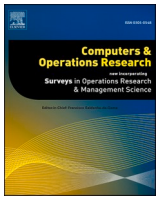
Computers & Operations Research 159 (2023) 106360

Contents lists available at ScienceDirect

Computers and Operations Research

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



Multi-objective energy-efficient hybrid flow shop scheduling using Q-learning and GVNS driven NSGA-II

Peize Li, Qiang Xue, Ziteng Zhang, Jian Chen *, Dequn Zhou

College of Economics and Management, Nanjing University of Aeronautics and Astronautics, Nanjing, China

ARTICLE INFO

Keywords:
Hybrid flow shop
Energy-efficient scheduling
Multi-objective optimization
Time-of-use tariffs
Q-learning

ABSTRACT

The urgent mission for carbon peak and carbon neutrality is demanding greater industrial sustainability. Energy-efficient hybrid flow shop scheduling problem (EEHFSP) has been raising increasing attention in recent years. This paper studies a new EEHFSP with uniform machines to minimize total tardiness, total energy cost, and carbon trading cost. Time-of-use tariffs and power down strategies are simultaneously adopted. A novel multi-objective mixed-integer nonlinear programming model for the problem is proposed. To solve the model, we propose a Q-learning and general variable neighborhood search (GVNS) driven non-dominated sorting genetic algorithm II (QVNS-NSGA-II). The novelty of the algorithm is that we incorporate Q-learning into GVNS to guide premium adaptive operator selection throughout the shaking and local search processes. A distinguishing feature is that the states and actions of Q-learning are set as neighborhood structures and local search operators. The Q-learning-driven GVNS is embedded into NSGA-II to promote the exploration and exploitation capability. Experimental results show that the proposed QVNS-NSGA-II outperforms NSGA-II, improved Jaya, and modified MOEA/D in terms of the quantity, quality of Pareto solutions, and computational efficiency. Sensitivity analysis also derives several managerial implications. The proposed approach can be applied to improve sustainability and productivity for hybrid flow shop manufacturers.

1. Introduction

Energy shortage is one of the most serious problems in many countries due to disrupted supply chains, such as the COVID-19 pandemic or the Russian-Ukrainian conflict. Besides, the majority of the energy we consumed is non-renewable, such as oil, natural gas, and coal in Fig. 1 (IEA, 2021). Energy consumption is accompanied by the release of large amounts of greenhouse gases. This situation exacerbates climate change, making energy-saving and carbon-reducing issues more of vital significance and indispensable for countries around the world. For example, in 2021, China launched a thirty-year plan "carbon peaking" and "carbon neutrality" aiming at reaching the CO2 emissions peak before 2030 and achieving carbon neutrality ahead of 2060 (The State Council of the People's Republic of China, 2021).

It is demonstrated that the industry sector consumed over 40% of electricity and 50% of coal over the last five decades in Fig. 2 (IEA, 2021). Thus, the industrial sector is primarily responsible for establishing sustainability to reduce energy-consuming and alleviate environmental impacts e.g., global warming. Efforts are therefore devoted to energy-efficient scheduling, mainly focusing on improving the ratio between energy input and the desired output of production or service systems, i.e., energy efficiency.

The *hybrid flow shop scheduling problem* (HFSP), also known as the *flexible flow shop scheduling problem*, is a important production scheduling problem widely confronted by many industries, such as electronics (Yue et al., 2023), steel (Jiang et al., 2023) and glass industries (Wang et al., 2020). The HFSP enables flexibility and is suitable for multi-variety and small-batch production (Ribas et al., 2010). It is composed of multiple production stages, where each stage consists of multiple parallel machines. Each job has to go through all stages in the same order string.

Most research assumes that parallel machines at each stage of the HFSP are identical for simplicity. However, the machines in the same stage can run at different speeds. Particularly, higher processing speeds require higher energy consumption rates but lead to shorter processing times, while lower processing speeds take the converse effect (Wu and Che, 2020). The scenario is referred to as a uniform machine environment, to which only 8% of the research on the HFSP contributes (Lee and

* Corresponding author.
E-mail address: jchen@nuaa.edu.cn (J. Chen).

https://doi.org/10.1016/j.cor.2023.106360
Received 24 November 2022; Received in revised form 4 June 2023; Accepted 21 July 2023
Available online 27 July 2023

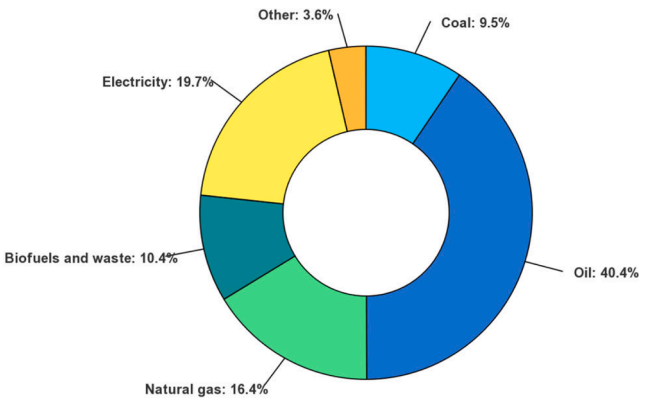


Fig. 1. Share of world total final consumption by source, 2019.

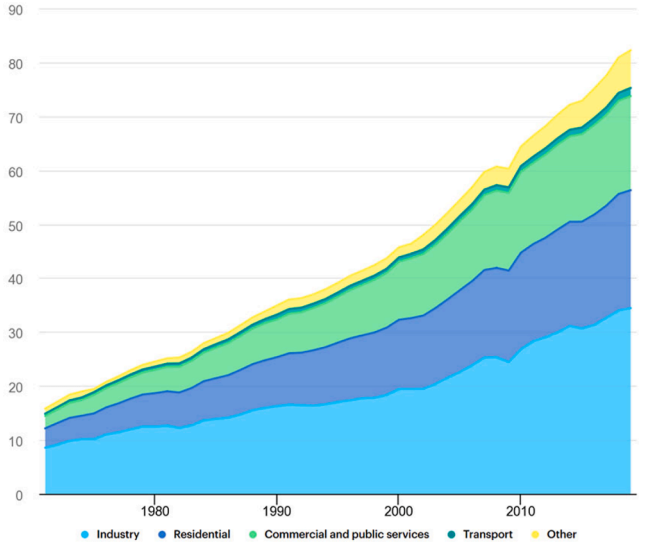


Fig. 2. Electricity total final consumption by sector, 1971–2019.

Loong, 2019). Therefore, this paper extends the study of HFSP with uniform parallel machines.

