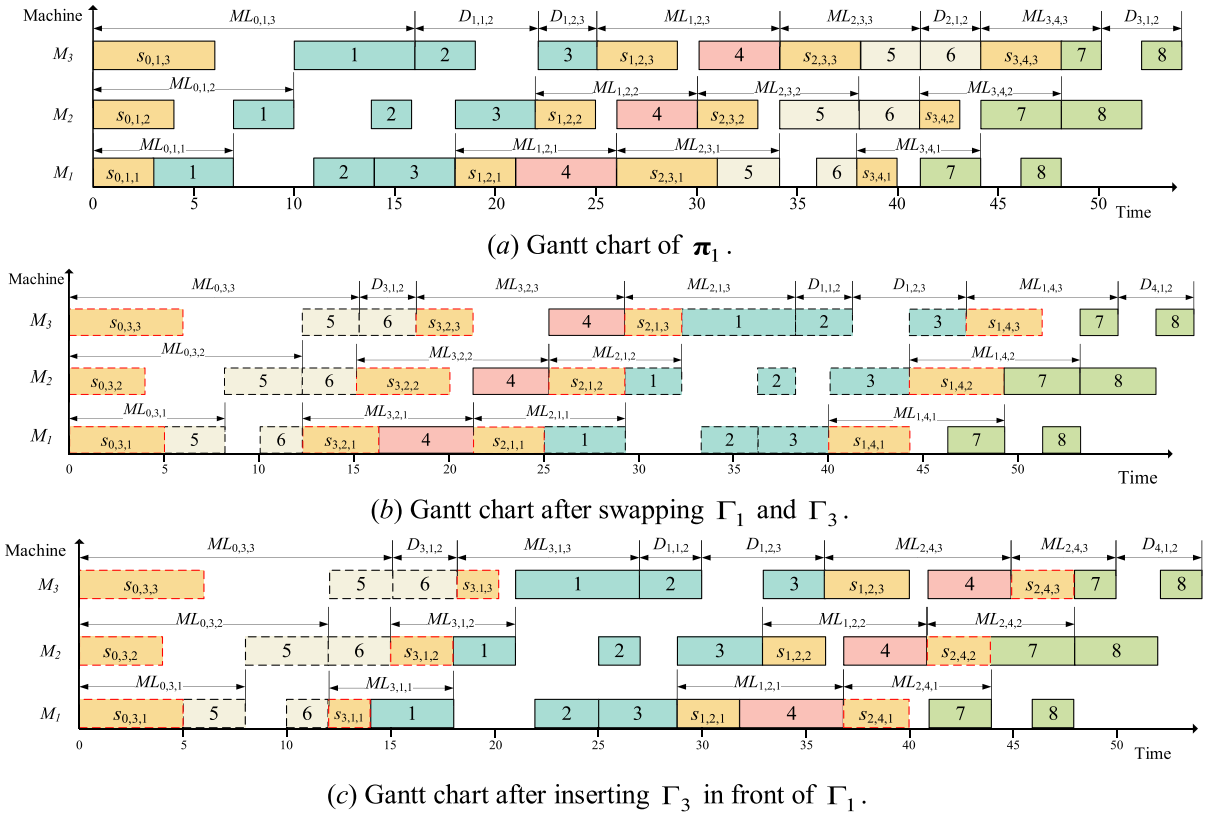


(*b*) Gantt chart after swapping Γ_2 and Γ_3.



(*c*) Gantt chart after inserting Γ_3.

Fig. 4. Illustration of **Property 1** and **Property 2**.



(a) Gantt chart of $\boldsymbol{\pi}_1$.

(b) Gantt chart after swapping Γ_1 and Γ_3.

(c) Gantt chart after inserting Γ_3 in front of Γ_1.

Fig. 5. Illustration of **Property 3** and **Property 4**.

cellular f and family Γ_b in cellular $f'(f \neq f'$ and each cellular has M machines) from $\boldsymbol{\pi}^C$. Next, insert Γ_b in front of Γ_a in $\boldsymbol{\pi}^C$. After performing this insert operator, a new solution $\boldsymbol{\pi}'$ is obtained. The CC of calculating $C_{\max}(\boldsymbol{\pi}')$ can be reduced from $O(n \times m)$ to $O(m)$.

Proof. Assuming that S_f and $S_{f'}$ families are assigned to cellulars f and f', respectively. According to Eqs. (15)-(17), $C^f_{\max}(\boldsymbol{\pi})$ and $C^{f'}_{\max}(\boldsymbol{\pi})$ can be calculated by Eq. (23). When families Γ_a and Γ_b ($a = 1,\ldots,S_f$, $b = 1,\ldots,S_{f'}$) are chosen from cellulars f and f', respectively, $C^f_{\max}(\boldsymbol{\pi})$ and $C^{f'}_{\max}(\boldsymbol{\pi})$ can be further represented as Eq. (24). After inserting family Γ_b in front of Γ_a in $\boldsymbol{\pi}^C$, $C^f_{\max}(\boldsymbol{\pi}')$ and $C^{f'}_{\max}(\boldsymbol{\pi}')$ can be calculated by Eq. (25). The completion time differences Δt_f and $\Delta t_{f'}$ in cellulars f and f' are obtained by Eq. (26). As shown in Eqs. (24)-(25), $\sum_{h=1}^{a-1} ML_{h-1,h,m}$ in front of Γ_a, $\sum_{h=a+1}^{S_f} ML_{h-1,h,m}$ behind Γ_a, and $\sum_{h=1}^{S_f}\sum_{i=1}^{n_h-1} D_{h,i,i+1}$ in cellular f are not changed after inserting Γ_b. While $\sum_{h=1}^{b-1} ML_{h-1,h,m}$, $\sum_{h=1}^{b}\sum_{i=1}^{n_h-1} D_{h,i,i+1}$ in front of Γ_b and $\sum_{h=b+2}^{S_f} ML_{h-1,h,m}$, $\sum_{h=b+1}^{S_f}\sum_{i=1}^{n_h-1} D_{h,i,i+1}$ behind Γ_b in cellular are also not changed after extracting Γ_b. $C^f_{\max}(\boldsymbol{\pi}')$ and $C^{f'}_{\max}(\boldsymbol{\pi}')$ can be calculated as Eq. (26). Thus, the CC of calculating $C_{\max}(\boldsymbol{\pi}')$ can be reduced from $O(n \times m)$ to $O(m)$ based on Eqs. (24)-(27).

$$\begin{cases} C^f_{\max}(\boldsymbol{\pi}) = \sum_{h=1}^{S_f} ML_{h-1,h,m} + \sum_{h=1}^{S_f}\sum_{i=1}^{n_h-1} D_{h,i,i+1} \\ C^{f'}_{\max}(\boldsymbol{\pi}) = \sum_{h=1}^{S_{f'}} ML_{h-1,h,m} + \sum_{h=1}^{S_{f'}}\sum_{i=1}^{n_h-1} D_{h,i,i+1} \end{cases}, \tag{23}$$

$$\begin{cases} C^f_{\max}(\boldsymbol{\pi}) = \sum_{h=1}^{a-1} ML_{h-1,h,m} + \sum_{h=a+1}^{S_f} ML_{h-1,h,m} + ML_{a-1,a,m} + \sum_{h=1}^{S_f}\sum_{i=1}^{n_h-1} D_{h,i,i+1} \\ C^{f'}_{\max}(\boldsymbol{\pi}) = \sum_{h=1}^{b-1} ML_{h-1,h,m} + \sum_{h=b+2}^{S_f} ML_{h-1,h,m} + ML_{b-1,b,m} + ML_{b,b+1,m} + \\ \quad \sum_{h=1}^{b}\sum_{i=1}^{n_h-1} D_{h,i,i+1} + \sum_{h=b+1}^{S_f}\sum_{i=1}^{n_h-1} D_{h,i,i+1} + \sum_{i=1}^{n_b-1} D_{b,i,i+1} \end{cases}, \tag{24}$$

$$\begin{cases} C^f_{\max}(\boldsymbol{\pi}') = \sum_{h=1}^{a-1} ML_{h-1,h,m} + \sum_{h=a+1}^{S_f} ML_{h-1,h,m} + ML_{a-1,b,m} + ML_{b,a,m} + \\ \quad \sum_{h=1}^{S_f}\sum_{i=1}^{n_h-1} D_{h,i,i+1} + \sum_{i=1}^{n_b-1} D_{b,i,i+1} \\ C^{f'}_{\max}(\boldsymbol{\pi}') = \sum_{h=1}^{b-1} ML_{h-1,h,m} + \sum_{h=b+2}^{S_f} ML_{h-1,h,m} + ML_{b-1,b+1,m} + \\ \quad \sum_{h=1}^{b}\sum_{i=1}^{n_h-1} D_{h,i,i+1} + \sum_{h=b+1}^{S_f}\sum_{i=1}^{n_h-1} D_{h,i,i+1} \end{cases}, \tag{25}$$

$$\begin{cases} \triangle t_f = C^f_{\max}(\boldsymbol{\pi}^{'}) - C^f_{\max}(\boldsymbol{\pi}) = ML_{a-1,b,m} + ML_{b,a,m} + \sum_{i=1}^{n_b-1} D_{b,i,i+1} - ML_{a-1,a,m} \\ \triangle t_{f^{'}} = C^{f^{'}}_{\max}(\boldsymbol{\pi}^{'}) - C^{f^{'}}_{\max}(\boldsymbol{\pi}) = ML_{b-1,b+1,m} - ML_{b-1,b,m} - ML_{b,b+1,m} - \sum_{i=1}^{n_b-1} D_{b,i,i+1} \end{cases}, \tag{26}$$

$$C_{\max}(\boldsymbol{\pi}^{'}) = \max\left(C^f_{\max}(\boldsymbol{\pi}) + \triangle t_f, C^{f^{'}}_{\max}(\boldsymbol{\pi}) + \triangle t_{f^{'}}\right). \tag{27}$$

As shown in Fig. 4(c), after extracting Γ_3 and inserting it into the position in front of Γ_2, the processing time before Γ_3 is not affected, and only $ML_{1,2,3}$ changes to $ML_{3,2,3}$ in the calculation of processing time after Γ_3 in cellular 1. Meanwhile, for cellular 2, only $ML_{3,4,3}$ changes to $ML_{0,4,3}$ in the calculation of the processing time after the extracting position of Γ_3.

Property 3. For a feasible solution $\boldsymbol{\pi}$, randomly select two families Γ_a and $\Gamma_b (b > a)$ in cellular f with m machines from $\boldsymbol{\pi}^C$. Next, swap the positions of Γ_a and Γ_b in $\boldsymbol{\pi}^C$. After performing this swap operator, a new solution $\boldsymbol{\pi}^{'}$ is obtained. The CC of calculating $C^f_{\max}(\boldsymbol{\pi}^{'})$ can be reduced from $O(n^f \times m)$ to $O(m)$.

Proof. Assuming that S_f families are assigned to cellular f. According to Eqs. (15)-(17), $C^f_{\max}(\boldsymbol{\pi})$ can be calculated by Eq. (28). When families Γ_a and Γ_b ($a = 1, ..., S_f$, $b = 2, ..., S_f$, $b > a$) are chosen from cellular f, $C^f_{\max}(\boldsymbol{\pi})$ can be further represented as Eq. (29), where the case of $\Gamma_a = \Gamma_{b-1}$ is also considered. After swapping two families Γ_a and Γ_b, $C^f_{\max}(\boldsymbol{\pi}^{'})$ can be calculated by Eq. (30). Then, the completion time differences $\triangle t_f$ in cellular f can be calculated by Eq. (31). As shown in Eqs. (29)-(30), if $\Gamma_a \neq \Gamma_{b-1}$, $\sum_{h=1}^{a-1} ML_{h-1,h,m}$ in front of Γ_a, $\sum_{h=b+2}^{S_f} ML_{h-1,h,m}$ behind Γ_b, and $\sum_{h=a+2}^{b-1} ML_{h-1,h,m}$ between two families in cellular f are not changed after swapping families Γ_a and Γ_b, $C^f_{\max}(\boldsymbol{\pi}^{'})$ can be calculated as Eq. (31) and similar derivations can be applied to $\Gamma_a = \Gamma_{b-1}$. Thus, the CC of calculating $C^f_{\max}(\boldsymbol{\pi}^{'})$ can be reduced from $O(n^f \times m)$ to $O(m)$ by using Eqs. (29)-(32).

$$C^f_{\max}(\boldsymbol{\pi}) = \sum_{h=1}^{S_f} ML_{h-1,h,m} + \sum_{h=1}^{S_f} \sum_{i=1}^{n_h-1} D_{h,i,i+1}, \tag{28}$$

$$\begin{cases} C^f_{\max}(\boldsymbol{\pi}) = \sum_{h=1}^{a-1} ML_{h-1,h,m} + \sum_{h=a+2}^{b-1} ML_{h-1,h,m} + \sum_{h=b+2}^{S_f} ML_{h-1,h,m} + \\ ML_{a-1,a,m} + ML_{a,a+1,m} + ML_{b-1,b,m} + ML_{b,b+1,m} + \sum_{h=1}^{S_f} \sum_{i=1}^{n_h-1} D_{h,i,i+1}, \Gamma_a \neq \Gamma_{b-1} \\ C^f_{\max}(\boldsymbol{\pi}) = \sum_{h=1}^{a-1} ML_{h-1,h,m} + \sum_{h=b+2}^{S_f} ML_{h-1,h,m} + ML_{a,b,m} + \\ ML_{a-1,a,m} + ML_{b,b+1,m} + \sum_{h=1}^{S_f} \sum_{i=1}^{n_h-1} D_{h,i,i+1}, \Gamma_a = \Gamma_{b-1} \end{cases}, \tag{29}$$

$$\begin{cases} C^f_{\max}(\boldsymbol{\pi}^{'}) = \sum_{h=1}^{a-1} ML_{h-1,h,m} + \sum_{h=a+2}^{b-1} ML_{h-1,h,m} + \sum_{h=b+2}^{S_f} ML_{h-1,h,m} + \\ ML_{a-1,b,m} + ML_{b,a+1,m} + ML_{b-1,a,m} + ML_{a,b+1,m} + \sum_{h=1}^{S_f} \sum_{i=1}^{n_h-1} D_{h,i,i+1}, \Gamma_a \neq \Gamma_{b-1} \\ C^f_{\max}(\boldsymbol{\pi}^{'}) = \sum_{k=1}^{a-1} ML_{h-1,h,m} + \sum_{h=b+2}^{S_f} ML_{h-1,h,m} + ML_{a-1,b,m} + \\ ML_{b,a,m} + ML_{a,b+1,m} + \sum_{h=1}^{S_f} \sum_{i=1}^{n_h-1} D_{h,i,i+1}, \Gamma_a = \Gamma_{b-1} \end{cases}, \tag{30}$$

$$\begin{aligned} \triangle t_f &= C^f_{\max}(\boldsymbol{\pi}^{'}) - C^f_{\max}(\boldsymbol{\pi}) \\ &= \begin{cases} ML_{a-1,b,m} + ML_{b,a+1,m} + ML_{b-1,a,m} + ML_{a,b+1,m} - \\ ML_{a-1,a,m} - ML_{a,a+1,m} - ML_{b-1,b,m} - ML_{b,b+1,m}, \Gamma_a \neq \Gamma_{b-1} \\ C^f_{\max}(\boldsymbol{\pi}^{'}) = ML_{a-1,b,m} + ML_{b,a,m} + ML_{a,b+1,m} - \\ ML_{a,b,m} - ML_{a-1,a,m} - ML_{b,b+1,m}, \Gamma_a = \Gamma_{b-1} \end{cases}, \end{aligned} \tag{31}$$

$$C^f_{\max}(\boldsymbol{\pi}^{'}) = C^f_{\max}(\boldsymbol{\pi}) + \triangle t_f. \tag{32}$$

Property 4. For a feasible solution $\boldsymbol{\pi}$, randomly select two families Γ_a and Γ_b in cellular f with m machines. Next, insert Γ_b in front of Γ_a in $\boldsymbol{\pi}^C$. After performing this insertion, a new solution $\boldsymbol{\pi}^{'}$ is obtained. The CC of calculating $C^f_{\max}(\boldsymbol{\pi}^{'})$ can be reduced from $O(n^f \times m)$ to $O(m)$.

Proof. Assuming that S_f families are assigned to cellular f. According to Eqs. (15)-(17), $C^f_{\max}(\boldsymbol{\pi})$ can be calculated by Eq. (33). When families Γ_a and Γ_b ($a = 1, ..., S_f$, $b = 2, ..., S_f$, $b > a$) are chosen from cellular f, $C^f_{\max}(\boldsymbol{\pi})$ can be further represented as Eq. (34). After inserting Γ_b in front of Γ_a, $C^f_{\max}(\boldsymbol{\pi}^{'})$ can be calculated by Eq. (35). Then, the completion time differences $\triangle t_f$ in cellular f can be calculated by using Eq. (36). As shown in Eqs. (34)-(35), $\sum_{h=1}^{a-1} ML_{h-1,h,m}$ in front of Γ_a, $\sum_{h=b+2}^{S_f} ML_{h-1,h,m}$ behind Γ_b, and $\sum_{h=a+1}^{b-1} ML_{h-1,h,m}$ between two families in cellular f are not changed after inserting Γ_b. Next, $C^f_{\max}(\boldsymbol{\pi}^{'})$ can be calculated as Eq. (36).

According to Eqs. (34)-(37), the CC of calculating $C^f_{\max}(\pi')$ can be reduced from $O(n^f \times m)$ to $O(m)$.

$$C^f_{\max}(\pi) = \sum_{h=1}^{S_f} ML_{h-1,h,m} + \sum_{h=1}^{S_f}\sum_{i=1}^{n_h-1} D_{h,i,i+1}, \tag{33}$$

$$C^f_{\max}(\pi) = \sum_{h=1}^{a-1} ML_{h-1,h,m} + \sum_{h=a+1}^{b-1} ML_{h-1,h,m} + \sum_{h=b+2}^{S_f} ML_{h-1,h,m} + ML_{a-1,a,m} + ML_{b-1,b,m} + ML_{b,b+1,m} + \sum_{h=1}^{S_f}\sum_{i=1}^{n_h-1} D_{h,i,i+1}, \tag{34}$$

shown in Eqs. (39)-(40), after inserting $J_{i'}$ in front of J_i, if $i = 1 \wedge i' = 2, \ldots, n_h - 1$, $\sum_{t=i'}^{i-2} D_{h,t,t+1}$ are not changed; if $i = 2, \ldots, n_h - 1 \wedge i' = n_h$, $\sum_{t=i'}^{i-2} D_{h,t,t+1}$, $\sum_{t=1}^{i-2} D_{h,t,t+1}$ are not changed; if $i = 1 \wedge i' = n_h$, $\sum_{t=i'}^{i-2} D_{h,t,t+1}$ are not changed; else, $\sum_{t=1}^{i-2} D_{h,t,t+1}$, $\sum_{t=i'}^{i-2} D_{h,t,t+1}$, $\sum_{t=i'+1}^{n_h-1} D_{h,t,t+1}$ are not changed. Next, T'_h can be calculated as Eq. (42). According to Eqs. (39)-(42), the CC of calculating T'_h can be reduced from $O(n_h \times m)$ to $O(m)$.

$$T_h = ML_{h-1,h,m} + \sum_{t=1}^{n_h-1} D_{h,t,t+1}, \tag{38}$$

$$T_h = \begin{cases} \sum_{t=i}^{i'-2} D_{h,t,t+1} + ML_{h-1,h,m} + D_{h,i'-1,i} + D_{h,i',i'+1}, i = 1 \wedge i' = 2, \ldots, n_h - 1 \\ \sum_{t=i}^{i'-2} D_{h,t,t+1} + \sum_{t=1}^{i-2} D_{h,t,t+1} + D_{h,i-1,i} + D_{h,i'-1,i'} + ML_{h,h+1,m}, i = 2, \ldots, n_h - 1 \wedge i' = n_h \\ \sum_{t=i}^{i'-2} D_{h,t,t+1} + ML_{h-1,h,m} + D_{h,i'-1,i'} + ML_{h,h+1,m}, i = 1 \wedge i' = n_h \\ \sum_{t=i}^{i'-2} D_{h,t,t+1} + \sum_{t=1}^{i-2} D_{h,t,t+1} + \sum_{t=i'+1}^{n_h-1} D_{h,t,t+1} + D_{h,i-1,i} + D_{h,i'-1,i'} + D_{h,i',i'+1}, otherwise \end{cases}, \tag{39}$$

$$T'_h = \begin{cases} ML'_{h-1,h,m} + D_{h,i',i} + D_{h,i'-1,i'+1} + \sum_{t=i}^{i'-2} D_{h,t,t+1}, i = 1 \wedge i' = 2, \ldots, n_h - 1 \\ \sum_{t=i}^{i'-2} D_{h,t,t+1} + \sum_{t=1}^{i-2} D_{h,t,t+1} D_{h,i-1,i'} + D_{h,i',i} + ML'_{h,h+1,m}, i = 1, \ldots, n_h - 1 \wedge i' = n_h \\ \sum_{t=i}^{i'-2} D_{h,t,t+1} + ML'_{h-1,h,m} + D_{h,i',i} + ML'_{h,h+1,m}, i = 1 \wedge i' = n_h \\ \sum_{t=i}^{i'-2} D_{h,t,t+1} + \sum_{t=1}^{i-2} D_{h,t,t+1} + \sum_{t=i'+1}^{n_h-1} D_{h,t,t+1} + D_{h,i-1,i'} + D_{h,i',i} + D_{h,i'-1,i'+1}, otherwise \end{cases}, \tag{40}$$

$$C^f_{\max}(\pi') = \sum_{h=1}^{a-1} ML_{h-1,h,m} + \sum_{h=a+1}^{b-1} ML_{h-1,h,m} + \sum_{h=b+2}^{S_f} ML_{h-1,h,m} + ML_{a-1,b,m} + ML_{b,a,m} + ML_{b-1,b+1,m} + \sum_{h=1}^{S_f}\sum_{i=1}^{n_h-1} D_{h,i,i+1}, \tag{35}$$

$$\Delta t_f = C^f_{\max}(\pi') - C^f_{\max}(\pi) = ML_{a-1,b,m} + ML_{b,a,m} + ML_{b-1,b+1,m} - ML_{a-1,a,m} - ML_{b-1,b,m} - ML_{b,b+1,m}, \tag{36}$$

$$C^f_{\max}(\pi') = C^f_{\max}(\pi) + \Delta t_f. \tag{37}$$

As shown in Fig. 5, a feasible solution $\pi_1 = [(J_1, J_2, J_3), (J_4), (J_5, J_6), (J_7, J_8)]$ is used as an example to explain **Property 3** and **Property 4**. In Fig. 5(*b*), after swapping the positions of Γ_1 and Γ_3, only $ML_{1,2,3}$ changes to $ML_{3,2,3}$ between Γ_1 and Γ_3, and the calculation of processing time after Γ_1 only has changed $ML_{3,4,3}$ to $ML_{1,4,3}$. In Fig. 5(*c*), after inserting family Γ_3, only $ML_{0,1,3}$ changes to $ML_{3,1,3}$ in the calculation of processing time after the family Γ_3, and the calculation of the processing time after the extracting position of Γ_3 only changes $ML_{3,4,3}$ to $ML_{2,4,3}$.