Usually, classical HFSP discussed production-related indicators such as due date, makespan, and total tardiness (Chen et al., 2020a; Chen et al., 2020b). Taking production efficiency as foundational consideration, the *energy-efficient hybrid flow shop scheduling problem* (EEHFSP) naturally focuses on multi-objective optimization for integrating both production and environmental concerns (e.g., makespan $C_{\max}$ and total energy consumption *TEC*).

This paper investigates an EEHFSP to minimize the three objectives concurrently, namely *total tardiness time* (*TT*), *total energy cost* (*TEC*), and *carbon trading cost* (*CTC*). *TT* is a typical time-related objective for make-to-order production, playing a key role in satisfying customers' demands.

TEC and *CTC* are included for energy efficiency, wherein *CTC* is first defined and considered for lowering carbon emissions. In practice, the carbon trading markets have been established in major economies, such as the US, the EU, and China. For example, the EU has established the largest carbon market EU Emissions Trading System. The greenhouse gases that can be emitted by plants are limited by a 'cap' on the number of emission allowances. Within the cap, companies receive or buy emission allowances, which they can trade as needed (European Commission, 2023).

Energy-efficient strategies (EES) at the operational level are developed by researchers to cut down *TEC* (Gahm et al., 2016, Li and Wang, 2022). These operational strategies can be divided into two categories: "energy supply" and "energy demand" (Gahm et al., 2016):

- Energy supply: This class includes the EES with respect to energy suppliers, e.g., *real-time pricing* (Khalaf and Wang, 2018), *time-of-use* (TOU) tariffs (Luo et al., 2013, Cui and Lu, 2021, Ding et al., 2021), and *critical peak pricing* (CPP) (Chen et al., 2022a; Chen et al., 2022b).
- Energy demand: This class includes the EES on how the demand side increases energy efficiency, e.g., *power down* (Dai et al., 2013, Wang et al., 2020) and *speed-scaling* mechanism (Wu and Che, 2020, Pan et al., 2022).

Concerning energy supply EES, two common strategies are considered, namely TOU tariffs and CPP. TOU tariffs refer that electricity prices fluctuate over a day according to temporal electricity demand. CPP charges punitive electricity prices during periods of high demand. TOU tariffs and CPP policy are often adopted jointly, providing us with the opportunity to shift machine running time from on-peak hours (high price) to low-peak hours (low price), which aims to save energy cost (Shrouf et al., 2014, Gahm et al., 2016, Ding et al., 2021).

With respect to energy demand EES, power down strategy is deployed in this paper. The power down strategy originated from one of the most famous works on EES done by Mouzon et al. (2007). The power down mechanism shuts down idle machines and resets them until needed, which can save a significant amount of energy without prolonging *TT*. Afterward, Mouzon and Yildirim (2008) extended this study by applying the concept of break-even duration, where the machine would be shut down when idle time exceeded the break-even duration. In this study, we jointly consider energy supply and energy demand EES to decrease both *TEC* and *CTC*.

The two-stage HFS has been proven an NP-hard problem even when the first stage has two identical parallel machines and the second stage has only one machine (Gupta, 1988). Consequently, the considered EEHFSP with multiple EES and objectives is NP-hard in a strong sense. Most of the research in the literature, therefore, adopts metaheuristics such as genetic algorithm (GA), tabu search (TS), and variable neighborhood search (VNS) to solve the problem (Chen et al., 2020a; Chen et al., 2020b, Zhao et al., 2021a).

Metaheuristics improve the solution in an iterative way using local search operators while at the same time trying to escape from local optima (Gendreau and Potvin, 2019). In fact, a single operator may perform differently during the search process. The reason lies in the fact that the search space of a combinatorial optimization problem is non-stationary and includes different search regions with dissimilar characteristics. Different operators specialize in different regions (Li et al., 2013). Therefore, researchers may deploy multiple local search operators to enhance the search robustness of metaheuristics (Karimi-Mamaghan et al., 2022, 2023; Zhao et al., 2017; Öztop et al., 2020).

A major concern naturally comes into mind when designing such a metaheuristic: which order should the search operator be deployed to guide the metaheuristic to global optima efficiently? Two fashions are commonly used: offline and online. In offline operator selection, operators are deployed by sequence at random without any knowledge from the former search. In contrast, online operator selection selects the most appropriate operators dynamically during the search process.

Adaptive operator selection (AOS) is one kind of online operator selection using extracted knowledge from the search environment (Li et al., 2013). The main steps of AOS are as follows: reward computation, credit assignment, operator selection and move acceptance. AOS evaluates the reward based on solution improvements after applying an operator and then assigns the credit to the operator to update knowledge. Depending on the learned experience, AOS selects the next operator and decides whether to accept a move or not. Sometimes AOS and hyper-heuristics can be used interchangeably. Refer to the review paper for details (Karimi-Mamaghan et al., 2022).

In recent years, there has been a growing research interest in integrating machine learning techniques into metaheuristics, enabling

metaheuristics to extract knowledge from data (Bengio et al., 2021; Karimi-Mamaghan et al., 2020, 2022, 2023; Talbi, 2021). Particularly, *reinforcement learning* (RL) as a subfield of machine learning, specializes in interacting over time with its environment to achieve a goal (Richard and Andrew, 2019). The property of RL makes it just suitable to drive AOS select operators at each step, which is rising in heat (Cai et al., 2021; Durgut et al., 2021; Karimi-Mamaghan et al., 2023; Richard and Andrew, 2019).

To solve the EEHFSP problem, this paper proposes a novel RL-driven hybrid meta-heuristic. We adopted *Q-learning*, a famous RL algorithm, to learn the optimal behavior of operators. The Q-learning-driven GVNS (QVNS) realizes AOS in the EEHFSP to select appropriate local search operators during iteration. Then QVNS is embedded into Non-dominated Sorting Genetic Algorithm II (NSGA-II) (Deb et al., 2002) to form a new algorithm called QVNS-driven NSGA-II (QVNS-NSGA-II). To the best of our knowledge, this is among the first research that combines metaheuristics with RL to solve the EEHFSP.

The main contributions of this paper are summarized.