Property 5. For a feasible solution π, randomly select two jobs J_i and $J_{i'}$ from family Γ_h. Next, insert $J_{i'}$ to the position in front of J_i in π^F. After performing such an insert operator, a new π' is obtained. The calculation of T'_h can be reduced from $O(n_h \times m)$ to $O(m)$.

Proof. Assuming that n_h ($n_h > 2$) jobs in Γ_h ($h = 1, \ldots, S$). According to Eqs. (15)-(17), T_h can be calculated by Eq. (38). When jobs J_i and $J_{i'}$ ($i = 1, \ldots, n_h - 1$, $i' = 2, \ldots, n_h$, $i' > i$) are chosen from cellular f, T_h can be further represented as Eq. (39), where four cases are also considered. As

$$\Delta T_h = T'_h - T_h = \begin{cases} ML'_{h-1,h,m} + D_{h,i',i} + D_{h,i'-1,i'+1} - ML_{h-1,h,m} - D_{h,i'-1,i} - D_{h,i',i'+1}, i = 1 \wedge i' = 2, \ldots, n_h - 1 \\ D_{h,i-1,i'} + D_{h,i',i} + ML'_{h,h+1,m} - D_{h,i-1,i} - D_{h,i'-1,i} - ML_{h,h+1,m}, i = 1, \ldots, n_h - 1 \wedge i' = n_h \\ ML'_{h-1,h,m} + D_{h,i',i} + ML'_{h,h+1,m} - ML_{h-1,h,m} - D_{h,i'-1,i'} - ML_{h,h+1,m}, i = 1 \wedge i' = n_h \\ D_{h,i-1,i'} + D_{h,i',i} + D_{h,i'-1,i'+1} - D_{h,i-1,i} - D_{h,i'-1,i'} - D_{h,i',i'+1}, otherwise \end{cases}, \tag{41}$$

$$T'_h = T_h + \Delta T_h. \tag{42}$$

As shown in Fig. 6, a feasible solution $\pi_2 = [(J_1, J_2, J_3, J_4, J_5, J_6), \ldots, (J_{n-n_S}, \ldots, J_n)]$is used as an example to explain **Property 5**. After performing the insertion, $ML_{0,1,3}$,$D_{1,2,3}$, $D_{1,3,4}$ and the calculation of processing time after Γ_1 are not affected, while $D_{1,1,2}$, $D_{1,4,5}$, $D_{1,5,6}$ has changed to $D_{1,1,5}$, $D_{1,4,5}$, $D_{1,4,6}$.

According to these properties, the CC for evaluating the solution can be significantly reduced. In addition, for the subsequent SubSections 4.3.1-4.3.3, the makespan of the newly generated solution can be quickly calculated by using these speed-up evaluation methods based on problem-specific properties after performing the local search operations.

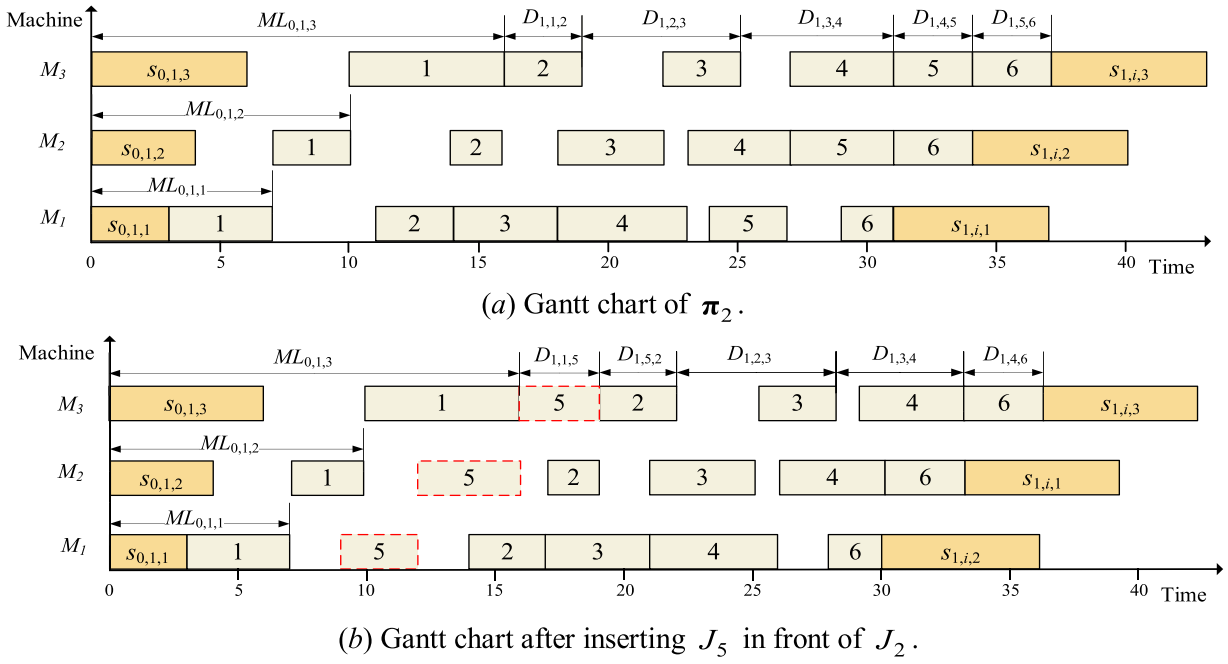


Fig. 6. Illustration of **Property 5**.

4. EEDA for DNFGSP_SDSTs

In this section, the knowledge-enhanced EDA for solving DNFGSP_SDSTs is described in detail. First, the hybrid initialization method, the EEDA-based global search, and three local search operators are proposed. Then, the perturbation and reinitialization methods are presented. Next, the computational complexity of the critical components of EEDA is analyzed. Finally, the framework of EEDA is given. The illustration of EEDA for DNFGSP_SDSTs is described in Fig. 7. Some necessary notations used in EEDA are listed in Table 6.

Table 6
Notations used in EEDA.

Symbol	*Description of Symbol*
popsize	The population size
p_{Γ_h}	The processing time of Γ_h
π_{best}	The best solution found so far
$Pop(G)$	The Gth generation population, $Pop(G) = [\pi_1^G, \pi_2^G, ..., \pi_{popsize}^G]$
PM^G	The Gth generation $(n+1) \times n$ probability model
$PM^G(x,y)$	The probability of row x, column y in PM^G
RM^G	The $(n+1) \times n$ counting matrix corresponding to PM^G
R_{PM}^G	The $1 \times n$ matrix corresponding to PM^G for roulette wheel
$R_{PM}^G(y)$	The column y of R_{PM}^G
$\pi_h^F(i)$	The ith job of Γ_h in π^F

4.1. Initialization method

Insight into previous work, high-quality initialization solutions are crucial for the speed, diversity, stability, and efficiency of HIOAs. To generate an excellent initial population with a certain diversity, two

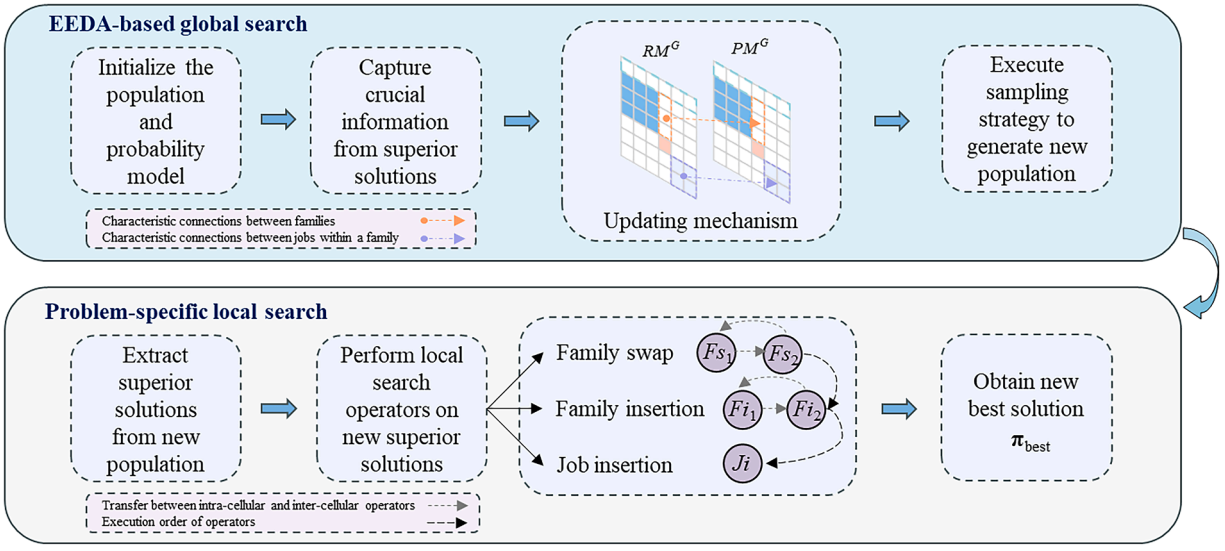


Fig. 7. Illustration of EEDA for DNFGSP_SDSTs.

Algorithm 1
TS-NEH.

Randomly generate $\boldsymbol{\pi}^F = [(\Gamma_1(1),...,\Gamma_1(n_1)),...,(\Gamma_S(1),...,\Gamma_S(n_S))]$. *//Firststage*
for $t = 1$ *to* S **do**
  Sort jobs within family according to descending $sp_i \rightarrow \Gamma_t = (\Gamma_t(1),...,\Gamma_t(n_t))$.
  Extract all jobs within Γ_t.
  for $l = 1$ *to* n_t **do**
    Text $\Gamma_t(l)$ in all the possible positions within Γ_t.
    Insert $\Gamma_t(l)$ in the position within Γ_t resulting in the minimum p_{Γ_t}.
  end for
end for
Get the improved job sequence $\boldsymbol{\pi}^F = [(\Gamma'_1(1),...,\Gamma'_1(n_1)),...,(\Gamma'_S(1),...,\Gamma'_S(n_S))]$.
Sort families according to descending $p_{\Gamma_h} \rightarrow \Gamma = \{\Gamma'_0, \Gamma'_1, ..., \Gamma'_S\}$. *//Secondstage*
for $t = 1$ *to* S **do**
  for $v = 1$ *to* F **do**
    Test family Γ_t in all the possible positions in $\boldsymbol{\pi}^C$.
    $Pos \leftarrow$ The position in $\boldsymbol{\pi}^C$ resulting in the minimum $C_{\max}$.
  end for
  Insert Γ_t in position Pos in $\boldsymbol{\pi}^C$.
end for
Output: A complete solution $\boldsymbol{\pi} = [\boldsymbol{\pi}^C, \boldsymbol{\pi}^F]$.

Algorithm 2
R-NEH.

Randomly generate $\boldsymbol{\pi}^F = [(\Gamma_1(1),...,\Gamma_1(n_1)),...,(\Gamma_S(1),...,\Gamma_S(n_S))]$. *//Firststage*
Sort families according to descending $p_{\Gamma_h} \rightarrow \Gamma = \{\Gamma'_0, \Gamma'_1, ..., \Gamma'_S\}$. *//Secondstage*
for $t = 1$ *to* S **do**
  for $v = 1$ *to* F **do**
    Test family Γ_t in all the possible positions in $\boldsymbol{\pi}^C$.
    $Pos \leftarrow$ The position in $\boldsymbol{\pi}^C$ resulting in the minimum $C_{\max}$.
  end for
  Insert Γ_t in position Pos in $\boldsymbol{\pi}^C$.
end for
Output: A complete solution $\boldsymbol{\pi} = [\boldsymbol{\pi}^C, \boldsymbol{\pi}^F]$.

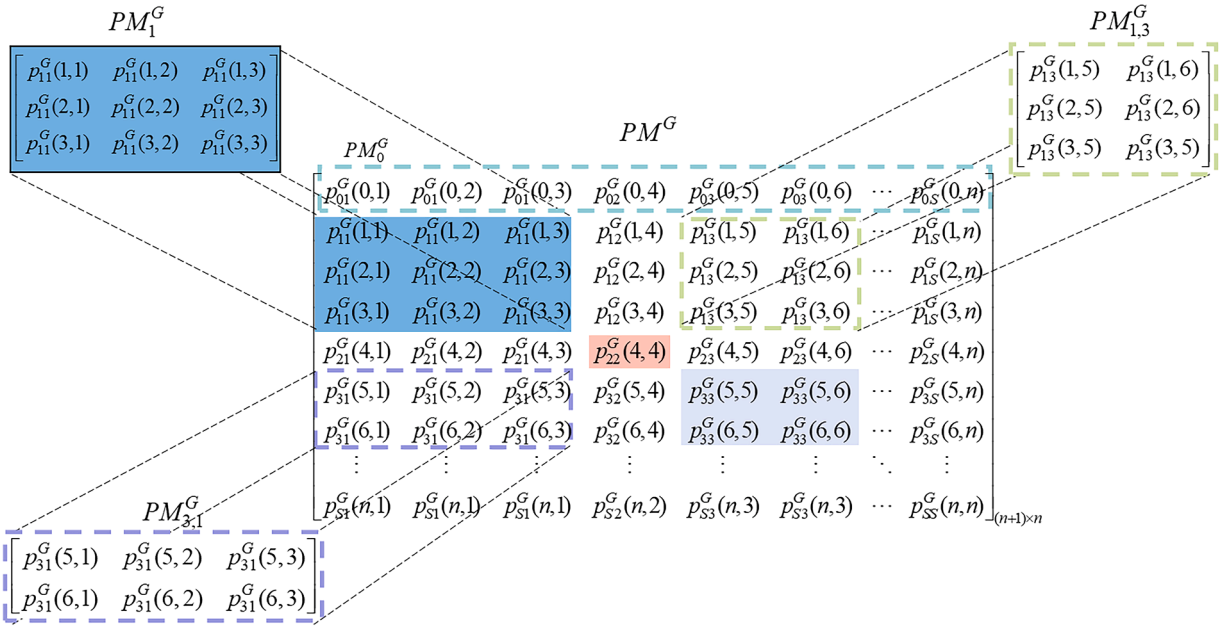


Fig. 8. Illustration of the probability model PM^G.

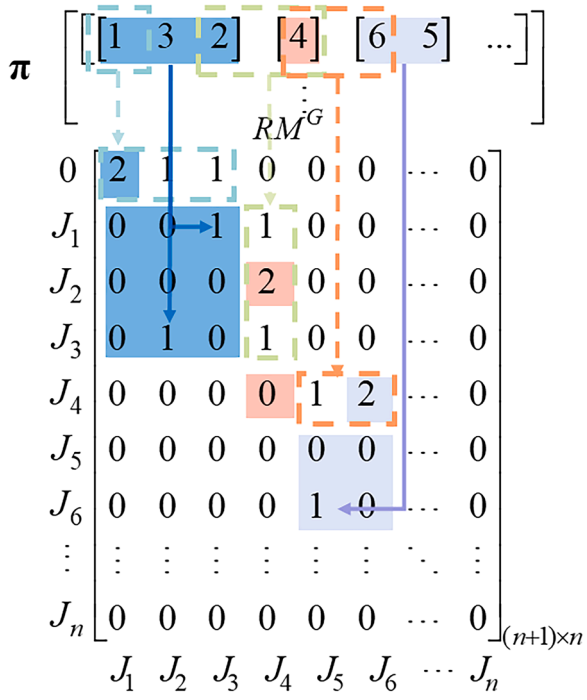


Fig. 9. Illustration of the updating mechanism for RM^G.

problem-specific heuristics are introduced in this section. When addressing DNFGSP_SDSTs, determining both the family sequence $\boldsymbol{\pi}^C$ and the job sequence $\boldsymbol{\pi}^F$ for each solution $\boldsymbol{\pi}$ in the population $Pop(G)$ is essential. Inspired by the constructive heuristics proposed in Ref. [66], we develop a two-stage heuristic based on NEH (TS-NEH) and a random heuristic based on NEH (R-NEH). TS-NEH primarily aims at refining the job orders of the families and reducing the completion time of the cellulars, while R-NEH is devised to ensure population diversity. In TS-NEH, the LPT rule [66] is adopted in the first stage to obtain initial job sequences for families. Then, the NEH rule is applied to further fine-tune these partial job sequences with the aim of minimizing the completion time of each family. Based on these partial job sequences, the total job sequence $\boldsymbol{\pi}^F$ is generated. In the second stage, the total processing time of each family is calculated separately and the initial family sequence Γ is generated using the LPT rule. Then, the NEH rule is applied on initial family sequence Γ to obtain to obtain the final family sequence $\boldsymbol{\pi}^C$. This process results in generating a high-quality initial solution $\boldsymbol{\pi} = [\boldsymbol{\pi}^C, \boldsymbol{\pi}^F]$. The pseudo-code of TS-NEH is presented in Algorithm 1. In R-NEH, the job sequence for each family is generated randomly, and the total job sequence $\boldsymbol{\pi}^F$ is formed from these partial job sequences. Then, both the LPT and NEH rules are applied similarly to the second stage of R-NEH. Both of these proposed heuristics are utilized to generate the initial population $Pop(0)$, with one solution generated via TS-NEH and the remaining solutions generated by R-NEH. The pseudo-code of R-NEH is shown in Algorithm 2.

4.2. EEDA-based global search

This section introduces an EEDA-based global search, which consists of three key components, *i.e.*, a probability model with problem-specific knowledge, a family-based updating mechanism, and a specific sampling strategy. This knowledge-dependent global search can effectively drive the search behavior towards promising regions. Each of the components of EEDA is detailed in the following subsections.

4.2.1. Probability model

As a crucial component of EDA, probability models are commonly utilized to capture potential patterns implicit in high-quality solutions [20]. In EEDA, the proposed probability model not only preserves information regarding relationships between jobs but also accumulates information about characteristic connections across families. The matrix representation of the proposed probabilistic model PM^G of the Gth generation is shown in Eq. (43). Note that partition matrices within PM^G are categorized based on the feature information of families and jobs. To visually demonstrate the proposed probability model PM^G, a graphical representation of it is shown in Fig. 8. Within PM^G, PM_0^G denotes the estimation of the first job in cellulars, PM_a^G records the inherent connections of jobs within Γ_a, and $PM_{a,b}^G$ represents connections among jobs across the families Γ_a and Γ_b. Note that $p_{aa}^G(x,y)$ indicates the probability value that the job following J_x is J_y in Γ_a, while $p_{ab}^G(x,y)$ denotes the probability value that the last job J_x of Γ_a is followed by the first job J_y of Γ_b. Through these partitioned matrices, PM^G establishes a close connection between the family and the job, ensuring that the structural features and valuable information of superior solutions are fully captured without omission.

$$PM^G = \begin{bmatrix} p_{01}^G(0,1) & \cdots & p_{0S}^G(0,n) \\ p_{11}^G(1,1) & \cdots & p_{1S}^G(1,n) \\ \vdots & \ddots & \vdots \\ p_{S1}^G(n,1) & \cdots & p_{SS}^G(n,n) \end{bmatrix}_{(n+1)\times n} \tag{43}$$

4.2.2. Family-based updating mechanism

To precisely guide the global search trend and track promising regions in the solution space, the probability model must be appropriately adapted to critical characteristics. Hence, a family-based updating mechanism is introduced to adjust the probability model, preserving promising patterns while preventing the disruption of connections between jobs and families. The counting matrix RM^G is designed to capture valuable information implicit in high-quality solutions so as to update the PM^G. Each element $r_{ab}^G(x,y)$ in RM^G corresponds to the probability value $p_{ab}^G(x,y)$ in PM^G, as depicted in Eq. (44).

$$RM^G = \begin{bmatrix} r_{01}^G(0,1) & \cdots & r_{0S}^G(0,n) \\ r_{11}^G(1,1) & \cdots & r_{1S}^G(1,n) \\ \vdots & \ddots & \vdots \\ r_{S1}^G(n,1) & \cdots & r_{SS}^G(n,n) \end{bmatrix}_{(n+1)\times n} \tag{44}$$

To calculate each element $r_{ab}^G(x,y)$ of RM^G, two indicator functions are employed to record information about the relationship between jobs and families extracted from high-quality solutions. $I\_r_{aa}^G(x,y)$ serves as an indicator function to record job correlations in Γ_a, which indicates job J_x is followed by job J_y in Γ_a. $I\_r_{ab}^G(x,y)$ indicates the relationship between J_x of Γ_a and J_y of Γ_b. If the first job of cellular is J_y and its family is Γ_b, it can be indicated by $I\_r_{0b}^G(0,y)$. To capture the information between families efficiently, when Γ_a is followed by Γ_b in the feasible solution, $I\_r_{ab}^G(x,y)$ is set to 1. In addition, if J_x is followed by J_y, $I\_r_{ab}^G(x,y)$ is set to 2. The calculations of $I\_r_{aa}^G(x,y)$, $I\_r_{ab}^G(x,y)$, $I\_r_{0b}^G(0,y)$, and $r_{ab}^G(x,y)$ are illustrated by Eqs. (45)-(48), respectively. The family-based updating mechanism enables the recording of the relationship information between jobs while recording the information between families within high-quality solutions, further exploring the relationship information

Algorithm 3
PickFirstJob.

Input: PM^G.
for $y=1$ *to* n **do**
  $R^G_{PM}(y)=\sum_{z=1}^{y} PM^G(0,z)$. //*Calculate the cumulative probability*
end for
Produce a random value p_r, where $p_r \in \left[0, \sum_{y=1}^{n} R^G_{PM}(y)\right)$.
if $p_r \in \left[0, R^G_{PM}(1)\right)$ **then**
  $\pi^F_h(i) \leftarrow 1$.
else
  for $t=1$ *to* $n-1$ **do** //*Roulette wheel selection*
    if $p_r \in \left[\sum_{y=1}^{t} R^G_{PM}(y), \sum_{y=1}^{t+1} R^G_{PM}(y)\right)$ **then**
      $\pi^F_h(i) \leftarrow t+1$, **break**.
    end if
  end for
end if
for $t=0$ *to* n **do** //*Avoid repeated selection of jobs*
  $PM^G(t, \pi^F_h(i))=0$.
end for
Output: $\pi^F_h(i)$.

Algorithm 4
PickNextJob.

Input: PM^G, $\pi^F_h(i-1)$.
for $y=\Gamma_h(1)$ *to* $\Gamma_h(n_h)$ **do**
  $R^G_{PM}(y)=\sum_{z=1}^{y} PM^G(\pi^F_h(i)), z)$. //*Calculate the cumulative probability*
end for
Produce a random value p_r, where $p_r \in \left[0, \sum_{y=1}^{n} R^G_{PM}(y)\right)$.
if $p_r \in \left[0, R^G_{PM}(1)\right)$ **then**
  $\pi^F_h(i) \leftarrow \Gamma_h(1)$.
else
  for $t=1$ *to* $n-1$ **do** //*Roulette wheel selection*
    if $p_r \in \left[\sum_{y=1}^{t} R^G_{PM}(y), \sum_{y=1}^{t+1} R^G_{PM}(y)\right)$ **then**
      $\pi^F_h(i) \leftarrow \Gamma_h(t)+1$, **break**.
    end if
  end for
end if
for $t=0$ *to* n **do** //*Avoid repeated selection of jobs*
  $PM^G(t, \pi^F_h(i))=0$.
end for
Output: $\pi^F_h(i)$.

between jobs and families. Fig. 9 shows the illustration of the updating mechanism for the counting matrix RM^G. The initialization method of the initial probability values of different partition matrices in the probability model PM^G is given in Eq. (49). The updating mechanism of PM^G is described by Eq. (50), where lr is the learning rate.

$$I\_r^G_{aa}(x,y)=\begin{cases}1, & \text{if } x=\Gamma_a(i) \text{ and } y=\Gamma_a(i+1)\\ 0, & \text{otherwise}\end{cases}, \quad a=1,...,S,\ \ i=1,...,n_a-1 \tag{45}$$

$$I\_r^G_{ab}(x,y)=\begin{cases}1, & \text{if } x=\Gamma_a(n_a),\ y=\Gamma_b(1) \text{ and } a+1=b\\ 0, & \text{otherwise}\end{cases}, \quad a=1,...,S-1 \tag{46}$$

$$I\_r^G_{0b}(0,y)=\begin{cases}2, \Gamma_b=\Gamma^f_1 \text{ and } \Gamma_b(0)=J_y\\ 1, \Gamma_b=\Gamma^f_1 \text{ and } \Gamma_b(0)\neq J_y\\ 0, \text{otherwize}\end{cases}, \quad b=1,2,...,S, f=1,2,...,F, J_y\in\Gamma_b \tag{47}$$

$$r^G_{ab}(x,y)=\sum_{t=1}^{popsize} I_t\_r^G_{ab}(x,y), a=0,1,2,..,S, b=1,2,...,S. \tag{48}$$

$$\begin{cases} p_{0b}^0(0,y) = 1/(S \times n_b) \\ p_{aa}^0(x,y) = 1/n_a^2 \\ p_{ab}^0(x,y) = 1/(n_a \times n_b \times (S-1)) \end{cases}, \quad a,b = 1,2,...,S, a \neq b, J_x \in \Gamma_a, J_y \in \Gamma_b \tag{49}$$

$$\begin{cases} p_{0b}^1(0,y) = (1-lr) \times p_{0b}^0(0,y) + lr \times r_{0b}^1(0,y) \Big/ \left(\sum_{l=1}^{S} \sum_{k=1}^{n_l} r_{0l}^1(0,k) \right) \\ p_{aa}^1(x,y) = (1-lr) \times p_{aa}^0(x,y) + lr \times r_{aa}^1(x,y) \Big/ \left(\sum_{k=1}^{n_a} \sum_{k'=1}^{n_a} r_{aa}^1(k,k') \right) \\ p_{ab}^1(x,y) = (1-lr) \times p_{ab}^0(x,y) + lr \times r_{ab}^1(x,y) \Big/ \left(\sum_{k=1}^{n_a} \sum_{k'=1}^{n_b} r_{ab}^1(k,k') \right) \end{cases}, \quad a,b = 1,2,...,S, a \neq b, J_x \in \Gamma_a, J_y \in \Gamma_b \tag{50}$$

According to the above, the steps of the family-based updating mechanism are as follows.

Step 1: Initialize the problem-specific probability model PM^0 via Eq. (49), and set $G = 1$.

Step 2: Select $hp \times popsize$ solutions from $Pop(G)$ and calculate RM^G by Eqs. (45)-(48), where hp represents the percentage of high-quality sub-populations.

Step 3: Update the probability model PM^G according to Eq. (50).

Step 4: Set $G = G + 1$. If termination conditions are not met, go to **Step 2**.

4.2.3. Sampling strategy

Suitable sampling strategies can improve the accuracy and stability of distribution estimation and increase computational efficiency. Probabilistic models can implicitly capture crucial characteristics inherent in the distribution of superior solutions in the solution space by mapping promising patterns to corresponding probability values. To effectively extract the correlation between families and jobs recorded in PM^G, this section provided a problem-specific sampling strategy to generate new solutions with quality and diversity. To generate the first family and its job sequence, we define *PickFirstJob* (shown in Algorithm 3) as the first job selection function, and *PickNextJob* (shown in Algorithm 4) as the next job selection function in the same family. The first job is picked by *PickFirstJob*, and the subsequent ones are chosen by using *PickNextJob* until all the jobs in this family have been selected. Then, the next job and its family are selected by *PickFamily* (shown in Algorithm 5), after which *PickNextJob* is used to obtain all jobs in Γ_k, and so on until all families and jobs are chosen. Thus, the new population generation process is denoted by *NewPopGeneration*, as detailed in Algorithm 6, and Fig. 10 illustrates the generation process of the feasible solution in a cellular. To be specific, first, PM_0^G is sampled by *PickFirstJob* to generate the first job J_2 of Γ_1, then PM_1^G is sampled by *PickNextJob* to obtain the two jobs J_1 and J_3. Next, *PickFamily* is used to sample $PM_{1,h}^G$ to select the next family Γ_3 and its first job J_6. Finally, the process repeats with *PickJob* and *PickFamily* until all families and jobs are selected.

4.3. Local intensification

It is common that embedding local searches in EDAs for intensification can effectively enhance the local exploitation capability, focusing on fine-grained seeking for promising regions found by global searches. In order to strike the balance between global exploration and local exploitation of EEDA, three problem-specific local search operators are proposed to refine the obtained solutions. These local search operators are described in detail in the following subsections.

4.3.1. Local search for family swap

As shown in Algorithm 7, a family Γ_a is randomly selected from the critical cellular c in $\boldsymbol{\pi}^C$, and then Γ_a is swapped with all other families in each cellular to find the swapped family resulting in the minimum $C_{\max}(\boldsymbol{\pi})$. Next, the positions of two families in $\boldsymbol{\pi}^C$ are exchanged to generate a new feasible solution $\boldsymbol{\pi}'$. When evaluating the solution $\boldsymbol{\pi}'$, the CC of calculating $C_{\max}(\boldsymbol{\pi}')$ can be reduced based on **Property 2** and **Property 4**.

4.3.2. Local search for family insertion

First, a family Γ_a is randomly taken from the critical cellular c (the one with the maximum completion time) in $\boldsymbol{\pi}^C$. Then, Γ_a is reinserted into all possible positions of each cellular to determine the inserted position with the minimum makespan. Next, insert Γ_a into the identified position of $\boldsymbol{\pi}^C$ to generate a new feasible solution $\boldsymbol{\pi}'$. The pseudo-code is shown in Algorithm 8, which provides a detailed representation of this process. To reduce the CC in the calculation of $C_{\max}(\boldsymbol{\pi}')$, **Property 3** and **Property 5** can be adopted here.

Algorithm 5
PickFamily.

Input: PM^G, $\pi_h^F(i-1)$.
for $y = 1$ *to* n **do**
  $R_{PM}^G(y) = \sum_{z=1}^{y} PM^G(\pi_h^F(i-1), z)$. *//Calculate the cumulative probability*
end for
Produce a random value p_r, where $p_r \in \left[0, \sum_{y=1}^{n} R_{PM}^G(y)\right)$.
if $p_r \in \left[0, R_{PM}^G(1)\right)$ **then**
  $\pi_h^F(i) \leftarrow \Gamma_h(1)$.
else
  for $t = 1$ *to* $n-1$ **do** *//Roulette wheel selection*
    if $p_r \in \left[\sum_{y=1}^{t} R_{PM}^G(y), \sum_{y=1}^{t+1} R_{PM}^G(y)\right)$ **then**
      $\pi_h^F(i) \leftarrow \Gamma_h(t+1)$, **break**.
    end if
  end for
end if
for $t = 0$ *to* n **do** *//Avoid repeated selection of jobs.*
  $PM^G(t, \pi_h^F(i)) = 0$.
end for
Output: $\pi_h^F(i)$.

Algorithm 6
NewPopGeneration.

Input: PM^G, $popsize$, F.
Set $Pop(G)=[\boldsymbol{\pi}_1,\boldsymbol{\pi}_2,...,\boldsymbol{\pi}_{popsize}]$.
for $k=1$ *to* $popsize$ **do**
  for $v=1$ *to* F **do**
    $\pi_h^F(1)\leftarrow PickFirstJob$. //*Algorithm 2*
    for $r=2$ *to* n_h-1 **do**
      $\pi_h^F(r)\leftarrow PickNextJob$. //*Algorithm 3*
    end for
    Assign Γ_h to cellular v in $\boldsymbol{\pi}^C$.
  end for
  for $l=1$ *to* $(S-F)$ **do**
    Determine the cellular s with the smallest current processing time.
    $\pi_h^F(1)\leftarrow PickFamily$ //*Algorithm 4*
    Assign Γ_h to cellular s in $\boldsymbol{\pi}^C$.
    for $r=2$ *to* n_h-1 **do**
      $\pi_h^F(r)\leftarrow PickNextJob$. //*Algorithm 3*
    end for
  end for
  $\boldsymbol{\pi}_k\leftarrow[\boldsymbol{\pi}^C,\boldsymbol{\pi}^F]$.
end for
Output: $Pop(G)$.

4.3.3. Local search for job insertion

Randomly select a family Γ_a from critical cellular c in $\boldsymbol{\pi}^C$, then a job J_x is randomly taken out from Γ_a in $\boldsymbol{\pi}^F$. Reinsert J_x into all possible positions in Γ_a to find the position with the smallest $C^c_{\max}(\boldsymbol{\pi})$. Next, insert the job into the found position to generate a new solution $\boldsymbol{\pi}'$. The pseudo-code of the procedure is provided as Algorithm 9. According to **Property 6**, the CC of calculating $C_{\max}(\boldsymbol{\pi}')$ can be reduced.

4.4. Perturbation and reinitialization

It is useful to add specific perturbation and reinitialization methods to the global search of EDAs to enhance the diversity of search behaviors and skip search stagnation, thus effectively combating the challenge of falling into local optima by introducing small variations to the feasible solutions [3,20]. In EEDA, if there is no improvement after α iterations, families are randomly selected from each cellular of $\boldsymbol{\pi}^C_{\mathbf{best}}$ to form a temporary set Y. Then, the families of the set are sequentially inserted into all possible positions in $\boldsymbol{\pi}^C_{\mathbf{best}}$ to identify the positions that minimize $C_{\max}(\boldsymbol{\pi}_{\mathbf{best}})$, until all the families in the set have been inserted. Next, $\boldsymbol{\pi}_{\mathbf{best}}$ is updated. Finally, due to the lack of updates to $\boldsymbol{\pi}_{\mathbf{best}}$ for several generations, this may result in overly similar information within the probability model. However, resetting the entire probability model would lead to the loss of previously retained high-quality information. Hence, all values of the probability model PM^G are partially reset by Eq. (51) to prevent premature convergence. The detailed description of the perturbation and reinitialization methods is shown in Algorithm 10.

$$\begin{aligned} &p^G_{xy}(a,b)=p^G_{xy}(a,b)\times(1-lr)+p^0_{xy}(a,b)\times lr,\\ &a=0,1,...,S,b=1,2,...,S,x=0,1,...,n_a,y=1,2,...,n_b \end{aligned}. \tag{51}$$

4.5. Computational complexity of EEDA

In EEDA, the analysis of CC for critical components involves the analysis of initialization, sorting operation, updating mechanism, sampling and local search strategies. Firstly, initializing the initial population $Pop(0)$ by Algorithm 1 with the CC of $O(n\times S^2)$ and by Algorithm 2 with the CC of $O[(popsize-1)\times S^2]$; Secondly, sorting $hp\times popsize$ high-quality solutions from $Pop(G)$ with the CC of $O(popsize\times n\times m)$; Thirdly, updating the probability model PM^G by using $hp\times popsize$ high-quality solutions with the CC of $O(hp\times popsize\times n\times(n+1))$; Fourthly, sampling the probability model PM^G to generate $popsize$ solutions with the CC of $O(popsize\times n)$; Finally, sorting new $hp\times popsize$ high-quality solutions from $Pop(G+1)$ with the CC of $O(popsize\log popsize)$ and performing three problem-specific local searches with the CC of $O(S\times m+n\times m)$. It is noteworthy that the five problem properties provided in Section 3.2 are employed to reduce the CC of these local searches from $O(S\times n\times m+n^2\times m)$ to $O(S\times m+n\times m)$. The CC of the critical components of EEDA can be detailed in Table 7. Thus, the total CC (TCC) of EEDA can be represented by Eq. (52). If critical components are not all active in the iteration, the CC of EEDA is about $O(popsize\log popsize+S\times m+n\times m)$; if these components are active, the CC can be represented as $O(hp\times popsize\times n\times(n+1))$, which is still acceptable for EEDA to address DNFGSP_SDSTs.

$$\begin{aligned} TCC_{(EEDA)}&=O(popsize\times S^2)+O(n\times S^2)\\ &+O(S\times m)+O(n\times m)+O(popsize\times n\times m)\\ &+O(popsize\log popsize)+O(hp\times popsize\times n^2)\\ &+O(popsize\times n) \end{aligned}. \tag{52}$$

4.6. Framework of EEDA with problem-specific knowledge

This subsection gives the framework of EEDA. In EEDA, the probability model with problem-specific knowledge is devised to accumulate valuable pattern information of high-quality solutions, and the family-based updating mechanism is designed to update this probability model. To generate new solutions with certain quality, the sampling strategy is proposed for sampling probability model to enhance the search capability of EEDA. Then, three local search operators are developed to refine the solutions found by the global search. Moreover, perturbation and reinitialization methods are presented to avoid premature convergence. The flowchart of EEDA is shown in Fig. 11. The detailed description are as follows.

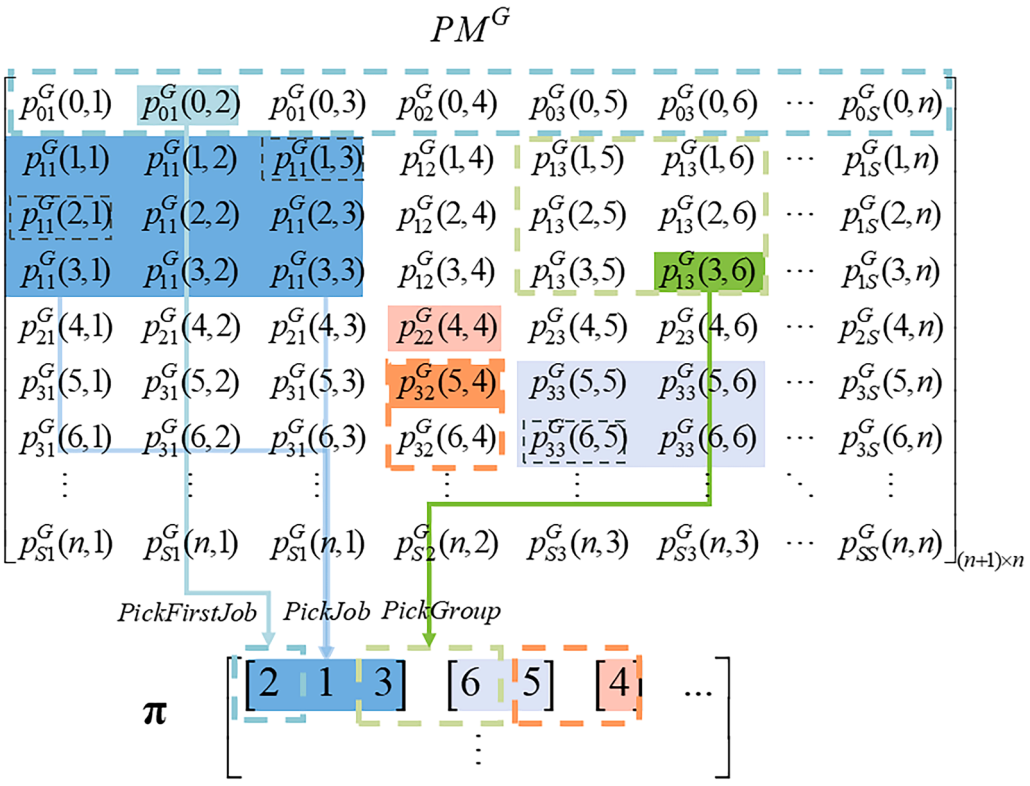


Fig. 10. Illustration of sampling from PM^G to generate a cellular of $\boldsymbol{\pi}$.

Algorithm 7
LsFam_Swap.

Input: $\boldsymbol{\pi}$.
for $v = 1$ *to* F **do**
  Find the critical cellular c.
  if there are two or more critical cellulars:
    Randomly select a critical cellular c.
  end if
end for
Randomly extract a family Γ_a from cellular c.
for $v = 1$ *to* F **do**
  Swap Γ_a with all other families in $\boldsymbol{\pi}^C$.
  Record the family in cellular v of $\boldsymbol{\pi}^C$ with the minimum $C_{\max}(\boldsymbol{\pi})$ as Fam.
end for
Swap Γ_a with Fam in $\boldsymbol{\pi}^C$ to generate a new solution $\boldsymbol{\pi}'$.
Output: $\boldsymbol{\pi}'$.

Algorithm 8
LsFam_Insert.