- We presented a novel multi-objective mixed-integer nonlinear programming (MINLP) model for EEHFSP to minimize time-related, environment-related objectives (i.e., *TT*, *TEC,* and *CTC*). The model considers both energy supply side EES (TOU tariffs and CPP) and energy demand side EES (power down mechanism) with uniform parallel machines. The properties and complexity of the presented model are analyzed.
- The Q-learning is incorporated into GVNS to perform AOS during the search, enabling the agent to select the appropriate operator adaptively in the shaking and local search phases. The algorithm complexity is also given. In QVNS, the Q-learning features a specialized reward function and Pareto solution move acceptance for EEHFSP. GVNS considers sets of problem-oriented neighborhood structures and local search operators.
- The proposed QVNS-NSGA-II is among the first RL-served metaheuristics for EEHFSP. QVNS is performed during each iteration to improve the incumbent solution for higher diversity and quality. Experimental results demonstrate the superiority of our algorithm in terms of quality, diversity, and computational efficiency. A sensitivity analysis is also conducted to yield managerial insights in selecting preferable Pareto solutions.

The remainder of this paper is organized as follows. In Section 2, works related to EEHFSP and RL-assisted metaheuristics are briefly reviewed. The detailed problem formulation and mathematical model are described in Section 3. Section 4 introduces the background of GVNS and Q-learning. Section 5 presents the whole picture of the proposed QVNS-NSGA-II. Then the proposed algorithm is numerically compared to the classical NSGA-II as well as two state-of-the-art algorithms in Section 6. Finally, Section 7 concludes the paper and discusses the possible directions of future research.

2. Literature review

Over the past three decades, extensive effort has been devoted to the flow shop scheduling problem (FSP). Refer to the comprehensive reviews given by Yenisey and Yagmahan (2014), and Alejandro Rossit et al. (2018). FSP can be roughly classified into three main categories: permutation FSP (PFSP), hybrid FSP (HFSP), and distributed FSP (DFSP). As the concern for environmental protection grows, the research on energy-efficient FSP has been increasing significantly since 2013 (Gao et al., 2020, Li and Wang, 2022). From a methodological perspective, researchers are carrying out extensive work on efficient metaheuristics. The integration of machine learning knowledge into metaheuristics has attracted enormous attention recently (Bengio et al., 2021; Karimi-Mamaghan et al., 2022).

In what follows, we mainly review two streams of works highly related to our research: (1) Energy-efficient flow shop scheduling, and (2) RL-based metaheuristics. At last, we summarize the research gaps in this paper.

2.1. Energy-efficient flow shop scheduling

In this section, we mainly focus on problem characteristics including production environment, energy-efficient strategies, and objectives. Table 1 shows the classification of research on energy-efficient FSP (EEFSP) within the last ten years.

Quite a few EEFSP studies concentrated on PFSP and DFSP, in which especially the number of papers related to DFSP is increasing in recent three years. HFSP is an extension of PFSP, with the advantage of production flexibility (Lei and Zheng, 2017). HFSP can also act as components of DFSP, for instance, Lu et al. (2022) investigated a distributed HFSP to minimize the $C_{\max}$ and *TEC*.

Aiming to EEFSP, most research jointly considers production-related objective $C_{\max}$ and energy-related objective *TEC*. *TT* plays a key role in improving customer satisfaction, however, few papers incorporate the objective of minimizing *TT*. Besides, green objectives such as total carbon emission (*TCE*), noise, and pollution are still hardly incorporated into EEFSP (Li and Wang, 2022). Dong and Ye (2022) optimized *TCE* and *TEC* under TOU prices on a distributed two-stage reentrant HFSP. Their study neglected the importance of production-related indicators.

Ghorbani Saber and Ranjbar (2022) consider the minimization of *TT* and *TCE* in PFSP. They developed a multi-objective decomposition-based heuristic algorithm, as well as a multi-objective VNS algorithm to solve the problem. As far as we are concerned, *CTC* has never been considered in EFFSP for the purpose of green scheduling. No paper investigates an EEHFSP problem to minimize *TT*, *TEC* and *CTC* simultaneously.

Furthermore, most research only takes one-side energy-efficient strategy, for example, either power down (energy demand), speed-scaling (energy demand), or TOU (energy supply) mechanisms.

With respect to energy demand strategies, many studies extensively investigated speed-scaling strategy in energy demand. The speed-scaling strategy assumes machines can adjust among discrete speeds dynamically during processing, so as to save energy costs. Ding et al. (2016) adopted speed-scaling strategy to reduce *TCE* of a PFSP and proposed a modified NEH heuristic and an iterated greedy algorithm to solve the model. Goli et al. (2023) first presented a novel metaheuristic to optimize a non-permutation FSP and lot-sizing. This model aimed to determine the lot size and determine each machine's speed to minimize $C_{\max}$ and *TEC* simultaneously. Wang et al. (2023a) presented an energy-efficient fuzzy HFSP model using speed-scaling strategy and employed extended NSGA-II to minimize fuzzy $C_{\max}$ and *TEC*.

Compared to speed-scaling strategy, less research focused on power down strategy, which shuts down idle machines to save energy consumption. Based on power down strategy, Dai et al. (2013) proposed a novel mathematical model for EEHFSP. An improved genetic-simulated annealing algorithm was adopted to make a compromise between $C_{\max}$ and *TEC*. Lu et al. (2022) extended the study of power down strategy into DFSP. They addressed the problem by designing a Pareto-based multi-objective hybrid iterated greedy algorithm.

Regarding energy supply strategy, Luo et al. (2013) are among the first to investigate EEHFSP considering uniform parallel machine and TOU prices. They introduced a new ant colony optimization to solve the problem. Ho et al. (2022) proposed a new MIP for two-machine PFSP with TOU electricity prices to minimize TEC. They employed an exact method by Logic-based Benders decomposition to solve the problem, and test results prove the method's superiority. An et al. (2023) discussed a complex maintenance planning and production scheduling problem for serial-parallel manufacturing systems under TOU tariffs. To solve the problem, they developed an energy-efficient two-stage maintenance strategy to minimize the sum of the *TEC* and *TT*.

It is found that the integration of both energy demand strategies and

Table 1
Classification of the research on EEFSP.

	Production Environment			Energy-efficient strategies							
Ref.	P	H/F	D	TOU	Uniform	On-off	Speed	Objectives	Algorithm	Local search	Selection
Dai et al. (2013)		✓				✓		C_{max}, *TEC*	GA, SA	2pXO	–
Luo et al. (2013)		✓		✓	✓			C_{max}, *TEC*	ACO	PT	–
Ding et al. (2016)	✓						✓	C_{max}, *TCE*	NEH, IG	NEH Insr, DC	–
Chen et al. (2019)			✓				✓	C_{max}, *TEC*	COA	Swap, Insr, SA,SD	AOS
Chen et al. (2020b)		✓					✓	C_{max}, *TEC*	NSGA-II	2pXO, OXO, RA	Random
Öztop et al. (2020)	✓			–	–	–	–	C_{max}	IG, VNS, QL	Insr, Swap	Sequential
Wu and Che (2020)	✓						✓	C_{max}, *TEC*	VNS	Swap, Insr	AOS
Wang et al. (2020)		✓		✓		✓		C_{max}, *TEC*	CH, TS, ACO	BS, PT	–
Cui and Lu (2021)	✓			✓		✓		*TEC*	GA, DP	OXO, Inv	Random
Cheng et al. (2022)	✓						✓	C_{max}, *TEC*	QL, HH	GW, Jaya, 2pXO	AOS
Dong and Ye (2022)		✓				✓		C_{max}, *TCE*, *TEC*	SSA, NSGA-III	OXO, RA, Insr	Random
Li et al. (2022)	✓			–	–	–	–	C_{max},	ABC, QL, NEH	Swap, 2Swap, B-Insr, DC, Insr-Inv	AOS
Lu et al. (2022)			✓			✓		C_{max}, *TEC*	MOHIG	OXO, Swap, Insr, DC	Random
Pan et al. (2022)			✓				✓	C_{max}, *TEC*	Jaya	OXO, Insr	Sequential
Shao et al. (2022)			✓	✓		✓		*TT*, *TEC*	VNS	G-Insr, G-Swap	Sequential
Zhao et al. (2021b)			✓	–	–	–	–	C_{max}	CWWO, VNS	Prop, DC	AOS
Zhao et al. (2021a)	✓						✓	C_{max}, *TEC*	TS, ILS	Insr, BS	Sequential
Zhao et al. (2022a)			✓				✓	*TT*, *TEC*	QL, HH	Insr, Swap, SA, SD	AOS
Zhao et al. (2022b)			✓				✓	*TT*, *TEC*, *RAB*	QL, BSO	Insr, Swap	AOS
An et al. (2023)		✓		✓	✓			*TT*, *TEC*	GA	OXO, 2pXO	Random
Cai et al. (2023)			✓	–	–	–	–	C_{max}	SFLA, QL	Swap, Insr, OXO	AOS
Goli et al. (2023)	✓*						✓	C_{max}, *TEC*	ALO, KA MOKSEO	RW, Swirl, Move	Sequential
Karimi-Mamaghan et al. (2023)	✓			–	–	–	–	C_{max}	IG, QL	DC, Insr	AOS
Wang et al. (2023c)		✓					✓	C_{max}, *TEC*, *STD*	FA, VNS, NSGA-II, IG	OXO, PBX, 2pXO, Swap, Insr, Inv	Sequential
Wang et al. (2023a)		✓					✓	C_{max}, *TEC*	NSGA-II	2pXO, Swap	Sequential
Yue et al. (2023)		✓		–	–	–	–	*TT*, *TEC*	HPSMO	Insr, PPX	Sequential
This study		✓		✓	✓	✓		*TT*, *TEC*, *CTC*	QVNS-NSGA-II	P-DC	AOS

"√" means the factor considered in the article, and "-" means not.
* Non-permutation flow shop that allows changes in the job order on different machines, which is a generalization of the permutation flow shop.
Notations:
Production Environment: P: Permutation flow shop, H/F: Hybrid/Flexible flow shop, D: Distributed flow shop.
Energy-efficient strategies: TOU: Time of use electricity prices, Uniform: uniform parallel machines, On-off: power down, Speed: Speed-scaling.
Objectives: C_{max}: makespan, *TEC*: Total Energy Consumption, *TCE*: Total Carbon Emission, *TT*: Total Tardiness, *STD*: Total Starting Time Deviation, *RAB*: Resource Allocation Balancing.
Local search: XO: Crossover, 1(2)pXO: 1(2)-point XO, PT: Pheromone Trails, Insr: Insertion, NEH Insr: NEH heuristic Insr, SA: Accelerate the speed of an operation, SD: Decelerate the speed of an operation, OXO: Order XO, Inv: Inverse, RA: Rearrange, BS: Block shift, B-Insr: Bind Insr, DC: Destruction-Construction, GW: Grey Wolf Optimization, G-Insr: Greedy Insr, G-Swap: Greedy Swap, Prop: Propagation RW: Random Walk, PPX: Precedence Preservative XO, P-DC: Pareto-based DC,
Algorithm: GA: Genetic Algorithm, SA: Simulated Annealing, ACO: Ant Colony Optimization, NEH: NEH heuristic, IG: Iterated Greedy, COA: Collaborative Optimization Algorithm, NSGA-II: The Non-dominated Sorting GA, VNS: Variable Neighborhood Search, QL: Q-learning, CH: Constructive Heuristic, TS: Tabu Search, DP: Dynamic Programming, HH: Hyper-heuristic, SSA: Salp Swarm Algorithm, ABC: Artificial Bee Colony Algorithm, MOHIG: Multi-objective Hybrid IG, CWWO: Cooperative Water Wave Optimization, ILS: Iterated local search, BSO: Brain Storm Optimization Idea, SFLA: Shuffled Frog-leaping Algorithm, ALO: Ant Lion Optimizer, KA: Keshtel Algorithm, MOKSEO: Multi-objective Keshtel and Social Engineering Optimizer, FA: Firefly Algorithm, HPSMO: Hybrid Pareto Spider Monkey Optimization.
Selection: AOS: Adaptive operator selection.

energy supply strategies has hardly been investigated. Wang et al. (2020) derived an EEHFSP from a real-world glass factory, in which parallel machines with eligibility are at stage 1 and a batch machine is at stage 2. They integrated power down and TOU strategies to save energy consumption, however, they did not extend this research into *k*-stage EEHFSP with uniform parallel machines. The studied problem in our study is a generalization of their work. So far, there is no literature on solving EEHFSP with *k*-stage uniform parallel machines using TOU prices and power down mechanism.

2.2. Rl-assisted metaheuristics

Due to the NP-hard complexity of EEFSP (Garey et al., 1976; Gupta, 1988), swarm intelligence and evolutionary algorithms (EA) are usually employed to solve the problems. Table 1 also presented a methodological classification of EFFSP.

It is demonstrated that most papers have employed Pareto-based metaheuristics considering multiple objectives simultaneously in a Pareto front, such as NSGA-II, artificial bee colony, and ant colony optimization. Lu et al. (2022) designed a Pareto-based hybrid iterated greedy algorithm for energy-efficient DFSP, wherein one cooperative initialization strategy and one knowledge-based multi-objective local search method were invented to boost the algorithm performance. Pan et al. (2022) employed a newly developed metaheuristic Jaya to solve DFSP. The Jaya algorithm is a Pareto-based EA regarded as easy to execute because it has only two parameters. Wang et al. (2023c) have incorporated a fast non-dominated sorting method and elite preserving strategy from NSGA-II into an EA called firefly algorithm to solve an EEHFSP.