Input: $\boldsymbol{\pi}$.
for $v = 1$ *to* F **do**
  Find the critical cellular c.
  if there are two or more critical cellulars:
    Randomly select a critical cellular c.
  end if
end for
Randomly select a family Γ_a from cellular c.
for $v = 1$ *to* F **do**
  Reinsert family Γ_a in all the possible positions in cellular v of $\boldsymbol{\pi}^C$.
  Record the position in cellular v of $\boldsymbol{\pi}^C$ with the minimum $C_{\max}(\boldsymbol{\pi})$ as Pos.
end for
Insert Γ_a at position Pos in $\boldsymbol{\pi}^C$ to generate a new solution $\boldsymbol{\pi}'$.
Output: $\boldsymbol{\pi}'$.

Algorithm 9
LsJob_Insert.

Input: $\boldsymbol{\pi}$.
Randomly select a family Γ_a from critical cellular c.
Randomly extract J_x from Γ_a.
for $t = 1$ *to* n_a **do**
  Reinsert J_x into all possible positions in Γ_a.
  Record the position in Γ_a with the minimum $C^c_{\max}(\boldsymbol{\pi})$ as Pos.
end for
Insert Γ_a into position Pos in Γ_a to generate a new solution $\boldsymbol{\pi}'$.
Output: $\boldsymbol{\pi}'$.

Step 1: Initialize key parameters of EEDA, including *popsize*, *hp*, *lr*, and α, all of which are calibrated in Section 5.1. Initialize probability model PM^G and generate initial population $Pop(0)$ by using the hybrid initialization method in Section 4.1 and SubSection 4.2.2.

Step 2: Evaluate each feasible solution and sort to obtain $popsize \times hp$ high-quality solutions.

Step 3: Calculate the counting matrix RM^G and update the probability model PM^G by RM^G via Eqs. (45)-(48) and Eq. (50) in SubSection 4.2.2.

Step 4: Generate $Pop(G+1)$ by sampling PM^G and update $\boldsymbol{\pi}_{\mathbf{best}}$ in SubSection 4.3.3.

Step 5: Perform three types of local search operators on $\boldsymbol{\pi}_{\mathbf{best}}$ in Section 4.3.

Step 6: Determine if the search is stagnant. If $\boldsymbol{\pi}_{\mathbf{best}}$ is not updated in successive α generations, perform perturbation and reinitialization methods given in Section 4.4.

Step 7: Examine the termination condition. If it is not met, go to **Step 2**, otherwise output $\boldsymbol{\pi}_{\mathbf{best}}$.

According to the above steps, it can be observed that the performance of EEDA is well emphasized and balanced by combining the advantages of global exploration and local exploitation. These strategies strengthen the ability of EEDA to solve DNFGSP_SDSTs.

5. Experimental comparisons and statistical analysis

In this section, a series of statistical experiments and computational comparisons are conducted to evaluate the efficiency and effectiveness of the proposed EEDA in solving the DNFGSP_SDSTs. Section 5.1 provides details of the experimental setups, including the introduction of the benchmark instances, the experimental environment, and the evaluation metrics. In Section 5.2, the parameters are carefully calibrated and analyzed to refine the performance of EEDA, ensuring the best behavior in handing various problem instances. Section 5.3 verifies the proposed MILP model on small- and medium-scale instances by using the Gurobi solver, confirming the validity of the MILP model in terms of both scale and computational complexity. In Section 5.4, the efficacy of each element of EEDA is examined, revealing the impact and implications of improvement strategies. Finally, comprehensive comparisons of EEDA *versus* several state-of-the-art approaches are performed, and the experimental results are analyzed in detail in Section 5.5. Some

Table 7
Computational complexity of critical components of EEDA.

Crucial components	CC	Analyzes
Initialization	$O((popsize-1)\times S^2)+O(n\times S^2)$	The CC for TS-NEH: $O(n\times S^2)$ The CC for R-NEH: $O[(popsize-1)\times S^2]$
Local search	$O(S\times m)+O(n\times m)$	The CC for family insertion: $O(S\times m)$ The CC for family swap: $O(S\times m)$ The CC for job insertion: $O(n\times m)$
Global search	$O(popsize\times n\times m)+O(popsize\log popsize)+O(hp\times popsize\times n\times(n+1))+O(popsize\times n)$	The CC for calculating: $O(popsize\times n\times m)$ The CC for sorting:$O(popsize\log popsize)$ The CC for updating mechanism: $O(hp\times popsize\times n\times(n+1))$ The CC for sampling: $O(popsize\times n)$

Algorithm 10
Perturbation and reinitialization.

Input: $\boldsymbol{\pi}_{\mathbf{best}}$, PM^G, PM^0.
for $v = 1$ *to* F **do** *//Perturbation*
  Randomly generate a number rm from 1 and 2.
  Randomly extract rm family from cellular v of $\boldsymbol{\pi}^C_{\mathbf{best}}$ and put them into set Υ.
end for
for $t = 1$ *to* $length(\Upsilon)$ **do**
  for $v = 1$ *to* F **do**
    Test family Γ_t in all the possible positions in cellular v of $\boldsymbol{\pi}^C_{\mathbf{best}}$.
    $Pos \leftarrow$ The position in cellular v of $\boldsymbol{\pi}^C_{\mathbf{best}}$ with the minimum $C_{\max}(\boldsymbol{\pi}_{\mathbf{best}})$.
  end for
  Insert Γ_t at position Pos in $\boldsymbol{\pi}^C_{\mathbf{best}}$ to update $\boldsymbol{\pi}_{\mathbf{best}}$.
end for
for $t = 1$ *to* $n+1$ **do** *//Reinitialization*
  for $v = 1$ *to* n **do**
    $PM^G(t,v) = PM^G(t,v)\times(1-lr) + PM^0(t,v)\times lr$. //Eq. (51)
  end for
end for
Output: $\boldsymbol{\pi}_{\mathbf{best}}$ and new PM^G.

conclusions are drawn, and the superiority of EEDA is discussed in Section 5.6.

5.1. Experimental setup

To evaluate the efficacy of EEDA for solving DNFGSP_SDSTs across various scenarios, a set of 810 instances was adapted from a benchmark dataset provided by Pan et al. [3], which is widely used in DFGSPs. This dataset was grouped into a total of 162 combinations, each of which contains five instances. Each instance is characterized by parameters including three different scales of setup times, six different cellular scales $F \in \{2,3,4,5,6,7\}$, three machine scales $m \in \{2,4,6\}$, and three family scales $S \in \{20,40,60\}$. For the instances of each combination, n_h and $p_{i,j}$ are uniformly distributed integers ranging from 1 to 10, and the setup times $s_{h,h,j}$ for small, medium, and large types are randomly generated integers from uniform distribution ranges [1,20], [1,40], and [1,60], respectively. All algorithms, including the MILP solver, were implemented by Python programming language, compiled by Python 3.11 in PyCharm Studio 2021. The MILP solver typically employs the branch-and-cut technique that incorporates cutting planes and branch-and-bound methods, which works by branching and bounding, while utilizing cutting planes to tighten the LP relaxation. The numerical experiments were executed independently on a PC equipped with an Intel(R) Core i7–8700 M CPU @ 3.20 GHz processor and 16 GB RAM with a 64-bit Windows 7 OS. To ensure fairness and accuracy, the same library functions and backend running environments were adopted, and no other programs were executed in parallel during algorithm implementation. Under the same experimental conditions, each algorithm was independently executed for 10 runs on 810 instances, with the same maximum elapsed CPU time of $\rho \times m \times S$ milliseconds, where ρ has been tested at three values $\rho \in \{100,200,300\}$. To compare the performance of the proposed EEDA with other efficient algorithms, the relative percentage increase (RPI) was used as a comparison standard, which is calculated by Eq. (53) to measure the performance of algorithms in the experiments.

$$RPI_i = \frac{c_i - c_{best}}{c_{best}} \times 100, \tag{53}$$

where c_{best} is the best fitness value found by all comparison algorithms for each instance, and c_i is the best fitness value determined by the given algorithm, *i.e.*, EEDA, TIG_{v1} [24], TIG_{v2} [49], CIG [67], CCEA [3], DIWO [68], CCABC [69], DABC [70] and IABC [55]. Clearly, smaller RPI_x means better performance of the algorithm. In addition, if there are R runs of each algorithm for each instance, the average RPI (ARPI) is employed as the response variable (RV) to evaluate the effectiveness of the algorithm, which is calculated as $ARPI = \sum_{i=1}^{R} RPI_i / R$, where R represents the total number of runs.

5.2. Sensitivity analysis of parameters

Since parameter calibration significantly impacts the performance of algorithms, superior search behaviors greatly depend upon their parameter configurations. To determine the proper parameter combination for EEDA, the design of experiment (DOE) method [71] is implemented to investigate the influence of parameters on EEDA. EEDA incorporates four key parameters: (1) population size *popsize* (2) percentage of high-quality sub-populations *hp*; (3) learning rate *lr*; (4) perturbation factor α. Suitable values (levels) for these parameters were determined both through experience and preliminary experiments, so the levels of parameters are listed below: $popsize = \{30,50,70,90\}$, $lr = \{0.05,0.1,0.3,0.5\}$, $hp = \{0.1,0.3,0.5,0.7\}$, and $\alpha = \{5,10,15,20\}$. Each of these parameters has four levels, resulting in a total of $4 \times 4 \times 4 \times 4 = 256$ different configurations, as shown in Table 8. To evaluate the effect of parameter settings and eliminate the risk of overfitting, we used the ARPI, an evaluation metric in the averaged sense, and inspired by Ref. [3], we regenerated 81 instances at three setup time scales with $F \in \{2, 4, 6\}$, $m \in \{2, 4, 6\}$, and $S \in \{20, 40, 60\}$. To adequately reveal the effects between parameters, each configuration was independently tested 20 times on a set of 81 instances, and the maximum elapsed CPU time was set to $300 \times m \times S$ milliseconds for parameter calibration. As a result, it required at least 268.8 CPU days to finish all calibration experiments. However, due to the parallel computing capability of PC with multiple cores, it took almost 13.4 days to acquire all numerical results. To analyze these results and assess the significance of parameter interactions, this section employed the multifactor analysis of variance (ANOVA). The ANOVA results for parameters are provided in Table 8, with *p*-values marked in **bold** when they are smaller than 0.05.

As observed from Table 9, the *p*-values of *popsize*, *lr*, and α are all less than 0.05, indicating the significance of these three parameters in influencing the performance of the proposed EEDA. It is worth noting that since *popsize* has the largest *F*-ratio, this implies that population size has a much greater effect on the efficiency of EEDA. Fig. 12 provides the main effect plots of the parameters, indicating that the impact of *popsize* is more significant than the other parameters. From Fig. 12, the most

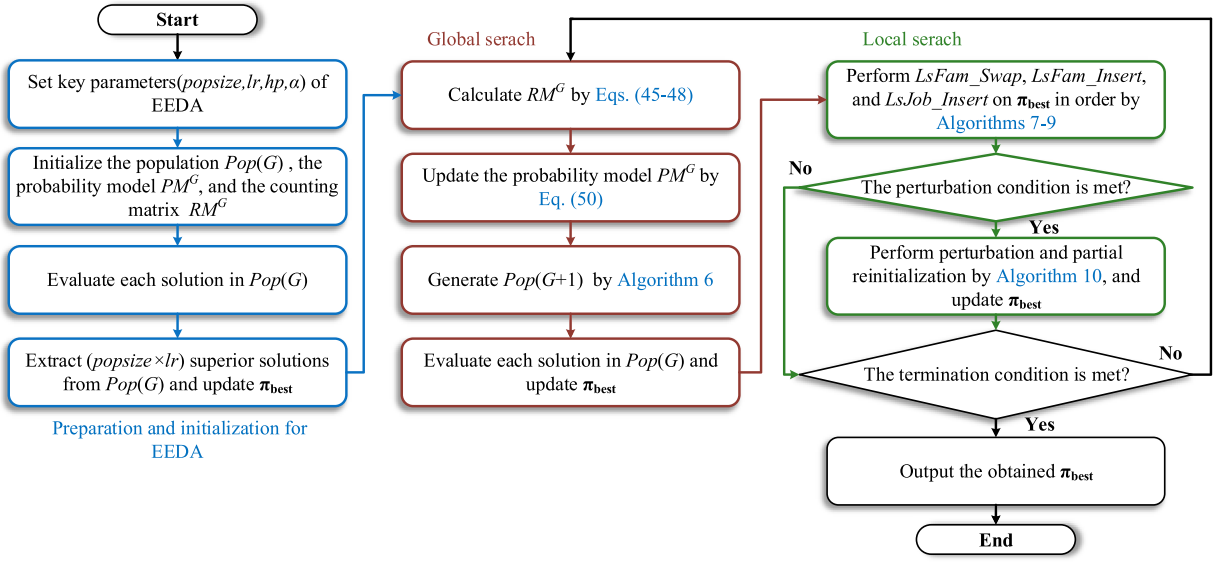


Fig. 11. The flowchart of EEDA for DNFGSP_SDSTs.

favorable performance is achieved with $popsize = 50$, while the least favorable outcome is associated with $popsize = 90$. This observation suggests that insufficient diversity of solutions may result from smaller *popsize*, and larger values of *popsize* may increase computational complexity and decrease search efficiency. In contrast, there is also a significant effect of lr in EEDA, with the F-ratio slightly lower than that of *popsize*. It can be seen in Fig. 12 that $lr = 0.1$ can yield the best results, potentially because smaller values of the learning rate lr may fail to retain sufficient valuable information in the probability model PM^G, whereas larger values may lead to homogenous information. The trend curves of hp in Fig. 12 further support the superior performance achieved by $hp = 0.3$. This phenomenon arises due to the suboptimal solutions occupy excessive search resources when hp exceeds a certain value. Conversely, when hp falls below a certain value, it risks constraining the informational diversity essential for the generation of high-quality solutions. Furthermore, for the parameter α, if it is too small, there is a risk of prematurely introducing perturbations before convergence, resulting in wastage of computational resources. Conversely, if α exceeds a certain value, it may lead to excessive iterations of futile searches, also resulting in a waste of computational resources. As indicated in Fig. 12, the best results are obtained when the perturbation factor α is set to 15, indicating that when 15 iterations are not improved then the perturbation is performed to re-activate the search. In addition, it is further observed from Fig. 12 that small changes in parameters lead to slight swings in ARPI and these variations remain small. Nonetheless, EEDA shows stability and strong robustness at different parameter settings, as evidenced by the consistent performance under varied conditions.

While main effects provide valuable insights into the choice of parameters, analyzing potential interactions between parameters is equally essential. Table 9 provides a detailed examination of the bifactor interactions, with corresponding interaction effect plots presented in Fig. 13. Notably, the p-values of $popsize * lr$, $popsize * \alpha$, $lr * \alpha$ are less than 0.05, indicating statistical significance in these interactions. However, it is noteworthy that the F-ratios of single parameters surpass those of the interactions, suggesting that the interaction effects between parameters may be remarkably weak. It can be concluded in Table 9 that the interactions of $popsize * hp$, $lr * hp$, $hp * \alpha$ are weak, and this finding is further confirmed by the interaction effects of parameter pairs in Fig. 13. Therefore, the best parameter configurations, derived from the above analysis, are suggested as $popsize = 50$, $lr = 0.1$, $hp = 0.3$, and $\alpha = 15$.

5.3. Evaluation of MILP model

This section provides a comprehensive comparison and evaluation of the MILP model (refer to SubSection 3.1.1 for details). As observed in previous studies, MILP models are typically tested and analyzed based on scale complexity (SC), computational complexity (CC), or both. Regarding SC, the effectiveness of MILP models can be evaluated using multiple indicators, including the number of binary decision variables (NBVs), the number of continuous decision variables (NCVs), and the number of constraints (NCs). In general, NBVs, NCVs, and NCs are proportional to instance sizes. In terms of CC, several aspects of MILP models can be compared, such as the number of optimal or best solutions found in a given time limit, CPU time, and optimality gap, among others [57]. Note that the gap denotes the difference in optimality between the obtained solutions and the optimal solutions within the time limit, which can be calculated as the value of the RPI. Obviously, smaller gap values suggest higher solution quality. The correctness of the MILP model is validated by using the Gurobi solver (version 9.5.0) to tackle 36 different small-scale and medium-scale instances within a time limit of 3600 s. If no optimal solution is found within this timeframe, the best solution found is returned. The computational results of the MILP model for these instances, including NBVs, NCs, and CPU time, are reported in Table 10. It is observed from Table 10 that the Gurobi solver can achieve the best makespan for relatively small-scale instances, ranging from sFG_2_4_2_1_1 to sFG_2_15_2_3_1, and several medium-scale instances are also tested as a comparison. Detailed results for large-scale instances are available upon request. However, for relatively large-scale instances, the MILP model may fail to find any feasible solution within an acceptable CPU time. This is because the Gurobi solver tackles MILP models by default using the branch-and-cut method, which integrates cut-plane and branch-and-bound (B&B) methods, depending mainly on the size of the solved problem. As instance scale increases, the branch-and-cut strategy becomes significantly challenging in dealing with MILP models due to the increased constraints (*i.e.*, NCs), decision variables (*i.e.*, NBVs and NCVs), and enlarged solution spaces, leading to difficulties in branching, seeking new bounds, and cutting subspace. Obviously, it is undeniable that the Gurobi solver can effectively solve small-scale instances to optimality, but it is not efficient in solving relatively large-scale instances. As a basis in comparison to calculate RPI values, EEDA is run independently 10 times for each instance, with the maximum CPU time set to $300 \times m \times S$ milliseconds. With the increased number of families S, setup times for more families and processing times for more jobs have to be considered, so the problem's complexity significantly increases. As demonstrated in Table 10, NBVs, NCs, and CPU times of the MILP model notably increase as problem complexity escalates, while solution quality does not significantly improve. Nevertheless, when compared with results from EEDA and other metaheuristics (*i.e.*, TIG_{v1} [24], TIG_{v2} [49], CCEA [3], and DIWO [68]), as shown in Table 13, although the MILP model requires more CPU time to solve the problem at hand, the RPI values of the solutions obtained by the MILP model are much larger. With the scale increases, EEDA achieves more satisfactory results than the MILP solver, and the runtime of the metaheuristic is quite lower than that of the MILP solver. This implies that the proposed EEDA outperform Gurobi solver, making it easier to obtain high-quality solutions compared to several state-of-the-art MILP solvers, thereby rendering EEDA more suitable for solving DNFGSP_SDSTs.

Table 8
The levels of parameters.

Parameters	Factor level			
	1	2	3	4
popsize	30	50	70	90
lr	0.1	0.3	0.5	0.7
hp	0.1	0.2	0.3	0.4
α	5	10	15	20

Table 9
Results of ANOVA for parameters calibration.

Sources	Sum of squares	Degree of freedom	Mean square	F-ratio	p-value
popsize	0.00013590	3	0.00004530	433.72	**0.0000**
lr	0.00004892	3	0.00001631	156.11	**0.0000**
hp	0.00000731	3	0.00000244	23.33	**0.0000**
α	0.00000289	3	0.00000096	9.24	**0.0000**
$popsize * lr$	0.00001085	9	0.00000121	11.54	**0.0000**
$popsize * hp$	0.00000045	9	0.00000005	0.48	0.8866
$popsize * \alpha$	0.00000459	9	0.00000051	4.88	**0.0000**
$lr * hp$	0.00000012	9	0.00000001	0.12	0.9991
$lr * \alpha$	0.00000446	9	0.00000050	4.75	**0.0000**
$hp * \alpha$	0.00000027	9	0.00000003	0.29	0.9777
Residual	0.00001964	189	0.00000010		
Total	0.00023559	255			

Note: All F-ratios are based on the residual mean square error.

5.4. Effectiveness of key components in EEDA

As detailed in the previous sections, EEDA comprises three pivotal components: (1) speed-up evaluation methods developed in Section 3.2; (2) family-based updating mechanism designed in SubSection 4.2.2; (3)

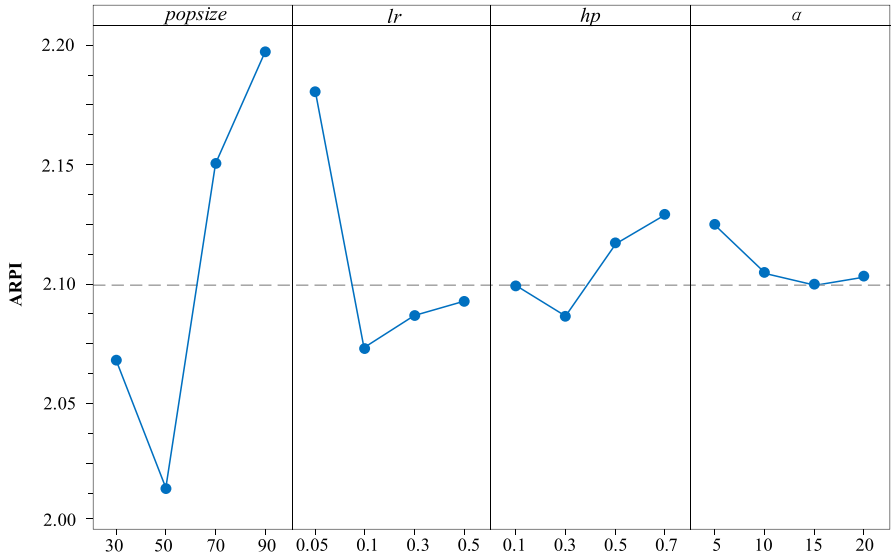


Fig. 12. Main effect plots of parameters.

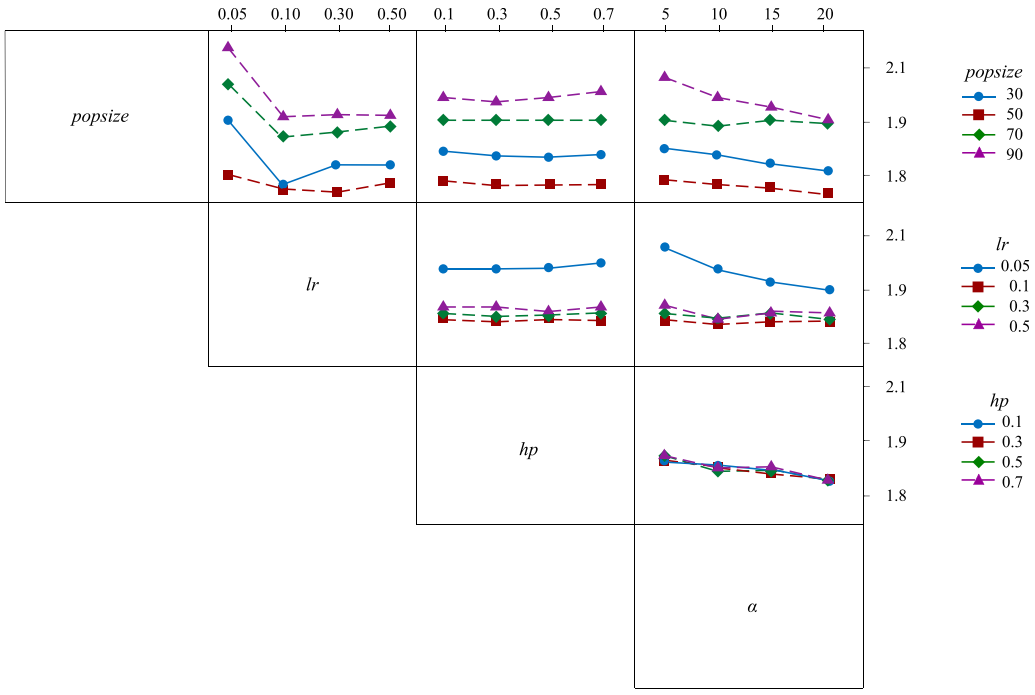


Fig. 13. Interaction effect plots of parameter pairs.

perturbation and reinitialization methods detailed in Section 4.4; (4) initialization method in Section 4.1. To investigate the efficacy of these components within EEDA, four variants, namely EEDA_{v1}, EEDA_{v2}, EEDA_{v3}, and EEDA_{v4} are created by separately eliminating the effect of each component and then compared with EEDA. To be specific, EEDA_{v1} is derived by excluding speed-up evaluation methods from EEDA. EEDA_{v2} is generated by substituting the probability model updating mechanism of EEDA with an alternative updating mechanism that only records the connected jobs between two different families in RM^G. EEDA_{v3} is created by removing both the perturbation and reinitialization methods from EEDA. EEDA_{v4} is generated from EEDA by using a randomly generated population to substitute the population generated by the initialization method. Each variant is altered in only one component that differs from the EEDA, and their key parameters are fine-tuned to achieve the best behavior. For EEDA and its three variants, their performance is evaluated by the solving of all 810 instances, each of which is executed independently 10 times, with the maximum elapsed CPU time of $\rho \times m \times S$ milliseconds, where $\rho = \{100, 200, 300\}$. The statistical results are reported in Table 11, grouped by F, S, and m, respectively. The best values of each group are highlighted in bold, the second-best values are marked in bold italics, and the third-best values are underlined. To visually evaluate the performance of the key components in EEDA, Fig. 14 shows the means plots of EEDA and its variants with 95% Tukey's HSD confidence intervals.

According to the findings from Table 11 and Fig. 14, the following conclusions can be drawn: (i) EEDA exhibits significantly superior performance compared to its three variants across all factors, *i.e.* F, S, and m. This indicates the effectiveness of the proposed improvement strategies in enhancing the effectiveness of EEDA. (ii) From Fig. 14, it is evident that the gap between EEDA and EEDA_{v3} is the most pronounced,

Table 10
NBVs, NCs, CPU times, and RPI values for MILP model.

Instances	NBVs	NCs	RPI	Time(s)	Instances	NBVs	NCs	RPI	Time(s)
sFG_2_4_2_1_1	230	1767	0	1.34	FG_2_20_2_1_1	8076	4.94E+20	11.04	3600
sFG_2_4_2_2_1	202	8560	0	4.41	FG_2_20_2_1_2	13,076	8.20E+21	15.47	3600
sFG_2_4_2_3_1	260	33,404	0	2.84	FG_2_20_2_1_3	16,170	1.15E+27	19.82	3600
sFG_2_6_2_1_1	384	2.63E+05	0	1.56	FG_2_20_2_1_4	10,520	1.43E+20	13.66	3600
sFG_2_6_2_2_1	314	66,221	0	7.26	FG_2_20_2_1_5	10,722	2.85E+20	12.57	3600
sFG_2_6_2_3_1	422	5.36E+05	0	20.44	FG_2_20_2_2_1	16,422	2.37E+27	15.22	3600
sFG_2_8_2_1_1	534	2.10E+06	0	32.51	FG_2_20_2_2_2	12,090	4.95E+20	13.45	3600
sFG_2_8_2_2_1	828	1.34E+08	0	27.42	FG_2_20_2_2_3	11,778	2.47E+20	13.74	3600
sFG_2_8_2_3_1	378	1.32E+05	0	11.86	FG_2_20_2_2_4	7225	1.21E+19	11.22	3600
sFG_2_10_2_1_1	1232	8.59E+09	1.67	3600	FG_2_20_2_2_5	15,690	8.20E+21	20.63	3600
sFG_2_10_2_2_1	1102	2.15E+09	0	3600	FG_2_20_2_3_1	20,691	9.10E+24	25.34	3600
sFG_2_10_2_3_1	2272	7.04E+13	5.36	3600	FG_2_20_2_3_2	12,786	1.41E+20	14.64	3600
sFG_2_12_2_1_1	1416	3.44E+10	0	3600	FG_2_20_2_3_3	13,206	3.19E+20	15.13	3600
sFG_2_12_2_2_1	1562	1.37E+11	0	3600	FG_2_20_2_3_4	13,955	6.93E+20	14.62	3600
sFG_2_12_2_3_1	1488	6.87E+10	0	3600	FG_2_20_2_3_5	12,397	4.95E+20	12.67	3600
sFG_2_15_2_1_1	2310	3.52E+13	4.76	3600	FG_2_40_2_1_1	40,440	1.15E+27	34.22	3600
sFG_2_15_2_2_1	2132	8.80E+12	2.37	3600	FG_2_40_2_1_2	47,430	3.37E+27	31.21	3600
sFG_2_15_2_3_1	1572	6.87E+10	0	3600	FG_2_40_2_1_3	52,010	5.58E+27	37.56	3600

indicating that the perturbation and reinitialization methods can exert the most substantial influence on the diversity and vitality of the search behavior within EEDA. (iii) As shown in Table 11, in certain cases of $\rho = 300$, the results of $EEDA_{v3}$ surpass those of $EEDA_{v1}$ and $EEDA_{v2}$, which reflects the efficiency and effectiveness of the speed-up evaluation method and the family-based updating mechanism, respectively. This implies that when ρ is large enough, the possibility of being trapped into the local optimum is thus reduced. (iiii) It can be observed from Table 11 that $EEDA_{v4}$ obtains its best results under $\rho = 300$, but still slightly behind EEDA. In Fig. 14, the length of the confidence interval for $EEDA_{v4}$ is similar to that for EEDA. The performance of $EEDA_{v4}$ indicates that the proposed initialization method can provide a high-quality and diverse initial population for EEDA, thus effectively avoiding invalid searches during the initial phase. Based on these results and analyses, it can be concluded that speed-up evaluation methods, family-based updating mechanism, perturbation and reinitialization method, and initialization method are all effective components of EEDA.

5.5. Comparison between different state-of-the-art algorithms

To the best of our knowledge, the DFGSP with both no-wait and SDSTs constraints has not yet been studied so far, and there are no existing comparative algorithms. To validate the efficacy of EEDA, in this section, we conduct a comparative analysis with existing state-of-the-art algorithms, including *i.e.*, TIG_{v1} [24], TIG_{v2} [49], CIG [67], CCEA [3], $DIWO_{v1}$, $DIWO_{v2}$, $DIWO_{v3}$ [68], CCABC [69], DABC [70], and IABC [55]. These algorithms are employed to solve other scheduling problems, which consist of minimizing the makespan of DFGSP, DFSP with no-wait constraint, and DFSP with SDSTs. To be specific, both TIG_{v1} and TIG_{v2} achieved remarkable results in tackling DFGSPs using the two-stage strategy, while CIG, CCEA, and CCABC adopted cooperative co-evolutionary strategies to yield satisfactory solutions in decomposing and solving subproblems of DFGSPs. DIWO and ABC, well known for their effectiveness in addressing DNFSPs, are also chosen as comparison algorithms. It's worth noting that IABC is an improved ABC algorithm with a destruction-construction mechanism proposed for solving DFSP, demonstrating considerable reference value. Moreover, typical algorithms for DFSPs may be effective for dealing with their extensions. Since DNFGSP_SDSTs is a generalization of DFSP, selecting several sophisticated algorithms to tackle the extension of DFSP holds important interest. The differences between $DIWO_{v1}$, $DIWO_{v2}$, and $DIWO_{v3}$ lie in their search strategies. $DIWO_{v1}$ employs a two-phase strategy, where the first stage focuses on searching for suitable family sequences, and the second stage aims at adjusting job sequences; $DIWO_{v2}$ directly searches for both family sequences and job sequences; and $DIWO_{v3}$ randomly determines the search sequences during each iteration. For a fair comparison, all algorithms are adapted to problem-specific characteristics, and the same decoding and population initialization methods are applied. Additionally, speed-up methods proposed in Section 3.2 are incorporated into all algorithms to enhance efficiency. Main parameters were set based on original suggestions, adapted to the problem, as shown in Table 12. Under the same experimental environments, all algorithm were conducted independently 10 times as previously mentioned on all instances, with the maximum elapsed CPU time of $\rho \times m \times S$ milliseconds as the termination criterion, where $\rho = \{100, 200, 300\}$. These statistical results under three termination criteria, grouped by F, S, and m, are reported in Tables 13-15, with the best, second-best, and third-best values marked accordingly. The best value in each group is marked in **bold**, the second one in ***italicized bold***, and the third one is underlined. To verify whether the results obtained in the above tables are statistically significant, Fig. 15 and Fig. 16 show the means plots with 95% Tukey's HSD confidence intervals and boxplots of EEDA and the compared algorithms, respectively.

As reported in Tables 13-15, EEDA achieves the best results in terms of ARPI values in almost all test instances except for several scenarios, remarkably outperforming other algorithms, whereas DABC, IABC, and $DIWO_{v1}$ achieve seemingly decent results in these scenarios. Nevertheless, there is still a considerable gap between the average results of both DABC, IABC, and $DIWO_{v1}$ against EEDA. Further analysis of these statistical results in Tables 13-15 reveals interesting insights and findings. As for $\rho = 100$ in Table 13, EEDA provides the best results in all scenarios, while TIG_{v1}, TIG_{v2}, DABC, and IABC also achieve promising results. This indicates that the convergence performance of these five algorithms is excellent. As shown in Table 14, all the algorithms under $\rho = 200$ obtained better results than those under $\rho = 100$. Although CCEA is still slightly inferior to EEDA, the performance of CCEA has improved significantly. This may be due to the fact that CCEA effectively utilizes the correlations between families and jobs. With the increase of the termination factor ρ, each algorithm achieves the best results when $\rho = 300$, and the average ARPI values for all algorithms in all instances are as follows: EEDA (1.95), TIG_{v1} (4.37), TIG_{v2} (4.19), CIG (4.56), CCEA (5.06), $DIWO_{v1}$ (5.16), $DIWO_{v2}$ (7.27), $DIWO_{v3}$ (6.57), CCABC (9.30), DABC (2.96), and IABC (3.84). It is clear that these ARPI averages are enough to emphasize the excellent performance of EEDA, while CCABC yields the worst results. As can be observed from Tables 13-15, the performance of TIG_{v1}, TIG_{v2}, CIG, and IABC in solving small-scale instances is remarkably better than that in solving large-scale instances, which may be because the large-scale instances are subjected to their larger solution spaces, and the destruction-construction on the only single solution may limit the search scope in the solution space. Unlike IG variants, IABC also gets outstanding performance under large-scale instances, which can be attributed to the search mechanism of ABC.

Table 11
The ARPI values of EEDA, $EEDA_{v1}$, $EEDA_{v2}$, $EEDA_{v3}$, $EEDA_{v4}$.

ρ	Alg.	F						S			m			Mean
		2	3	4	5	6	7	20	40	60	2	4	6	
100	EEDA	**1.71**	**1.94**	**2.10**	**2.23**	**2.44**	**2.75**	**2.03**	**2.12**	**2.43**	**2.63**	**2.14**	**1.81**	**2.19**
	$EEDA_{v1}$	2.09	2.35	2.41	***2.44***	***2.61***	2.80	2.29	2.47	2.59	2.72	2.40	2.22	2.45
	$EEDA_{v2}$	1.88	2.38	2.49	2.51	2.64	***2.78***	2.32	2.46	***2.56***	***2.67***	2.36	2.31	2.44
	$EEDA_{v3}$	2.54	2.59	2.67	2.71	2.75	2.84	2.57	2.70	2.78	2.78	2.69	2.58	2.68
	$EEDA_{v4}$	***1.92***	***2.21***	***2.40***	2.57	2.63	2.84	***2.16***	***2.38***	2.73	2.91	***2.32***	***2.04***	***2.43***
200	EEDA	**1.51**	**1.73**	**1.92**	**2.11**	**2.32**	**2.63**	**1.89**	**1.98**	**2.24**	**2.45**	**2.03**	**1.64**	**2.04**
	$EEDA_{v1}$	1.76	1.98	2.21	***2.32***	***2.57***	3.03	2.18	***2.27***	2.48	2.62	***2.24***	2.08	2.31
	$EEDA_{v2}$	1.91	2.14	***2.02***	2.47	2.62	2.86	2.24	2.33	2.44	***2.57***	2.31	2.13	2.34
	$EEDA_{v3}$	2.27	2.32	2.47	2.49	2.60	***2.67***	2.38	2.47	2.56	2.58	2.48	2.35	2.47
	$EEDA_{v4}$	***1.65***	***1.90***	2.12	2.34	***2.57***	2.93	***2.08***	2.33	***2.35***	2.69	2.25	***1.81***	***2.25***
300	EEDA	**1.44**	**1.66**	**1.77**	**2.06**	**2.29**	**2.37**	**1.76**	**1.89**	**2.16**	**2.31**	**1.93**	**1.57**	**1.93**
	$EEDA_{v1}$	1.64	1.79	1.95	2.24	2.36	2.53	1.92	2.06	2.27	2.36	2.02	1.87	2.08
	$EEDA_{v2}$	1.91	1.76	***1.83***	2.36	2.35	***2.39***	1.97	2.10	2.23	2.42	2.05	1.83	2.11
	$EEDA_{v3}$	2.07	2.13	2.16	2.24	***2.29***	2.42	2.06	2.23	2.37	***2.32***	2.23	2.12	2.23
	$EEDA_{v4}$	***1.50***	***1.74***	1.86	***2.09***	2.34	2.43	***1.78***	***1.98***	***2.21***	2.39	***1.97***	***1.62***	***1.99***

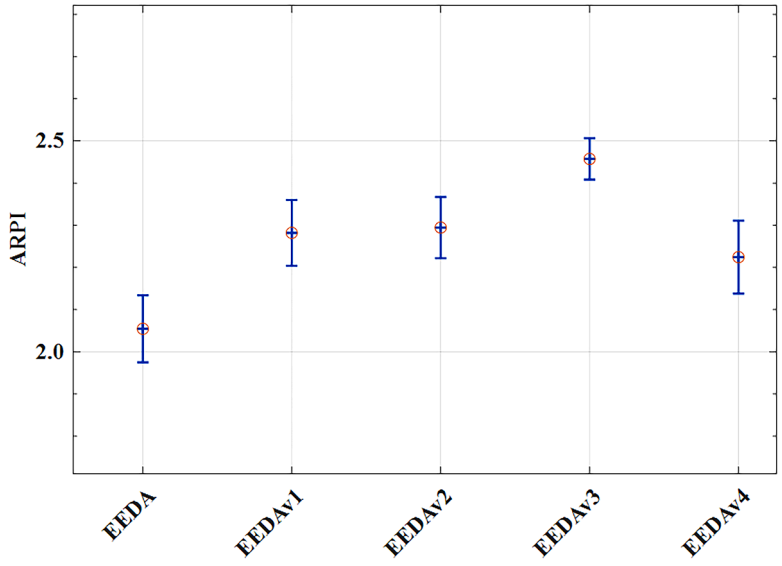


Fig. 14. Means plots with 95% Tukey's HSD confidence intervals of EEDA and its variants.

For the variants of ABC, DABC achieves better results overall than other variants due to the improved search mechanism in the employed bee and scout bee phases, and IABC achieves better results in small-scale instances by adding the destruction-construction mechanism, while CCABC performs worse due to its failure to be properly based on the problem characteristics. As illustrated in Figs. 15 and 16, TIG_{v1}, TIG_{v2}, and IABC perform reasonably well, and CIG excels within certain contexts, but its performance slightly lags behind that of the other IGs. The IG's search behavior suggests that TIG_{v1}, TIG_{v2}, IABC, and CIG can perform deeper exploration of the solution space. Moreover, the results yielded by $DIWO_{v1}$ are all the best among the variants of DIWO, which demonstrates that the two-stage search strategy of $DIWO_{v1}$ is effective in solving DNFGSP_SDSTs.

Although the superiority of EEDA was analyzed above, the interaction effect was still worth further investigation. Interaction effect analysis, as a commonly used statistical analysis method, is useful in revealing the interaction between factors and their impact on results. In this study, the test instances are mainly composed of three factors, including the number of cellulars (F), families (S), and machines (m), each tested on three termination factors (ρ). To elucidate how these factors interact with the performance of EEDA, a multifactor ANOVA was used to clarify some interesting findings. Figs. 17 and 18 depict interaction plots and box plots of all the algorithms with different factors, respectively. From these figures, it is evident that changes in any of the factors have slight effects on the efficacy of EEDA compared to other competitors, suggesting that EEDA is stable under different scenarios. Furthermore, the ARPI values yielded by EEDA are significantly lower than those of others, which also indicates the superior performance of EEDA. It is clear from Fig. 17(a) that each algorithm's performance improves as the termination time increases, while EEDA significantly outperforms all algorithms across all termination factors. In Fig. 17(a), the ranking of algorithms at $\rho = 300$, from bottom to top, is

Table 12
The parameters of the compared algorithms.

Algorithm	Parameter
TIG_{v1} [24]	$pv = 0.7$, $tv = 0.9$
TIG_{v2} [54]	$FamD = 6, T^{first} = 0.8$, $T_0 = 0.6$
CIG [67]	$FamD = 8, FamIterator = 5$, $JobIterator = 1$
CCEA [3]	$AS = 5, PS = 10, \alpha = 5, d = 6$, $\xi = 1.0$
$DIWO_{v1}$ [68]	$PS_0 = 10, PS_{max} = 50, \delta = 5, S_{max} = 20, \rho = 0.9$, $S_{min} = 1$
$DIWO_{v2}$ [68]	$PS_0 = 10, PS_{max} = 50, \delta = 5, S_{max} = 20$, $S_{min} = 1$
$DIWO_{v3}$ [68]	$PS_0 = 10, PS_{max} = 50, \delta = 5, S_{max} = 20, \alpha = 0.9$, $S_{min} = 1$
CCABC [69]	$\theta^{'} = 50, PS = 10$
DABC [70]	$popsize = 30, SN = 8$, $TN = 5$
IABC [55]	$PS = 30, d = 5, \lambda = 70$

Table 13
The ARPI values of EEDA and compared algorithms with $\rho = 100$.