The aforementioned metaheuristics are proven to be effective to find Pareto solutions in specific problem settings. A well-problem-tailored metaheuristic can find its niche given a problem set. Among the widely employed metaheuristics for EEHFSP, NSGA-II has been proven to be one of the most promising EAs applied to this problem so far (Chen

et al., 2020b,a; Ding et al., 2021; Wang et al., 2023c,a). Therefore, we choose NSGA-II as the algorithm framework to develop a more efficient metaheuristic for EEHFSP.

In recent years, machine learning techniques attract increasing attention in solving optimization problems. Wang et al. (2023b) developed a deep reinforcement learning method, LSTM-TD(0) to directly solve non-permutation FSP with the minimization of $C_{\max}$. The work of Wang et al. (2023b) is referred to as "End to end learning" which outputs solutions directly from instances. However, only using machine learning cannot be suitable for complex combinatorial optimization problems, for example, EEHFSP with multiple objectives (Bengio et al., 2021).

Most works applied RL techniques in their metaheuristics in the aspect of parameter setting. Chen et al. (2020a) designed self-learning GA based on Q-learning and SARSA to choose key parameters automatically, i.e., crossover rate and mutation rate. The proposed algorithm is applied to a flexible job shop taking $C_{\max}$ as the objective. Öztop et al. (2020) proposed a general variable neighborhood search (GVNS) through Q-learning to solve the no-idle PFSP with the minimization of $C_{\max}$. The Q-learning was adopted to adjust the parameters of the algorithm dynamically, for example, the parameter of the acceptance criterion.

Some papers have adopted RL techniques in FSP for AOS. Traditional AOS employs simple added-value methods without any reinforcement learning knowledge, such as Chen et al. (2019) and Wu and Che (2020). Zhao et al. (2021) used reinforcement learning techniques to learn AOS in a distributed assembly no-idle FSP. A propagation operator based on the Q-learning and VNS is introduced to a newly developed EA, namely cooperative water wave optimization. Cai et al. (2023) defined a novel Q-learning process to help shuffled frog-leaping algorithm select a local search operator dynamically in a DFSP with the objective of $C_{\max}$. Karimi-Mamaghan et al. (2023) integrated a Q-learning into a perturbation mechanism, boosting an IG algorithm to solve a PFSP with $C_{\max}$. Li et al. (2022) proposed an improved artificial bee colony algorithm with Q-learning for solving PFSP with minimizing the $C_{\max}$.

The above work proved that integrating RL-powered AOS with metaheuristics is promising to improve exploration ability. However, the major literature focused on AOS in single-objective FSP, ignoring the technical problems incurred by multi-objective optimization, e.g., how to define reward function in the Q-learning process and how to compare two Pareto solutions. In this regard, Cheng et al. (2022) formed a multi-objective Q-learning-based hyper-heuristic with bi-criteria to select three low-level heuristics, i.e., Grew wolf operator, Jaya operator, and GA operator. Our work differs from theirs in selecting local search operators rather than complete metaheuristics.

To the best of our knowledge, among studies with operator selection, no one has studied RL-driven AOS in EEHFSP. Thus, designing an effective RL-driven metaheuristic in multi-objective EEHFSP is a gap need to fill in.

2.3. Research gaps

To sum up, we can identify the research gaps as follows:

- Most research on EEHFSP focused on the minimization of $C_{\max}$ and *TEC*. Limited literature considers the combination of *CTC*, *TT* and *TEC*.
- The literature considering both the supply side and demand side EES is scarce. No study generally integrates TOU price, CPP and power down strategy on a *k*-stage EEHFSP with uniform parallel machines.
- The majority of previous metaheuristics for EEHFSPs extract no knowledge during the search, resulting in blind operator selection. The AOS driven by RL algorithms is hardly investigated in EEHFSP.
- Most literature on AOS focused on single-objective optimization problems, ignoring the technical problems incurred by multi-objective optimization. Q-learning- and GVNS-driven multi-objective metaheuristics for EEHFSP have seldom been studied yet.

3. Problem formulation and modeling

3.1. Problem formulation

Consider an HFS consisting of s stages. Each stage $k \in \{1, \ldots, m\}$ has a set of uniform parallel machines M_k. It is assumed that machines in parallel at each stage run at different speeds. The speed of machine m_{ik} is v_{ik}. The operation o_{jk} requires p_{jk} units of time to be processed. If the operation o_{jk} is processed on machine m_{ik}, it actually requires p_{jk}/v_{ik} time units to be finished (Chen et al., 2022a; Chen et al., 2022b).

There are a set of jobs J to be scheduled. Each job $j \in J$ needs to go through all stages. In each stage $k \in \{1, \ldots, m\}$, its operation $o_{jk} \in \{o_{j1}, \ldots, o_{jm}\}$ is processed. An operation o_{jk} can be processed on any machine $m_{ik} \in M_k$ at stage $k \in \{1, \ldots, m\}$.

Each job $j \in J$ has its due date d_j. The completion time of the job $j \in J$ at the stage k is denoted as C_{jk}. When a job is completed later than its due date d_j, tardiness $T_j = C_{jm} - d_j$ occurs. The total tardiness of jobs $TT = \sum_{j \in J} (\max\{0, C_{jm} - d_j\})$.

The basic assumptions of HFSP are included:

- The release time of all jobs is zero, and all machines and jobs are available at time zero
- Each job can be processed at one machine at a time.
- Each machine can process at most one job at a time.
- The job cannot be interrupted once it begins, namely, preemption is not allowed

In addition to basic assumptions, the additional assumptions are given as follows:

- All machines have four states: processing, standby, reset, and shutdown, and each state corresponds to different energy consumption. Machines can be turned down or kept idle after completing a job.
- Each uniform machine has its processing speed. Higher speed incurs higher energy consumption.

Energy consumption is considered for machining jobs. Three types of energy consumption are considered as follows:

- The basic process energy consumed per unit time (i.e., power) for uniform machines at stage k is pe_k. Based on the power conversion factor λ_{ik} introduced by Mansouri et al. (2016), let each v_{ik} correspond to a λ_{ik}. Thus, the actual process power of machine m_{ik} is $\lambda_{ik} pe_k$. Usually, a fast machine has higher power than a slow one.
- When a machine stays standby state, the standby energy is incurred, see Fig. 3. The standby power of machine m_{ik} is denoted by se_{ik}.
- In order to reduce the standby energy consumption, power down mechanism is introduced. That means a machine can be turned off to save energy when it is idle. But when turning the machine back on, a reset energy will be generated. The reset time is t_{reset} and the reset

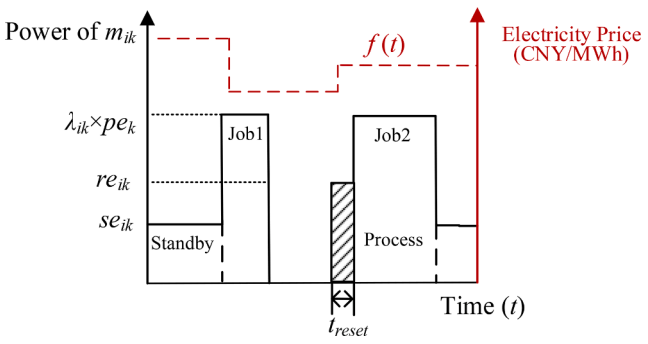


Fig. 3. Three types of machine power.

power of machine m_{ik} is given as; so the reset energy can be easily obtained by $re_{ik} \cdot t_{reset}$ (Ding et al., 2021; Luo et al., 2013).

TOU mechanism is also considered that the electricity price of different periods varies. Let $f(t)$ denote the electricity price at period $t \in N$, where t is an integer. Thus, the total energy cost $TEC = \sum_{t\in N} f(t) \sum_{k\in S} \sum_{i\in M_k} E_{ik}^t$. Note that CPP mechanism is defined in $f(t)$, which charges critical pricing for energy demand peaks. For simplicity, E_{ik}^t is defined as the power consumption of each machine at time $t \in N$, which is determined by the three types of power, i.e., process power, standby power, and reset power.

In addition, in order to take environmental protection into consideration, carbon trading cost CTC is considered. Assume that a factory has a carbon emission allowance denoted by EA. Once the carbon emission exceeds the allowance, the factory needs to pay for additional carbon emission rights in the carbon emission trading market. The carbon emission coefficient μ converts electricity consumption into carbon emissions. We can obtain $CTC = [\sum_{t\in N} \sum_{k\in S} \sum_{i\in M_k} (E_{ik}^t \times \mu) - EA] \times C_p$, where C_p is the price of carbon emissions per ton in the carbon trading market. Note that CTC and TEC are somewhat related, but are not necessarily proportional. CTC is directly related to energy consumption; however, TEC is not only affected by energy consumption but also by TOU.

The goal of the MOE-HFS problem is to minimize three objectives, which are:

- Total tardiness TT,
- Total energy cost TEC,
- Carbon trading cost CTC.

The MOE-HFS problem does not only determine how to batch jobs onto machines at each stage and when to process, but also determines machine states at each period. According to $\alpha|\beta|\gamma$ notation, it can be summarized as $HFs(Qm_1, \ldots, Qm_k)|\text{TOU}, on-off|\{TT, CTC, TEC\}$. Thereby, HFs denotes an HFS with s production stages; Q stands for uniform machines with different speeds (Lee and Loong, 2019).

3.2. Mathematical modeling

To formally model the problem, the notation is defined as follows in Table 2:

Min

$$TT = \sum_{j\in J} \left(\max\{0, C_{js} - d_j\}\right) \tag{1}$$

$$TEC = \sum_{t\in N} f(t) \sum_{k\in S} \sum_{i\in M_k} E_{ik}^t \tag{2}$$

$$CTC = \left[\sum_{t\in N} \sum_{k\in S} \sum_{i\in M_k} (E_{ik}^t \times \mu) - EA\right] \times C_p \tag{3}$$

Subject to

$$\sum_{k\in S} \sum_{i\in M_k} \sum_{j\in J} a_{ikj}^t \leqslant 1, t \in N \tag{4}$$

$$\sum_{t\in N} \sum_{i\in M_k} b_{ikj}^t = 1, j \in J, k \in S \tag{5}$$

$$\sum_{t\in N} b_{ik0}^t = 1, i \in M_k, k \in S \tag{6}$$

$$b_{ikj}^1 = a_{ikj}^1, i \in M_k, j \in J, k \in S \tag{7}$$

$$b_{ikj}^t \geqslant a_{ikj}^t - a_{ikj}^{t-1}, i \in M_k, j \in J, k \in S, t > 1 \tag{8}$$

Table 2
The notation.

Sets	
J	Set of jobs, $j, h \in J = \{1, \ldots, n\}$
S	Set of stages, $k \in S = \{1, \ldots, m\}$
M_k	Set of machines at the stage $k \in S$, $i \in M_k = \{1, \ldots, l_k\}$
N	Set of periods, $t \in N$
Parameters	
d_j	Due time of the job j, $j \in J$
p_{jk}	Processing time of operation o_{jk} (job j at the stage k)
v_{ik}	The processing speed of machine i at stage k, e.g., $v_{ik} \in \{v_1, v_2, v_3\}$ for three speed modes
λ_{ik}	The conversion factor for processing speed v_{ik}
pe_k	The basic power of machines at the stage k
re_{ik}	Reset energy consumed per unit time for machine m_{ik}
se_{ik}	Standby energy consumed per unit time for machine m_{ik}
$f(t)$	Energy price in period t (CNY/MWh)
t_{reset}	Time consumed to turn on and turn off the machine
B	A very large positive number
C_p	Price of carbon emissions per ton in the carbon trading market
EA	Emission allowances
μ	Carbon emission per ton of electricity consumed (tCO_2/MWh)
Decision Variables	
a_{ikj}^t	1 if the operation o_{jk} is processed on the machine m_{ik} in time t, and 0 otherwise
b_{ikj}^t	1 if operation o_{jk} begins on the machine m_{ik} in time t, and 0 otherwise
χ_{ikjh}	1 if job j immediately precedes job h on the machine m_{ik}, otherwise 0, $j \neq h$, $j, h \in J$
x_{ik}^t	1 if the machine m_{ik} is processing or idle, otherwise 0 (see Table 3)
y_{ik}^t	1 if the machine m_{ik} is in reset or idle, otherwise 0 (see Table 3)
E_{ik}^t	Energy consumption of machine m_{ik} in the period t
S_{jk}	Start time of job j at the stage k
C_{jk}	Completion time of job j at the stage k
C_{js}	Completion time of job j at last stage s
T_j	Tardiness time of job j