ρ =100	Instance		EEDA	TIG1	TIG2	CIG	CCEA	DIWO1	DIWO2	DIWO3	CCABC	DABC	IABC
$F = 2$	$m = 2$	$S = 20$	**1.69**	2.73	***2.64***	4.09	5.93	7.26	9.31	7.29	9.96	4.21	3.54
		$S = 40$	**1.42**	2.42	***2.38***	3.43	4.74	5.99	7.74	6.58	10.17	3.69	3.70
		$S = 60$	**1.24**	***1.74***	2.10	2.83	4.57	5.71	6.07	6.34	9.16	3.42	3.90
	$m = 4$	$S = 20$	**2.24**	2.76	***2.64***	5.36	6.23	5.65	9.12	7.50	9.99	4.43	2.91
		$S = 40$	**1.55**	3.16	***2.90***	4.04	4.72	4.84	8.21	6.77	8.10	3.55	3.22
		$S = 60$	**1.33**	2.59	***2.46***	2.49	4.45	4.51	7.62	6.26	7.49	3.29	3.65
	$m = 6$	$S = 20$	**2.42**	3.68	3.43	5.65	6.40	6.37	7.35	7.39	11.15	3.27	***2.53***
		$S = 40$	**1.94**	3.98	3.76	4.56	5.76	4.68	6.83	5.77	9.20	***2.39***	2.79
		$S = 60$	**1.53**	3.54	3.35	4.23	5.26	4.31	6.01	5.05	7.19	***1.98***	3.17
	Mean		**1.71**	2.96	***2.85***	4.08	5.34	5.48	7.59	6.55	9.16	3.36	3.27
$F = 3$	$m = 2$	$S = 20$	**2.32**	2.96	***2.52***	5.04	6.49	6.94	10.04	6.99	9.67	4.94	3.93
		$S = 40$	**1.51**	2.51	***2.31***	3.70	5.31	5.56	9.19	6.42	8.83	4.02	3.98
		$S = 60$	**1.34**	2.72	***2.23***	2.51	4.84	4.98	7.00	5.85	7.52	3.59	5.08
	$m = 4$	$S = 20$	**2.05**	3.83	3.43	6.42	6.00	7.17	9.65	9.71	10.98	3.83	***3.17***
		$S = 40$	**1.88**	3.73	***3.51***	4.92	5.14	6.56	7.83	7.11	8.97	3.52	3.65
		$S = 60$	**1.64**	3.44	3.19	2.99	4.59	5.89	7.51	6.92	8.92	***2.47***	4.41
	$m = 6$	$S = 20$	**2.74**	5.30	5.41	5.84	6.55	5.17	8.41	9.50	11.03	3.81	***2.96***
		$S = 40$	**2.13**	4.67	4.86	5.98	6.21	4.93	7.37	6.26	10.02	***2.92***	3.04
		$S = 60$	**1.84**	4.96	4.70	5.31	5.62	4.20	5.74	4.81	9.11	***2.43***	4.15
	Mean		**1.94**	3.79	3.57	4.74	5.64	5.71	8.08	7.06	9.45	***3.50***	3.82
$F = 4$	$m = 2$	$S = 20$	**2.55**	4.49	***3.92***	4.81	6.34	8.20	10.61	9.64	10.31	4.91	4.69
		$S = 40$	**1.86**	***4.13***	4.24	4.47	5.05	5.67	8.61	8.21	10.04	4.24	4.17
		$S = 60$	**1.44**	3.66	3.69	4.28	4.72	5.10	7.58	7.73	8.76	***3.31***	6.34
	$m = 4$	$S = 20$	**2.28**	4.24	4.36	7.34	6.39	7.28	10.22	7.75	10.58	***2.79***	4.14
		$S = 40$	**2.11**	4.79	4.81	5.69	5.02	5.67	8.41	6.30	10.75	***2.69***	4.04
		$S = 60$	**1.74**	3.93	4.17	4.91	4.47	5.00	7.23	5.27	10.13	***2.44***	6.04
	$m = 6$	$S = 20$	**2.84**	6.48	6.96	6.76	7.38	6.43	8.98	9.84	9.37	4.83	***3.51***
		$S = 40$	**2.19**	6.84	7.25	5.75	6.49	5.35	7.95	6.60	8.82	3.96	***3.41***
		$S = 60$	**1.93**	5.52	5.62	5.08	6.10	4.61	6.31	5.15	6.95	***3.31***	4.94
	Mean		**2.10**	4.90	5.00	5.45	5.77	5.92	8.43	7.39	9.52	***3.61***	4.59
$F = 5$	$m = 2$	$S = 20$	**2.64**	***4.43***	4.49	5.54	4.93	7.56	9.76	9.04	10.13	5.43	4.59
		$S = 40$	**1.91**	4.04	5.20	5.17	3.60	7.10	9.21	8.49	9.91	***3.22***	5.26
		$S = 60$	**1.64**	4.95	4.46	3.93	***2.95***	5.95	8.29	7.06	8.84	3.12	6.57
	$m = 4$	$S = 20$	**2.39**	4.91	4.76	6.35	8.43	7.33	8.98	8.72	11.30	***3.62***	3.70
		$S = 40$	**2.24**	5.35	5.53	6.46	5.85	5.60	9.95	8.12	9.03	***2.78***	4.65
		$S = 60$	**1.83**	5.25	4.88	6.05	4.75	5.19	7.60	7.20	8.30	***2.68***	6.20
	$m = 6$	$S = 20$	**2.94**	6.99	7.14	7.38	7.97	7.47	8.80	9.45	11.53	4.39	***3.54***
		$S = 40$	**2.29**	6.86	6.71	7.08	7.69	6.45	8.85	7.57	10.35	***3.87***	4.11
		$S = 60$	**2.21**	6.32	6.47	6.78	5.71	4.98	6.91	4.67	7.93	***3.26***	5.27
	Mean		**2.23**	5.46	5.51	6.08	5.76	6.40	8.71	7.81	9.70	***3.60***	4.87
$F = 6$	$m = 2$	$S = 20$	**2.78**	5.49	5.89	6.16	***5.28***	8.09	10.45	9.22	10.43	5.38	5.35
		$S = 40$	**2.31**	5.87	5.84	5.54	***3.86***	7.60	9.66	8.44	9.85	4.39	6.16
		$S = 60$	**1.79**	5.60	5.74	4.21	***3.17***	6.37	8.88	7.12	9.57	3.73	7.24
	$m = 4$	$S = 20$	**2.68**	6.12	6.08	6.79	8.61	7.85	9.62	9.33	11.37	***4.19***	4.77
		$S = 40$	**2.50**	5.79	5.99	6.91	6.26	6.01	10.65	8.69	10.78	***3.87***	5.31
		$S = 60$	**1.94**	5.67	5.64	6.47	5.08	5.57	8.15	7.71	10.49	***3.41***	6.03
	$m = 6$	$S = 20$	**3.09**	7.59	8.32	7.89	8.52	7.99	9.42	10.11	11.87	4.47	***4.30***
		$S = 40$	**2.61**	6.66	6.92	7.50	7.07	6.91	9.49	8.10	11.26	***3.87***	4.73
		$S = 60$	**2.33**	7.15	7.35	7.26	6.11	5.34	8.08	7.31	10.97	***3.41***	5.90
	Mean		**2.45**	6.22	6.42	6.53	6.00	6.86	9.38	8.45	10.73	***4.08***	5.53
$F = 7$	$m = 2$	$S = 20$	**3.22**	6.07	6.03	6.48	5.56	8.52	10.59	9.71	11.37	***5.44***	5.64
		$S = 40$	**2.72**	7.48	7.26	5.83	4.64	8.01	10.17	8.89	10.78	***4.37***	6.83
		$S = 60$	**2.23**	6.48	6.62	4.44	***3.34***	6.72	9.36	7.50	10.49	3.84	7.04
	$m = 4$	$S = 20$	**3.24**	7.19	7.14	7.14	8.66	8.26	10.13	9.82	12.33	***4.42***	5.82
		$S = 40$	**2.64**	6.94	7.34	7.27	6.93	6.33	10.06	9.15	11.72	***3.79***	6.17
		$S = 60$	**2.01**	7.07	7.47	6.81	5.35	6.10	8.59	8.12	11.43	***3.26***	5.82
	$m = 6$	$S = 20$	**3.36**	7.73	7.47	8.30	8.66	8.42	9.93	10.64	12.84	***4.30***	4.86
		$S = 40$	**2.76**	6.45	6.12	7.89	7.44	7.28	9.99	8.53	12.22	***3.73***	5.65
		$S = 60$	**2.60**	7.34	6.64	7.63	6.43	5.64	8.76	7.70	11.92	***3.36***	5.83
	Mean		**2.75**	6.97	6.90	6.87	6.33	7.25	9.73	8.89	11.68	***4.06***	5.96

as follows: EEDA, DABC, IABC, TIG_{v2}, TIG_{v1}, CIG, CCEA, $DIWO_{v1}$, $DIWO_{v3}$, $DIWO_{v2}$, and CCABC. From Fig. 17(*b*), the curve of EEDA has the smallest impact with the increase of F, whereas TIG_{v1} and TIG_{v2} are relatively more influenced. The overlapping intervals of TIG_{v1} and TIG_{v2} indicate that their performance is comparable in the statistical sense. With smaller F, the curves of TIGs are close to those of both DABC and IABC but higher than EEDA. As cellular F increases, CIG and CCEA gradually catch up, while $DIWO_{v1}$ improves significantly. Notably, $DIWO_{v1}$ and CCEA exhibit overlapping in their intervals, while CCEA also shows better performance when F becomes larger. As shown in Fig. 17(*c*), DABC and the three variants of DIWO are less affected by S, improving solution quality with increasing termination time. Fig. 17(*d*) demonstrates that all algorithms perform better with increasing m, likely due to the fact that the CC is not greatly affected by m. As can be observed in Figs. 17 and 18, EEDA consistently outperforms other algorithms across all the factors. For most of the cases in Fig. 18, although the box plots of TIG_{v1} and TIG_{v2} have lower bounds, their box lengths are significantly longer than those of the others, and the box length of CIG is also longer. This suggests that although the desired results can be yielded via single soluation iterations under the effect of certain factors, overall the quality of the solutions is not stable. In contrast, despite the higher lower bound of CCEA, its box length remains short in most cases,

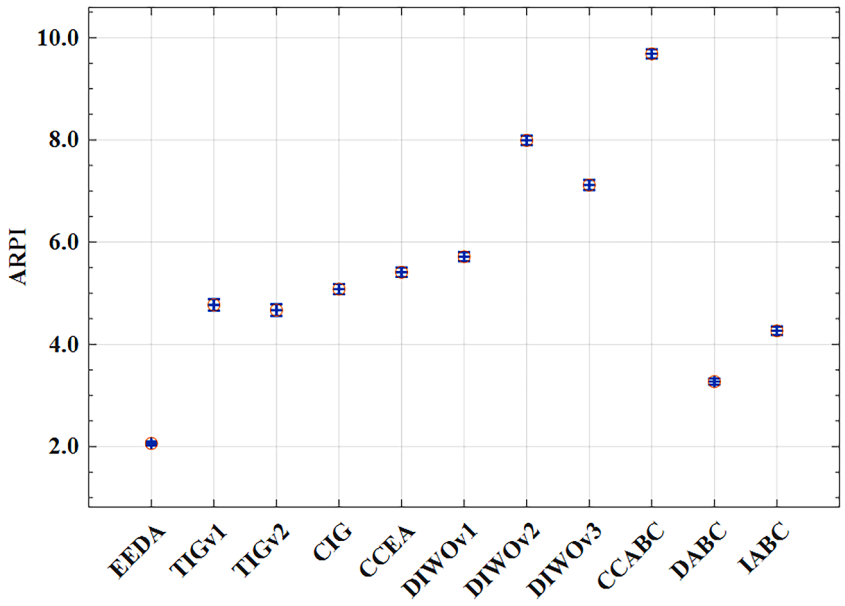


Fig. 15. Means plots with 95% Tukey's HSD confidence intervals of all algorithms.

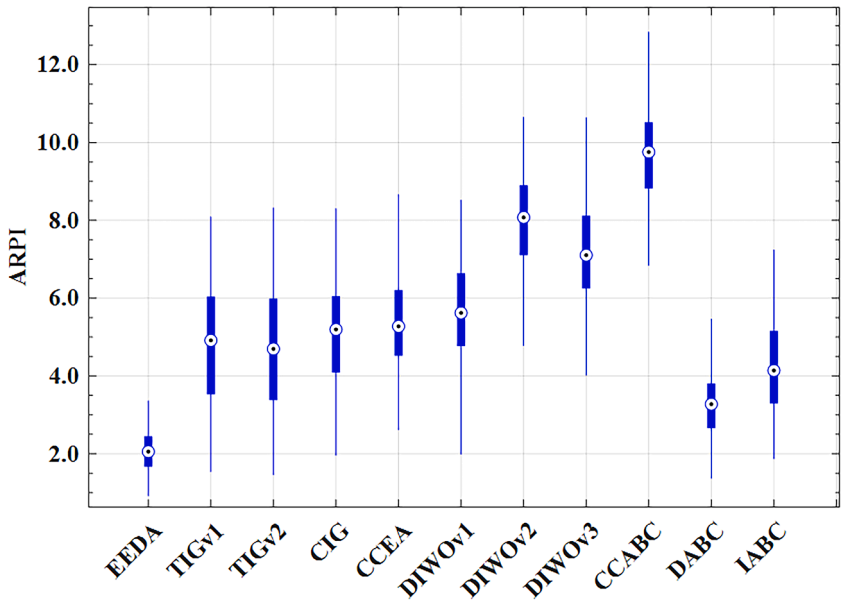


Fig. 16. Box plots for all the algorithms.

showing significant stability of the cooperative coevolutionary strategy. Notably, the box lengths of the EEDA are the shortest, indicating its superior stability. Finally, Fig. 19 depicts the Gantt chart of the best solution found by EEDA for FG_2_20_6_1_1, with the makespan of 582.

5.6. Discussion

Through the conducted experiments, the effectiveness of crucial components in EEDA and the efficacy of EEDA have been systematically examined and evaluated. In comparison to state-of-the-art algorithms employed in the experiments, the proposed EEDA consistently demonstrates stable and efficient performance across instances of different scales. The superiority of EEDA can be concluded as follows. (1) Unlike the destruction-construction mechanism of IG, EEDA strategically learns valuable information hidden in high-quality solutions through a problem-specific probability model. This approach circumvents the inadvertent destruction of promising patterns extracted from high-quality solutions, contributing to the algorithm's efficacy. (2) The experimental results of CCABC are the least favorable among all compared algorithms, while DABC and IABC exhibit remarkable effectiveness. Notably, the distinction between CCABC, IABC, and DABC lies in the employed bee phase and scout bee phase. In the employed bee phase, DABC intelligently switches between searching the current neighborhood and moving on to the next if a superior solution is not found. The scout bee phase employs a VND-based method, effectively executing neighborhood searches. These well-developed search strategies in DABC can reduce blind exploration, enhancing its overall effectiveness. Compared to DABC, IABC gains better results in small-scale instances by introducing the destruction-construction mechanism and weakening the employed bee phase. As for solving large-scale instances, DABC outperforms IABC due to the limited times of destruction and construction of both family and job sequences. For CCABC, both

Table 14
The ARPI values of EEDA and compared algorithms with $\rho = 200$.

ρ =200	Instance		EEDA	TIG1	TIG2	CIG	CCEA	DIWO1	DIWO2	DIWO3	CCABC	DABC	IABC
$F = 2$	$m = 2$	$S = 20$	**1.20**	2.73	***2.64***	3.82	5.41	6.22	8.63	7.24	9.64	4.77	3.56
		$S = 40$	**2.13**	2.42	***2.31***	4.92	5.67	5.32	8.46	6.92	9.67	3.61	3.13
		$S = 60$	**2.23**	3.09	3.24	5.17	5.82	4.75	6.92	6.83	10.67	***2.45***	3.29
	$m = 4$	$S = 20$	**1.36**	***2.42***	2.46	3.24	4.37	5.14	7.26	6.12	9.82	3.96	2.91
		$S = 40$	**1.53**	3.06	2.52	3.77	4.36	4.62	7.67	6.29	8.02	***2.42***	2.68
		$S = 60$	***1.72***	3.54	3.44	4.23	5.26	3.48	6.47	5.42	8.98	**1.64**	3.09
	$m = 6$	$S = 20$	**1.05**	***1.74***	1.83	2.72	4.23	4.37	5.81	5.91	8.94	2.93	2.23
		$S = 40$	**1.06**	2.53	2.15	2.43	4.12	4.33	7.16	5.84	7.49	***1.83***	2.20
		$S = 60$	**1.36**	3.14	3.09	3.94	4.83	3.16	5.76	4.79	7.23	***1.37***	2.69
	Mean		**1.52**	2.74	***2.63***	3.80	4.90	4.60	7.13	6.15	8.94	2.78	2.86
$F = 3$	$m = 2$	$S = 20$	**2.05**	***2.56***	2.63	4.64	5.62	6.44	9.26	6.91	9.38	4.94	3.61
		$S = 40$	**1.93**	3.53	***3.42***	5.84	5.84	6.64	8.92	7.82	10.52	3.70	3.82
		$S = 60$	**2.51**	5.21	5.14	5.34	6.24	4.90	7.84	7.21	10.57	***2.66***	4.58
	$m = 4$	$S = 20$	**1.43**	***2.37***	2.44	3.48	4.64	5.24	8.52	6.46	8.65	4.02	2.98
		$S = 40$	**1.72**	3.53	3.49	4.54	4.72	5.24	7.34	6.58	8.77	***2.72***	3.25
		$S = 60$	***1.94***	4.56	4.66	5.46	5.32	3.96	6.94	6.04	9.68	**1.71**	3.75
	$m = 6$	$S = 20$	**1.27**	2.68	***2.37***	2.44	4.45	4.74	6.62	6.31	7.51	3.59	2.88
		$S = 40$	**1.19**	3.14	3.21	2.86	4.24	4.66	6.32	5.94	8.73	***1.91***	2.71
		$S = 60$	***1.57***	4.94	3.91	4.88	5.12	3.32	5.92	5.18	8.89	**1.51**	3.54
	Mean		**1.73**	3.61	3.47	4.39	5.13	5.02	7.52	6.49	9.19	***2.97***	3.46
$F = 4$	$m = 2$	$S = 20$	**2.24**	***3.68***	3.84	5.26	5.72	7.54	9.76	8.78	9.94	4.32	4.62
		$S = 40$	**2.21**	4.05	4.23	6.24	5.94	6.74	9.42	7.14	10.17	***2.96***	***2.96***
		$S = 60$	**2.67**	6.54	6.28	5.64	6.54	6.00	8.34	8.96	9.12	***3.63***	3.86
	$m = 4$	$S = 20$	**1.72**	4.05	4.12	4.42	4.87	5.34	8.02	7.54	9.71	***3.91***	4.24
		$S = 40$	**1.95**	4.59	4.62	4.68	4.52	5.34	7.84	5.88	10.32	***2.09***	3.64
		$S = 60$	**2.05**	6.79	5.84	5.26	5.42	5.06	7.44	6.14	8.64	***2.71***	5.63
	$m = 6$	$S = 20$	**1.41**	3.74	3.64	2.91	4.56	4.84	7.12	7.12	8.59	***2.49***	3.42
		$S = 40$	**1.36**	3.98	4.06	3.47	4.34	4.76	6.82	4.98	9.78	***1.84***	2.86
		$S = 60$	**1.68**	5.48	5.32	4.68	5.56	4.42	6.02	4.88	7.02	***2.31***	***2.31***
	Mean		**1.92**	4.77	4.66	4.73	5.27	5.56	7.86	6.82	9.25	***2.92***	3.73
$F = 5$	$m = 2$	$S = 20$	**2.41**	***4.13***	4.34	5.08	5.94	6.98	9.02	8.26	9.78	5.14	4.66
		$S = 40$	**2.35**	4.73	4.57	5.78	6.28	6.78	8.34	7.98	10.80	***3.47***	4.52
		$S = 60$	**2.78**	6.66	6.64	5.82	6.76	6.90	8.18	8.62	11.00	***3.99***	5.90
	$m = 4$	$S = 20$	**1.84**	4.92	4.96	4.76	4.82	6.58	8.54	7.78	9.59	***3.09***	4.17
		$S = 40$	**2.18**	5.83	5.24	5.88	4.74	5.48	9.18	7.46	8.82	***2.61***	4.35
		$S = 60$	**2.24**	6.38	6.27	5.47	5.71	6.02	8.23	6.98	9.97	***3.59***	5.32
	$m = 6$	$S = 20$	**1.62**	4.37	4.31	3.68	4.72	5.58	7.74	6.54	8.66	***3.09***	3.82
		$S = 40$	**1.67**	5.55	4.68	5.52	4.68	4.92	7.14	6.66	8.19	***1.56***	3.81
		$S = 60$	**1.92**	6.04	6.06	4.92	5.62	4.74	6.54	4.46	7.87	***2.63***	4.88
	Mean		**2.11**	5.40	5.23	5.21	5.47	6.00	8.10	7.19	9.41	***3.24***	4.60
$F = 6$	$m = 2$	$S = 20$	**2.67**	5.42	5.56	5.62	***4.84***	7.45	9.62	8.42	10.05	5.11	5.04
		$S = 40$	**2.64**	5.62	5.72	6.17	8.09	7.23	8.90	8.51	10.86	***3.85***	5.69
		$S = 60$	**2.89**	8.09	7.67	7.13	7.66	7.36	8.73	9.19	11.30	***3.64***	6.73
	$m = 4$	$S = 20$	**2.24**	5.55	5.51	5.08	3.61	7.02	8.93	7.74	9.54	***3.23***	4.20
		$S = 40$	**2.31**	5.69	5.64	6.27	5.70	5.63	9.79	7.96	10.34	***3.73***	4.99
		$S = 60$	**2.46**	6.47	6.45	6.78	6.40	6.42	8.78	7.45	10.77	***3.37***	5.53
	$m = 6$	$S = 20$	**1.81**	5.47	5.43	3.93	***3.01***	5.95	8.26	6.59	9.29	3.22	3.99
		$S = 40$	**1.84**	5.42	5.34	5.89	4.67	5.25	7.62	7.10	10.09	***3.31***	4.44
		$S = 60$	**2.08**	7.03	6.83	6.57	5.57	5.06	7.56	6.76	10.51	***2.67***	5.36
	Mean		**2.33**	6.08	6.02	5.94	5.50	6.37	8.69	7.75	10.31	***3.57***	5.11
$F = 7$	$m = 2$	$S = 20$	**3.01**	5.91	7.42	5.90	***5.08***	7.82	9.74	8.84	10.86	5.46	5.85
		$S = 40$	**3.07**	6.59	6.64	6.47	8.49	7.59	9.34	8.94	11.70	***3.62***	6.26
		$S = 60$	**3.18**	7.73	6.93	7.48	8.04	7.73	9.16	9.65	12.14	***3.46***	6.48
	$m = 4$	$S = 20$	**2.62**	7.55	6.75	5.33	4.29	7.37	9.38	8.13	10.34	***3.47***	5.35
		$S = 40$	**2.53**	6.99	6.82	6.59	6.28	5.91	9.28	8.36	11.17	***3.41***	5.75
		$S = 60$	**2.72**	5.83	5.76	7.12	6.72	6.74	9.22	7.82	11.60	***3.13***	5.18
	$m = 6$	$S = 20$	**2.17**	6.39	6.19	4.12	3.16	6.25	8.67	6.92	10.09	***2.66***	4.34
		$S = 40$	**1.98**	7.12	6.93	6.18	4.91	5.71	8.00	7.46	10.91	***2.94***	5.31
		$S = 60$	**2.44**	6.71	6.21	6.90	5.85	5.31	8.15	7.10	11.34	***2.91***	5.25
	Mean		**2.64**	6.76	6.63	6.23	5.87	6.71	8.99	8.13	11.13	***3.45***	5.53