$$\sum_{t\in N} a_{ikj}^t = \sum_{t\in N} b_{ikj}^t \times \frac{p_{jk}}{v_{ik}}, i \in M_k, j \in J, k \in S \tag{9}$$

$$x_{ik}^t y_{ik}^t = \sum_{j\in J} a_{ikj}^t, i \in M_k, k \in S, t \in N \tag{10}$$

$$E_{ik}^t = x_{ik}^t (1 - y_{ik}^t) se_{ik} + y_{ik}^t (1 - x_{ik}^t) re_{ik} + \sum_{j\in J} a_{ikj}^t \lambda_{ik} pe_k, i \in M_k, k \in S, t \in N \tag{11}$$

$$\sum_{h\in J\cup\{0\}} \chi_{ikjh} = \sum_{t\in N} b_{ikj}^t, i \in M_k, j \neq h, j \in J \tag{12}$$

$$\sum_{j\in J\cup\{0\}} \chi_{ikjh} = \sum_{t\in N} b_{ikh}^t, i \in M_k, j \neq h, h \in J \tag{13}$$

$$S_{jk} = \sum_{i\in M_k} \left(\sum_{t\in N} b_{ikj}^t \times t\right), i \in M_k, k \in S, t \in N \tag{14}$$

$$C_{jk} = \sum_{i\in M_k} \left(\sum_{t\in N} b_{ijk}^t \times \left(t + \frac{p_{jk}}{v_{ik}} - 1\right)\right), i \in M_k, k \in S, t \in N \tag{15}$$

$$C_{jk} \leqslant S_{j(k+1)}, j \in J, k \in S \tag{16}$$

$$C_{hk} - \frac{p_{hk}}{v_{ik}} - C_{jk} \geqslant B(\chi_{ikjh} - 1), j \neq h, i \in M_k, k \in S, j \in J \cup \{0\}, h \in J \tag{17}$$

$$\chi_{ikjh} \in \{0, 1\}, j \neq h, i \in M_k, k \in S, j \in J \cup \{0\}, h \in J \tag{18}$$

$$a_{ikj}^t \in \{0, 1\}, i \in M_k, j \in J, k \in S \tag{19}$$

$$x_{ik}^{t}, y_{ik}^{t} \in \{0,1\}, i \in M_k, k \in S, t \in N \quad (20)$$

Eq. (1–3) are three objective functions to minimize, namely *TT*, *TEC* and *CTC*. Constraint (4) ensures that only one job is assigned to a machine at a time. Constraint (5) enforces that each operation $o_{jk} \in \{o_{j1}, ..., o_{jm}\}$, $j \in J$, $k \in S$, has to be processed and should not be interrupted once it begins. In other words, an operation is started exactly once.

It is worth mentioning that for modeling, we define a dummy job indexed by 0 with no processing time. Eq. (6) ensures that the dummy job is assigned to each machine. Constraints (7-9) define the connections between indicators a_{ikj}^{t} and b_{ikj}^{t}. It is obvious that if o_{jk} is produced by machine m_{ik} at $t = 1$, then it must begin at that time. Eq. (8) ensures that when a_{ikj}^{t} changes, b_{ikj}^{t} follows. Eq. (9) also defines the actual processing time of an operation.

Eq. (10) establishes the connection among a_{ikj}^{t}, x_{ik}^{t} and y_{ik}^{t}. $a_{ikj}^{t} = 1$ only when $x_{ik}^{t} = 1$ and $y_{ik}^{t} = 1$. Constraint (11) defines the energy consumption of machine m_{ik} in the period t. Table 3 lists the states of a machine defined by both x_{ik}^{t} and y_{ik}^{t}. For example, when a machine is in reset state in time t, $x_{ik}^{t} = 0, y_{ik}^{t} = 1$, and its corresponding power is re_{ik}.

Constraints (12-13) ensure that for a job, there should be only one job immediately before it and one immediately after it on the same machine. Constraints (14-15) define the start time and completion time of an operation. Note that there exists a term for subtracting one time unit because t begins at 1. Constraints (16-18) specify the connections between start time and completion time. Constraint (16) imposes that the next operation of a job cannot start until its previous operation has been completed. Constraint (17) enforces that job h should start after completing its preceding job j. Constraints (18-20) define all binary decision variables.

The proposed model is a multi-objective mixed integer program. Since there is no best solution for a multi-objective optimization problem (MOP), Pareto-optimal solutions are used. A set of Pareto-optimal solutions is called non-dominated solutions or Pareto-optimal front. The solutions in this set cannot dominate each other, in other words, there is no solution that is better than others in all objectives. Therefore, this paper attempts to find Pareto-optimal solutions to the proposed model.

3.3. Analysis of EEHFSP properties

Much research has proved that *TT* and *TEC* are mutually contradictory (Shao et al., 2022; Zhao et al., 2022b), however, the relationship between *TEC* and *CTC* is not clear. Here we give properties of the proposed EEHFSP to better indicate the conflict between *TEC* and *CTC*.

Property 1. *For a TOU strategy, there exist two solutions with the same energy consumption whose CTCs are the same but whose TECs are different.*

Proof. Here we give a simple example that satisfies this property. *CTC* is proportional to energy consumption. To better illustrate, the calculation of *CTC* is substituted by energy consumption.

In Fig. 4(a) $TEC = 6 \times 2 + 5 \times 5 + 4 \times 2 = 45$, $CTC = (2 + 5 + 2) \times 1 = 9$. In Fig. 4(b) $TEC = 6 \times 5 + 5 \times 2 + 4 \times 2 = 48$, $CTC = (5 + 2 + 2) \times 1 = 9$. It can be seen that the *CTCs* are the same, while *TEC* fluctuates with TOU tariffs.

Property 2. *For a TOU strategy, there exist conflicts between TEC and CTC.*

In Fig. 5(a) $TEC = 6 \times 5 + 5 \times 2 + 4 \times 2 = 48$, $CTC = (5 + 2 + 2) \times 1 = 9$. In Fig. 5 (b) $TEC = 6 \times 2 + 5 \times 2 + 4 \times 6 = 46$, $CTC = (2 + 2 + 6) \times 1 = 10$. It is observed that for two solutions the decrease of *TEC* incurs the increase of *CTC*.

The above properties of the proposed EEHFSP justify the necessity of considering both *TEC* and *CTC*. *TEC* reflects the energy cost while *CTC* reflects the carbon emission. The change of *TEC* is not always consistent with that of *CTC*.