employed and scout bee phases fail to efficiently search for neighborhoods based on problem characteristics. Similarly, for EEDA, the demonstrated effectiveness of speed-up evaluation methods indicates that these search strategies are efficient for exploring these neighborhoods. (3) Despite incorporating a reinitialization method, CCEA tends to be trapped in local optima. The reason may be that the valuable information retained in the reference of CCEA becomes homogenous after several generations, making it difficult to ensure the quality of newly generated solutions. CCABC may also face similar issues. In contrast, by utilizing family-based updating mechanism and specific sampling strategy, EEDA can capture diverse information from high-quality solutions, generating superior solutions with certain quality and enriching population diversity. Therefore, EEDA can yield better results than both CCEA and CCABC, which commonly employ effective cooperative co-evolutionary frameworks. (4) Among the variations of DIWO, DIWO_{v1} has a better performance than DIWO_{v2} and DIWO_{v3}. DIWO_{v1} employs a two-stage strategy, operating on family sequences and job sequences sequentially. While effective to some extent, there is still a lack of cooperation between the two stages in DIWO_{v1}. The excellent efficacy of EEDA can be attributed to its emphasis on the relationships between families and jobs, which allows global search to be guided by problem-specific knowledge. Consequently, it can be concluded that the

Table 15
The ARPI values of EEDA and compared algorithms with $\rho = 300$.

$\rho = 300$	Instance		EEDA	TIG1	TIG2	CIG	CCEA	DIWO1	DIWO2	DIWO3	CCABC	DABC	IABC
$F = 2$	$m = 2$	$S = 20$	**1.14**	2.39	***2.21***	3.41	5.01	6.04	7.83	6.40	9.24	4.22	3.08
		$S = 40$	**1.19**	2.21	***2.04***	2.88	4.05	5.03	6.57	5.11	9.41	3.47	3.07
		$S = 60$	**0.95**	1.54	***1.46***	2.40	3.92	4.80	5.24	4.53	8.60	2.52	3.17
	$m = 4$	$S = 20$	**1.90**	2.11	***1.91***	4.43	5.25	4.75	7.67	6.49	9.27	4.55	2.51
		$S = 40$	**1.35**	2.70	***2.10***	3.37	4.04	4.11	6.95	5.91	7.76	3.28	2.61
		$S = 60$	**0.92**	2.21	***1.76***	2.14	3.82	3.84	6.48	5.49	7.27	2.64	2.93
	$m = 6$	$S = 20$	***2.06***	***2.72***	2.76	4.66	5.38	5.33	6.26	6.32	10.19	3.30	**1.87**
		$S = 40$	**2.14**	3.14	2.94	3.79	4.87	3.98	5.84	5.75	8.64	2.33	**2.14**
		$S = 60$	**1.32**	2.77	2.62	3.52	4.47	3.69	5.19	5.56	7.03	***1.65***	2.44
	Mean		**1.44**	2.42	***2.20***	3.40	4.53	4.62	6.45	5.73	8.60	3.10	2.65
$F = 3$	$m = 2$	$S = 20$	**1.83**	2.24	***2.20***	4.17	4.21	5.78	8.41	8.09	9.01	4.37	2.86
		$S = 40$	**1.26**	2.14	***2.02***	3.10	3.14	4.68	7.73	5.49	8.34	3.53	3.76
		$S = 60$	**1.11**	2.35	***1.96***	2.14	2.62	4.22	5.98	4.33	7.29	3.13	4.38
	$m = 4$	$S = 20$	**1.82**	3.13	***2.93***	5.27	7.00	5.97	8.10	8.25	10.05	3.23	2.25
		$S = 40$	**1.52**	3.33	2.99	4.08	4.94	4.68	6.64	6.17	8.45	***2.33***	3.02
		$S = 60$	**1.13**	2.77	2.73	2.53	4.06	4.15	4.78	4.43	8.41	***1.58***	3.95
	$m = 6$	$S = 20$	***2.25***	4.67	4.51	4.81	6.64	4.37	7.10	6.08	10.10	2.71	**2.04**
		$S = 40$	**1.98**	4.08	4.07	4.92	6.41	2.58	6.27	4.02	9.29	2.41	***2.34***
		$S = 60$	**1.52**	3.42	3.38	4.39	4.83	***1.99***	4.97	4.37	8.56	2.33	3.12
	Mean		**1.60**	3.12	2.98	3.94	4.87	4.27	6.67	5.69	8.83	***2.85***	3.08
$F = 4$	$m = 2$	$S = 20$	**2.10**	***3.27***	3.31	3.98	4.49	6.80	8.87	8.36	9.52	3.80	3.37
		$S = 40$	**1.52**	3.61	3.57	***2.92***	3.35	4.77	7.27	5.77	9.31	3.42	4.04
		$S = 60$	**1.23**	3.32	3.13	***1.96***	2.80	4.31	6.44	4.61	8.28	2.12	5.57
	$m = 4$	$S = 20$	**2.06**	3.61	3.67	6.01	7.47	6.06	8.56	6.69	9.73	***2.55***	3.11
		$S = 40$	**1.93**	4.10	4.03	3.89	5.27	4.77	7.10	5.53	9.87	***1.96***	3.50
		$S = 60$	**1.27**	3.54	3.52	3.27	4.33	4.24	6.16	4.70	9.38	***1.94***	5.16
	$m = 6$	$S = 20$	***2.46***	5.90	5.75	5.55	7.08	5.38	7.56	8.20	8.77	4.12	**2.40**
		$S = 40$	**1.87**	6.13	5.98	4.74	5.92	4.52	6.73	7.06	8.33	3.32	***2.94***
		$S = 60$	**1.54**	4.92	4.67	4.21	5.15	3.93	5.43	6.67	6.84	***2.91***	4.14
	Mean		**1.78**	4.27	4.18	4.06	5.09	4.98	7.13	6.40	8.89	***2.90***	3.80
$F = 5$	$m = 2$	$S = 20$	**2.26**	***3.68***	3.77	4.57	5.94	6.28	8.19	8.05	9.38	4.56	4.22
		$S = 40$	**1.76**	4.41	4.34	4.28	5.31	5.91	7.75	6.54	9.20	***2.67***	3.98
		$S = 60$	**1.58**	3.90	3.75	3.29	4.13	4.99	7.01	4.22	8.34	***2.67***	5.21
	$m = 4$	$S = 20$	**2.25**	4.23	3.98	5.22	5.86	6.10	7.56	7.46	10.31	***3.02***	3.88
		$S = 40$	**2.03**	5.24	4.60	5.31	4.37	4.72	8.34	6.98	8.49	***2.23***	3.80
		$S = 60$	**1.94**	4.99	4.09	4.98	3.93	4.39	6.46	6.25	7.91	***2.16***	4.91
	$m = 6$	$S = 20$	**2.67**	6.01	5.89	6.05	6.14	6.21	7.42	7.72	10.50	3.50	***3.25***
		$S = 40$	**2.21**	5.75	5.55	5.81	6.03	5.40	7.46	7.28	9.55	***3.13***	3.45
		$S = 60$	**1.88**	5.44	5.36	5.57	5.24	4.22	5.91	6.14	7.62	***2.67***	4.42
	Mean		**2.06**	4.85	4.59	5.01	5.22	5.36	7.34	6.74	9.03	***2.96***	4.12
$F = 6$	$m = 2$	$S = 20$	**2.64**	4.87	4.90	5.07	6.21	6.71	8.74	8.58	9.62	***4.53***	4.92
		$S = 40$	**2.32**	4.99	4.85	4.57	4.30	6.32	8.11	6.97	9.15	***2.80***	4.75
		$S = 60$	**1.76**	4.91	4.78	3.51	4.04	5.34	7.49	6.34	8.93	***2.79***	5.97
	$m = 4$	$S = 20$	**2.45**	5.05	5.04	5.57	6.41	6.51	8.07	7.95	10.37	***3.37***	3.94
		$S = 40$	**2.21**	5.11	4.97	5.67	4.28	5.04	8.90	7.44	9.90	***3.26***	4.13
		$S = 60$	**1.67**	4.87	4.69	5.32	3.84	4.69	6.90	6.66	9.67	***2.87***	4.90
	$m = 6$	$S = 20$	**2.73**	7.32	6.84	6.46	6.97	6.63	7.92	7.87	10.77	***3.18***	3.82
		$S = 40$	**2.27**	5.83	5.71	6.14	5.94	5.77	7.97	7.24	10.28	***2.93***	3.75
		$S = 60$	**1.99**	6.35	6.06	5.95	5.15	4.51	6.85	6.18	10.05	***2.48***	4.79
	Mean		**2.23**	5.48	5.32	5.36	5.24	5.72	7.88	7.25	9.86	***3.13***	4.55
$F = 7$	$m = 2$	$S = 20$	**2.95**	5.32	6.61	5.33	4.71	7.05	8.85	9.00	10.37	***3.78***	4.90
		$S = 40$	**2.45**	6.83	5.99	4.80	3.97	6.64	8.52	7.31	9.90	***2.55***	5.56
		$S = 60$	**2.03**	5.76	5.47	3.69	2.94	5.61	7.87	6.65	9.67	***2.27***	5.86
	$m = 4$	$S = 20$	**3.06**	5.94	5.89	5.86	7.84	6.85	8.48	8.34	11.14	***3.16***	4.60
		$S = 40$	**2.45**	6.31	6.05	5.96	5.81	5.30	8.43	7.81	10.65	***2.50***	4.97
		$S = 60$	**2.17**	6.43	6.16	5.59	4.54	5.11	7.25	6.98	10.42	***2.18***	4.63
	$m = 6$	$S = 20$	**3.07**	6.99	6.16	6.78	7.43	6.97	8.32	8.25	11.55	***3.33***	3.80
		$S = 40$	**2.59**	5.24	5.08	6.45	6.21	6.06	8.37	7.60	11.05	***2.89***	4.59
		$S = 60$	**2.37**	6.05	5.49	6.25	5.41	4.74	7.39	6.49	10.81	***2.50***	4.63
	Mean		**2.57**	6.10	5.88	5.63	5.43	6.04	8.16	7.60	10.62	***2.80***	4.84

effectiveness of EEDA on DNFGSP_SDSTs is highly successful since the devised search strategies are based on problem-specific knowledge.

6. Findings, research limitations, and recommendations for future search

6.1. Findings

The research on DFGSPs has emerged as a focal point in smart manufacturing research, while the study of knowledge-enhanced EDAs (EDAs) is a frontier in the evolutionary computation domain. Due to the fact that DFGSPs with no-wait constraints and setup times have considerable challenges such as strongly coupled complexities, how to design efficient EDAs to solve DNFGSP_SDSTs based on problem characteristics and solution structures is a crucial concern urgently needs to be solved, yet relevant research and reports are still scarce. This article presents in-depth research from aspects of problem property analysis, algorithm framework development, and search strategy design, aiming at the effective development of probabilistic models by using problem-specific knowledge, focusing on achieving significant breakthroughs in terms of the overall performance of EDAs and learning-based paradigms. By analyzing the characteristics of FGSPs, we design problem-specific

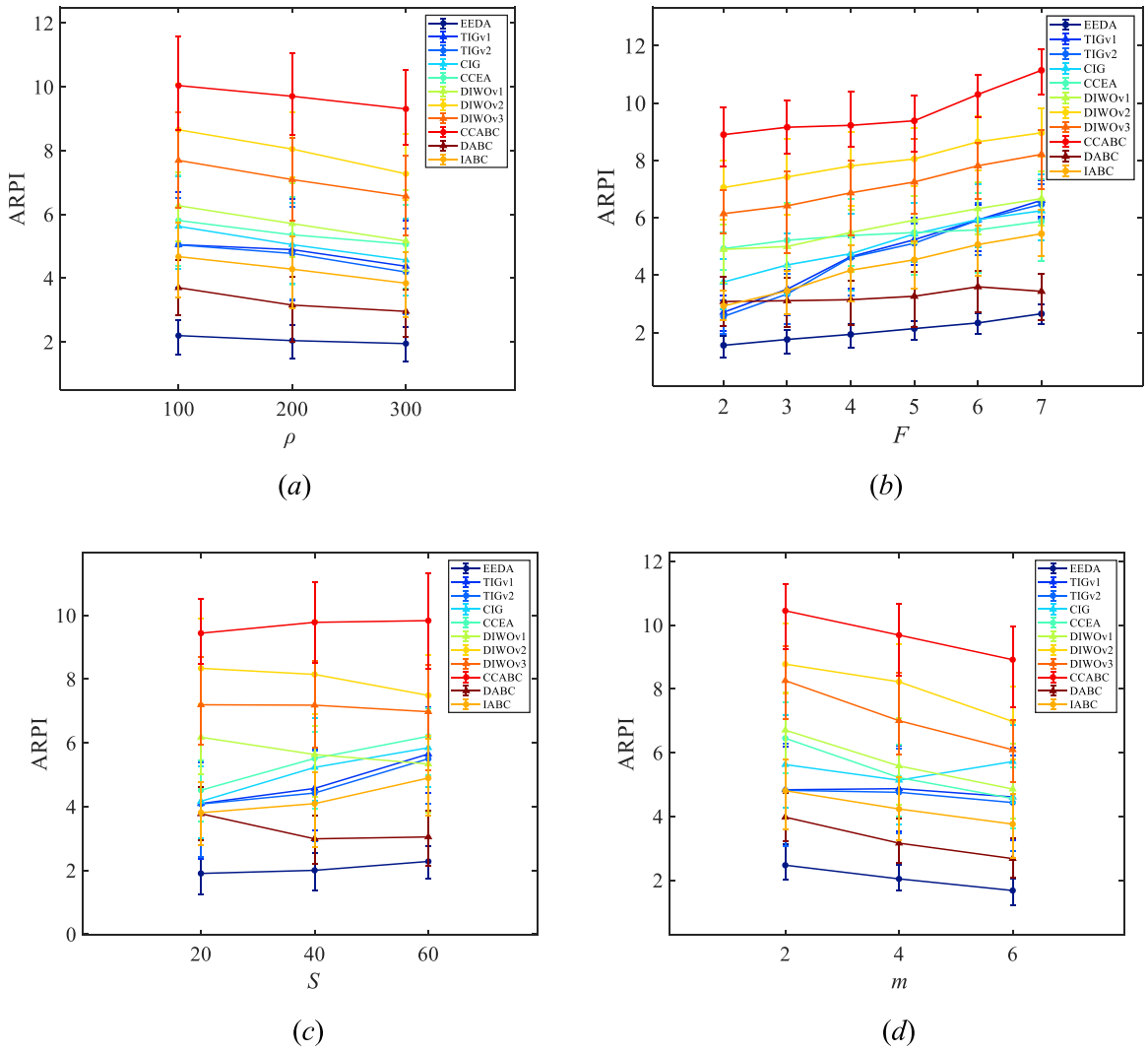


Fig. 17. Interactions for all the algorithms. (*a*) Interactions of algorithms and ρ. (*b*) Interactions of algorithms and *F*. (*c*) Interactions of algorithms and *S*. (*d*) Interactions of algorithms and *m*.

speed-up evaluation methods, fast search strategies for specific neighborhood structures, and enhanced EDA (EEDA) by considering domain knowledge. In EEDA, an effective probability model and a family-based updating mechanism are established to extract the implicit information within and between families through partitioned matrices, adequately revealing the crucial characteristics inherent in superior solutions and the concealed connections among them, learning the potential promising patterns and enriching the searching behaviors, which further offer new ideas and ways for designing high-performance EDAs. Some of the findings are as follows. 1) a MILP model was developed for this strongly NP-hard problem and its validity was verified by the advanced Gurobi solver; 2) a problem-dependent probability model was established that can not only accumulate the frequencies of building blocks but also accurately capture the distribution of these blocks implicit in the sequences of superior solutions; 3) the efficacy of speed-up evaluation methods, family-based updating mechanism, perturbation and reinitialization methods were verified as critical components of EEDA; 4) the effectiveness and efficiency of the proposed EEDA with problem-specific knowledge was comprehensively compared and confirmed.

Several conclusions can be drawn from these findings, which provide insights and inspiration for future researchers. Firstly, as an effective knowledge-enhanced learning paradigm, EEDA relies on rational representation of crucial characteristics and exact extraction of potential patterns, provides the possibility of significant breakthroughs in guiding global search towards promising regions, and thus can moderately reduce search blindness as compared to most of the existing HIOAs that are based on trial-and-error evolutionary mechanisms. Secondly, through probabilistic modeling based on problem-specific knowledge, EEDA can capture crucial connections between families and jobs so as to avoid omitting valuable information, adapting well to changes in the evolutionary process *versus* the most commonly used breadth-first and depth-first search strategies. Consequently, EEDA emerges as the top-performing algorithm and has achieved the best performance among state-of-the-art algorithms in addressing DNFGSP_SDST with the criterion of minimizing the makespan.

6.2. Research limitations

The limitations and impediments of this research are twofold: the interactivity between the global and local searches of EEDA, and the applicability of EEDA in addressing other types of problems. Firstly, the lack of substantial and sufficient interaction between global and local search may cause failure to effectively enable the sharing and transfer of information between the two. It is suggested that the valuable information extracted and estimated by the probability model during the global search should be used to regulate and guide the local search. In addition, the search scope should be further restricted by problem properties to avoid ineffective search, so that the limited search

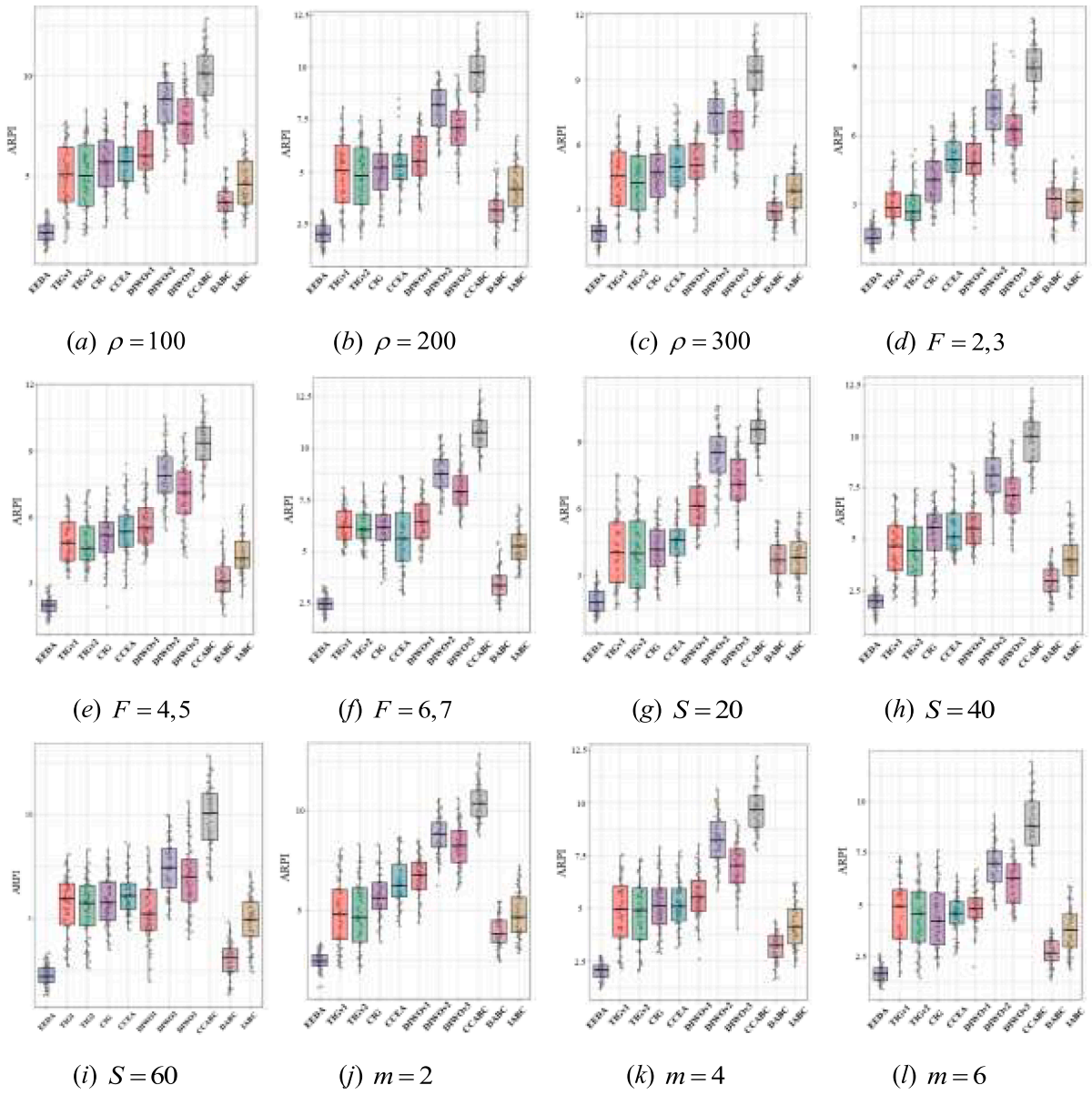


Fig. 18. Box plots of all the algorithms with different factors.

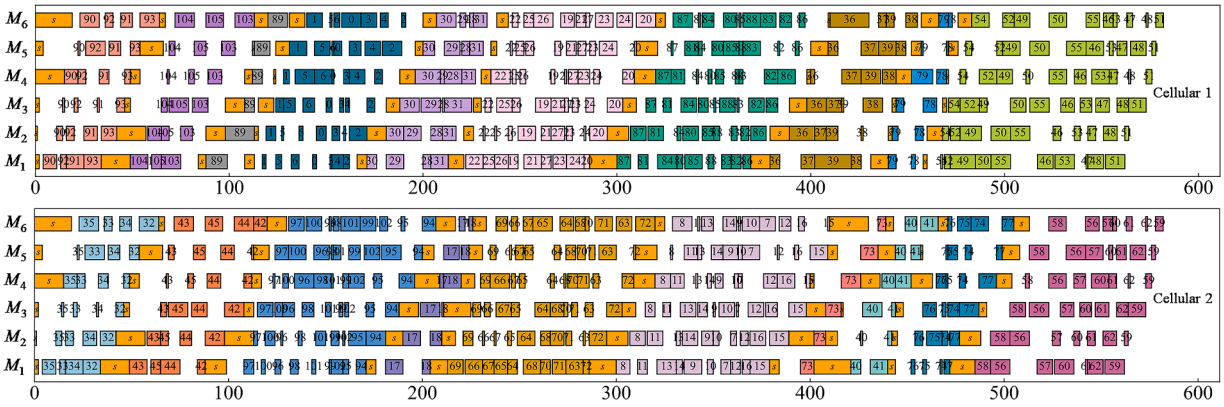


Fig. 19. Gantt chart of the best solution found by EEDA for FG_2_20_6_1_1.

resources are targeted to promising regions within the solution space to enhance search efficiency. Secondly, EEDA was designed based on problem-specific knowledge analyzed and extracted from the characteristics of DNFGSPs, which may not be sufficiently applicable to address other types of shop scheduling problems. Hence, additional attempts are required to explore the potential of knowledge-enhanced

EDAs and to widen their application scope.

6.3. Recommendations for future research

Future research efforts can be outlined in two aspects, focusing on the extension of DFGSPs and the design of efficient EDAs. First, knowledge-based search strategies and search mechanisms, as well as knowledge-enhanced learning-based paradigms, need to be investigated in depth to develop effective co-evolutionary search mechanisms, and complementary collaborative search strategies, and then present robust and effective approaches to tackle the energy-efficient DNFGSP_SDST. Second, attempts will be made to enhance the global search capability of EEDA while exploring its potential applicability to FGSPs and their extended problems. In addition, it is of interest to extend EEDA to solve distributed production, transportation, and assembly integration scheduling problems, which can effectively improve the generality of EDAs and broaden their practical engineering applications.

CRediT authorship contribution statement

Zi-Qi Zhang: Writing – original draft, Methodology, Investigation, Funding acquisition, Conceptualization. **Yan-Xuan Xu:** Writing – original draft, Software, Methodology, Investigation. **Bin Qian:** Writing – review & editing, Supervision, Methodology, Funding acquisition. **Rong Hu:** Writing – review & editing, Funding acquisition, Supervision. **Fang-Chun Wu:** Software, Investigation. **Ling Wang:** Supervision, Project administration.

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.

Data availability

Data will be made available on request.

Acknowledgments

The authors are sincerely grateful to 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).

References

[1] W. Shao, Z. Shao, D. Pi, A network memetic algorithm for energy and labor-aware distributed heterogeneous hybrid flow shop scheduling problem, Swarm. Evol. Comput. 75 (2022) 101190.
[2] Y. Hou, Y. Fu, K. Gao, H. Zhang, A. Sadollah, Modelling and optimization of integrated distributed flow shop scheduling and distribution problems with time windows, Expert. Syst. Appl. 187 (2022) 115827.
[3] Q.K. Pan, L. Gao, L. Wang, An Effective Cooperative Co-Evolutionary Algorithm for Distributed Flowshop Group Scheduling Problems, IEEe Trans. Cybern. 52 (2022) 5999–6012.
[4] C.E. Okwudire, H.V. Madhyastha, Distributed manufacturing for and by the masses, 372 (2021) 341–342.
[5] J. Behnamian, S.M.T. Fatemi Ghomi, A survey of multi-factory scheduling, J. Intell. Manuf. 27 (2016) 231–249.
[6] H.-B. Song, J. Lin, A genetic programming hyper-heuristic for the distributed assembly permutation flow-shop scheduling problem with sequence dependent setup times, Swarm. Evol. Comput. 60 (2021) 100807.
[7] B. Naderi, R. Ruiz, The distributed permutation flowshop scheduling problem, Comput. Oper. Res. 37 (2010) 754–768.
[8] R. Radharamanan, A heuristic algorithm for group scheduling, Comput. Ind. Eng. 11 (1986) 204–208.
[9] J.S. Neufeld, J.N.D. Gupta, U. Buscher, A comprehensive review of flowshop group scheduling literature, Comput. Oper. Res. 70 (2016) 56–74.
[10] A. Allahverdi, The third comprehensive survey on scheduling problems with setup times/costs, Eur J Oper Res 246 (2015) 345–378.
[11] A. Allahverdi, C.T. Ng, T.C.E. Cheng, M.Y. Kovalyov, A survey of scheduling problems with setup times or costs, Eur. J. Oper. Res. 187 (2008) 985–1032.
[12] V.R. Ghezavati, M. Saidi-Mehrabad, An efficient hybrid self-learning method for stochastic cellular manufacturing problem: a queuing-based analysis, Expert. Syst. Appl. 38 (2011) 1326–1335.
[13] M. Ebrahimi, S.M.T. Fatemi Ghomi, B. Karimi, Hybrid flow shop scheduling with sequence dependent family setup time and uncertain due dates, Appl. Math. Model. 38 (2014) 2490–2504.
[14] W.S. Shao, D.C. Pi, Z.S. Shao, A pareto-based estimation of distribution algorithm for solving multiobjective distributed no-wait flow-shop scheduling problem with sequence-dependent setup time, IEEE Trans. Automat. Sci. Eng. 16 (2019) 1344–1360.
[15] W. Shao, Z. Shao, D. Pi, Effective constructive heuristics for distributed no-wait flexible flow shop scheduling problem, Comput. Oper. Res. 136 (2021) 105482.
[16] K.-C. Ying, Z.-J. Lee, C.-C. Lu, S.-W. Lin, Metaheuristics for scheduling a no-wait flowshop manufacturing cell with sequence-dependent family setups, Int. J. Adv. Manuf. Technol. 58 (2012) 671–682.
[17] S. Behjat, N. Salmasi, Total completion time minimisation of no-wait flowshop group scheduling problem with sequence dependent setup times, Eur. J. Ind. Eng. 11 (2017) 22–48.
[18] S.-W. Lin, K.-C. Ying, Makespan optimization in a no-wait flowline manufacturing cell with sequence-dependent family setup times, Comput. Ind. Eng. 128 (2019) 1–7.
[19] C.-Y. Cheng, P. Pourhejazy, K.-C. Ying, Y.-H. Liao, New benchmark algorithms for No-wait flowshop group scheduling problem with sequence-dependent setup times, Appl. Soft. Comput. 111 (2021) 107705.
[20] B. Qian, Z.Q. Zhang, R. Hu, H.P. Jin, J.B. Yang, A matrix-cube-based estimation of distribution algorithm for no-wait flow-shop scheduling with sequence-dependent setup times and release times, IEEE Trans. Syst., Man, Cybernet.: Syst. 53 (2023) 1492–1503.
[21] S. Yuan, T. Li, B. Wang, A discrete differential evolution algorithm for flow shop group scheduling problem with sequence-dependent setup and transportation times, J. Intell. Manuf. 32 (2021) 427–439.
[22] R.L. Graham, E.L. Lawler, J.K. Lenstra, A.H.G.R. Kan, Optimization and approximation in deterministic sequencing and scheduling: a survey, in: P. L. Hammer, E.L. Johnson, B.H. Korte (Eds.), Annals of Discrete Mathematics, Elsevier, 1979, pp. 287–326.
[23] N. İnce, D. Deliktaş, İ.Hakan Selvi, A comprehensive literature review of the flowshop group scheduling problems: systematic and bibliometric reviews, Int. J. Prod. Res., 1–30.
[24] Y. Wang, Y. Han, Y. Wang, J. Li, K. Gao, Y. Liu, An effective two-stage iterated greedy algorithm for distributed flowshop group scheduling problem with setup time, Expert. Syst. Appl. (2023) 233.
[25] N. Karimi, M. Zandieh, H.R. Karamooz, Bi-objective group scheduling in hybrid flexible flowshop: a multi-phase approach, Expert. Syst. Appl. 37 (2010) 4024–4032.
[26] A. Costa, F.A. Cappadonna, S. Fichera, A hybrid genetic algorithm for minimizing makespan in a flow-shop sequence-dependent group scheduling problem, J. Intell. Manuf. 28 (2017) 1269–1283.
[27] A. Costa, F.V. Cappadonna, S. Fichera, Minimizing makespan in a Flow Shop Sequence Dependent Group Scheduling problem with blocking constraint, Eng. Appl. Artif. Intell. 89 (2020) 103413.
[28] M. Solimanpur, A. Elmi, A tabu search approach for group scheduling in buffer-constrained flow shop cells, Int. J. Comput. Integr. Manuf. 24 (2011) 257–268.
[29] D. Lu, R. Logendran, Bi-criteria group scheduling with sequence-dependent setup time in a flow shop, J. Operat. Res. Soc. 64 (2013) 530–546.
[30] D. Hajinejad, N. Salmasi, R. Mokhtari, A fast hybrid particle swarm optimization algorithm for flow shop sequence dependent group scheduling problem, Scientia Iranica 18 (2011) 759–764.
[31] K.-C. Ying, J.N.D. Gupta, S.-W. Lin, Z.-J. Lee, Permutation and non-permutation schedules for the flowline manufacturing cell with sequence dependent family setups, Int. J. Prod. Res. 48 (2010) 2169–2184.
[32] S.-W. Lin, K.-C. Ying, C.-C. Lu, J.N.D. Gupta, Applying multi-start simulated annealing to schedule a flowline manufacturing cell with sequence dependent family setup times, Int. J. Prod. Econ. 130 (2011) 246–254.
[33] N. Salmasi, R. Logendran, M.R. Skandari, Total flow time minimization in a flowshop sequence-dependent group scheduling problem, Comput. Oper. Res. 37 (2010) 199–212.
[34] Y. Du, J.-Q. Li, C. Luo, L.-l. Meng, A hybrid estimation of distribution algorithm for distributed flexible job shop scheduling with crane transportations, Swarm. Evol. Comput. 62 (2021) 100861.
[35] Q.K. Pan, R. Ruiz, An estimation of distribution algorithm for lot-streaming flow shop problems with setup times, Omega-Int. J. Manag. Sci. 40 (2012) 166–180.
[36] Z.Q. Zhang, R. Hu, B. Qian, H.P. Jin, L. Wang, J.B. Yang, A matrix cube-based estimation of distribution algorithm for the energy-efficient distributed assembly permutation flow-shop scheduling problem, Expert. Syst. Appl. 194 (2022) 116484.
[37] Z.-Q. Zhang, B. Qian, R. Hu, H.-P. Jin, L. Wang, A matrix-cube-based estimation of distribution algorithm for the distributed assembly permutation flow-shop scheduling problem, Swarm. Evol. Comput. (2021) 60.

[38] Z.Q. Zhang, R. Hu, B. Qian, H.P. Jin, L. Wang, J.B. Yang, A matrix cube-based estimation of distribution algorithm for the energy-efficient distributed assembly permutation flow-shop scheduling problem, Expert. Syst. Appl. (2022) 194.
[39] Z.Q. Zhang, B. Qian, R. Hu, H.P. Jin, L. Wang, J.B. Yang, A matrix-cube-based estimation of distribution algorithm for blocking flow-shop scheduling problem with sequence-dependent setup times, Expert. Syst. Appl. 205 (2022) 117602.
[40] B. Naderi, N. Salmasi, Permutation flowshops in group scheduling with sequence-dependent setup times, Eur. J. Ind. Eng. 6 (2012) 177–198.
[41] T. Keshavarz, N. Salmasi, Makespan minimisation in flexible flowshop sequence-dependent group scheduling problem, Int. J. Prod. Res. 51 (2013) 6182–6193.
[42] C.-D. Liou, Y.-C. Hsieh, Y.-Y. Chen, A new encoding scheme-based hybrid algorithm for minimising two-machine flow-shop group scheduling problem, Int. J. Syst. Sci. 44 (2013) 77–93.
[43] C.-D. Liou, Y.-C. Hsieh, A hybrid algorithm for the multi-stage flow shop group scheduling with sequence-dependent setup and transportation times, Int. J. Prod. Econ. 170 (2015) 258–267.
[44] T. Keshavarz, N. Salmasi, M. Varmazyar, Minimizing total completion time in the flexible flowshop sequence-dependent group scheduling problem, Ann. Oper. Res. 226 (2015) 351–377.
[45] Y. Li, X. Li, J.N.D. Gupta, Solving the multi-objective flowline manufacturing cell scheduling problem by hybrid harmony search, Expert. Syst. Appl. 42 (2015) 1409–1417.
[46] M.T. Yazdani Sabouni, R. Logendran, Lower bound development in a flow shop electronic assembly problem with carryover sequence-dependent setup time, Comput. Ind. Eng. 122 (2018) 149–160.
[47] Z. Huang, J. Yang, Scheduling optimization in flowline manufacturing cell considering intercell movement with harmony search approach, Mathematics (2020).
[48] A. Goli, T. Keshavarz, Just-in-time scheduling in identical parallel machine sequence-dependent group scheduling problem, J. Ind. Manag. Optimiz. 18 (2022) 3807–3830.
[49] Z.-Y. Wang, Q.-K. Pan, L. Gao, Y.-L. Wang, An effective two-stage iterated greedy algorithm to minimize total tardiness for the distributed flowshop group scheduling problem, Swarm. Evol. Comput. (2022) 74.
[50] J.E. Schaller, J.N.D. Gupta, A.J. Vakharia, Scheduling a flowline manufacturing cell with sequence dependent family setup times, Eur. J. Oper. Res. 125 (2000) 324–339.
[51] S. Arabameri, N. Salmasi, Minimization of weighted earliness and tardiness for no-wait sequence-dependent setup times flowshop scheduling problem, Comput. Ind. Eng. 64 (2013) 902–916.
[52] S.Y. Wang, L. Wang, M. Liu, Y. Xu, An effective estimation of distribution algorithm for solving the distributed permutation flow-shop scheduling problem, Int. J. Prod. Econ. 145 (2013) 387–396.
[53] W.S. Shao, D.C. Pi, Z.S. Shao, Optimization of makespan for the distributed no-wait flow shop scheduling problem with iterated greedy algorithms, Knowl. Based. Syst. 137 (2017) 163–181.
[54] Z.Y. Wang, Q.K. Pan, L. Gao, Y.L. Wang, An effective two-stage iterated greedy algorithm to minimize total tardiness for the distributed flowshop group scheduling problem, Swarm. Evol. Comput. (2022) 74.
[55] Y.-Z. Li, K. Gao, L.-L. Meng, P.N. Suganthan, A problem-specific knowledge based artificial bee colony algorithm for scheduling distributed permutation flowshop problems with peak power consumption, Eng. Appl. Artif. Intell. 126 (2023) 107011.
[56] X. He, Q.-K. Pan, L. Gao, L. Wang, P.N. Suganthan, A greedy cooperative Co-evolution ary algorithm with problem-specific knowledge for multi-objective flowshop group scheduling problems, IEEE Trans. Evol. Comput. (2021), 1-1.
[57] W. Niu, J.-q. Li, A two-stage cooperative evolutionary algorithm for energy-efficient distributed group blocking flow shop with setup carryover in precast systems, Knowl. Based. Syst. (2022) 257.
[58] X. Zhang, X. Liu, A. Cichon, G. Królczyk, Z. Li, Scheduling of energy-efficient distributed blocking flowshop using pareto-based estimation of distribution algorithm, Expert. Syst. Appl. 200 (2022) 116910.
[59] J. Ceberio, E. Irurozki, A. Mendiburu, J.A. Lozano, A review on estimation of distribution algorithms in permutation-based combinatorial optimization problems, Prog. Artificial Intell. 1 (2012) 103–117.
[60] B. Jarboui, M. Eddaly, P. Siarry, An estimation of distribution algorithm for minimizing the total flowtime in permutation flowshop scheduling problems, Comput. Oper. Res. 36 (2009) 2638–2646.
[61] S.H. Chen, M.C. Chen, P.C. Chang, Q.F. Zhang, Y.M. Chen, Guidelines for developing effective Estimation of Distribution Algorithms in solving single machine scheduling problems, Expert. Syst. Appl. 37 (2010) 6441–6451.
[62] S.Y. Wang, L. Wang, An estimation of distribution algorithm-based memetic algorithm for the distributed assembly permutation flow-shop scheduling problem, IEEE Trans, Syst, Man Cybernet,-Syst, 46 (2016) 139–149.
[63] B. Qian, Z.-C. Li, R. Hu, A copula-based hybrid estimation of distribution algorithm for m-machine reentrant permutation flow-shop scheduling problem, Appl. Soft. Comput. 61 (2017) 921–934.
[64] F. Zhao, B. Zhu, L. Wang, An estimation of distribution algorithm-based hyper-heuristic for the distributed assembly mixed no-idle permutation flowshop scheduling problem, IEEE Trans. Syst., Man, and Cybernet.: Systems 53 (2023) 5626–5637.
[65] Z.-Q. Zhang, B. Qian, R. Hu, J.-B. Yang, Q-learning-based hyper-heuristic evolutionary algorithm for the distributed assembly blocking flowshop scheduling problem, Appl. Soft. Comput. 146 (2023) 110695.
[66] S. Hatami, R. Ruiz, C. Andres-Romano, The distributed assembly permutation flowshop scheduling problem, Int. J. Prod. Res. 51 (2013) 5292–5308.
[67] Z.-Y. Wang, Q.-K. Pan, L. Gao, X.-L. Jing, Q. Sun, A cooperative iterated greedy algorithm for the distributed flowshop group robust scheduling problem with uncertain processing times, Swarm. Evol. Comput. (2023) 79.
[68] H.Y. Sang, Q.K. Pan, J.Q. Li, P. Wang, Y.Y. Han, K.Z. Gao, P. Duan, Effective invasive weed optimization algorithms for distributed assembly permutation flowshop problem with total flowtime criterion, Swarm. Evol. Comput. 44 (2019) 64–73.
[69] K. Peng, Q. Pan, B. Zhang, An improved artificial bee colony algorithm for steelmaking–refining–continuous casting scheduling problem, Chin. J. Chem. Eng. 26 (2018) 1727–1735.
[70] H. Li, X. Li, L. Gao, A discrete artificial bee colony algorithm for the distributed heterogeneous no-wait flowshop scheduling problem, Appl. Soft. Comput. (2021) 100.
[71] D.C. Montgomery, Design and Analysis of Experiments, John Wiley & Sons, 2008.