The EEHFSP model can be denoted as $HFs(Qm_1, ..., Qm_k)|\text{TOU}, on-off|\{TT, CTC, TEC\}$. Gupta (1988) has proved that the HFSP with two stages with the makespan objective is NP-hard when the first stage has two identical parallel machines and the second stage has only one machine. Based on the complexity hierarchies for scheduling problems (Pinedo, 2016), the HFSP with total tardiness objective can be deduced to be NP-hard. Considering our problem contains k stages with uniform parallel machines, and we minimize three objectives concurrently along with two practical EES, i.e., TOU and power down, the proposed EEHFSP is an NP-hard problem.

Table 3
The states of a machine.

x_{ik}^{t}	y_{ik}^{t}	State	Power
1	1	Processing	pe_{ik}
1	0	Standby	se_{ik}
0	1	Reset	re_{ik}
0	0	Shutdown	0

4. Background of GVNS and Q-learning

4.1. GVNs

VNS is a metaheuristic that changes the neighborhood structures systematically to escape from the local optima. VNS jumps from the current solution to a new one only if a solution with higher quality has been found (Mladenovic and Hansen, 1997). The main loop includes a shaking procedure to escape from the local optimum, a local search to improve the solution, a neighborhood change procedure, and an acceptance procedure.

The VNS can easily be changed to other variants depending on the search depth and step length of neighborhood change. In our study, we refer to the general VNS (GVNS) wherein variable neighborhood descent (VND) is integrated into the local search procedure. VND adopts multiple local search operators in a sequential or nested fashion to improve the solution (Hansen et al., 2010; Shao et al., 2022). The detailed procedure is shown in **Algorithm 1.**

Algorithm 1: General variable neighborhood search

1: **Input:** Set of shaking neighborhood structures $N_k(k = 1, 2, ..., k_{\max})$, solution x, set of local search operators $N_l'(l = 1, 2, ..., l_{\max})$
2: **Output:** an improved solution x
3: **Set** $k = 1$
4: **While** $k < k_{\max}$
5: Generate a solution x' at random from the k-th neighborhood of x $(x' \in N_k(x))$
6: # Shaking procedure
7: Generate improved solution x'' using VND $(x'' \in N_l'(x), l = 1, 2, ..., l_{\max})$
8: # Local search procedure (Hansen and Mladenovic, 2001)
9: **If** x'' is better than the incumbent x **then**
10: Set $x = x''$
11: **Else**
12: Set $k = k + 1$
13: **End if**
14: **End while**

4.2. Q-learning

Q-learning, first developed by Watkins (1989), is a model-free and off-policy RL algorithm based on temporal difference, which converges to the optimum action-state value independent of the target policy being followed. At each step, the agent perceives the state s of the environment and chooses the best action a from a set of actions based on learning knowledge. Then the environment would move to the next state s' while at the same time gives feedback called reward. The agent tries to maximize the expected cumulative reward through trial-and-error interactions with the environment over time. In Q-learning, this expected

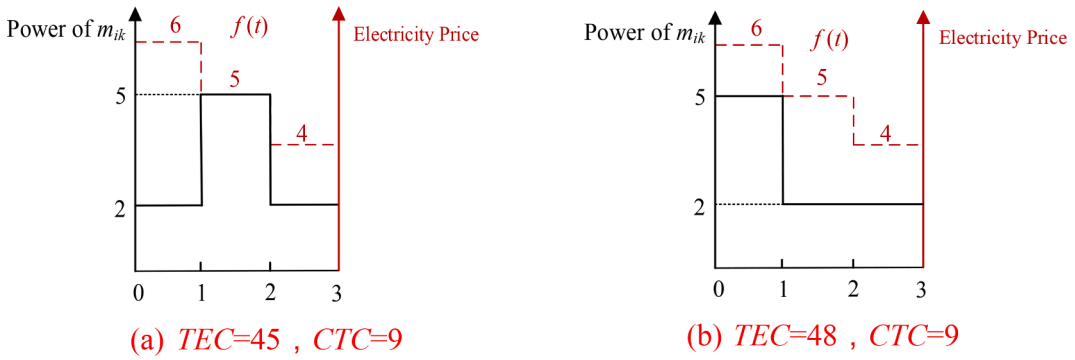


Fig. 4. Same *CTCs* and different *TECs*.

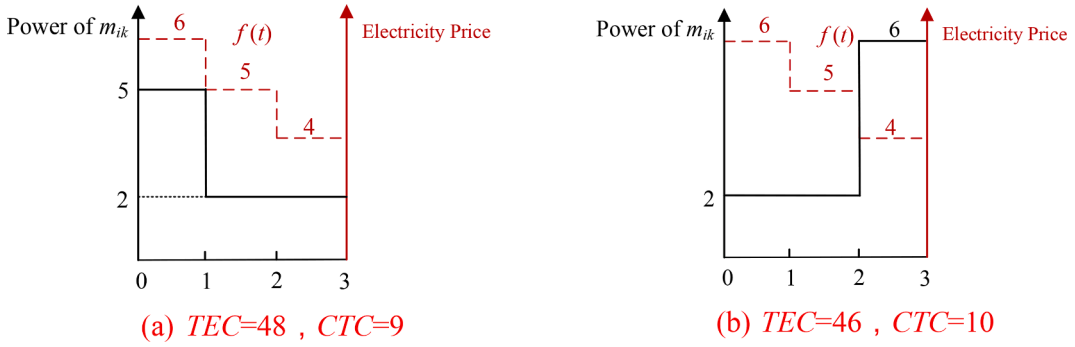


Fig. 5. *TEC* and *CTC* change in opposite directions.

cumulative reward of an action-state pair $(s,a) \in S \times A$ is estimated by action-state value $Q(s,a) \in \mathbb{R}$, representing the learned experience.

List all the actions in rows and states in columns, fill in each entry with $Q(s,a)$ related to (s,a), then we can get the brain of Q-learning, Q-table. The initial Q-table is a zero-value matrix as shown in Table 4. The agent will explore from state to state until the stop condition is satisfied. Each exploration is called an episode.

And the updating value of $Q(s,a)$ is shown as below (Watkins, 1989):

$$Q(s,a) \leftarrow Q(s,a) + \alpha\left[r + \gamma \max_{a'} Q(s',a') - Q(s,a)\right] \tag{21}$$

Where $s_t = s$ is the current state, a is the action performed, s' is the next state after a is performed, and a' is the next action in state s'. $\alpha(0 \leqslant \alpha < 1)$ is the learning rate controlling the ratio of accepted new information. r is the reward after performing action a, and it is calculated by the reward function. $\gamma(0 \leqslant \gamma \leqslant 1)$ is the discount factor determining the influence of the future reward $\max_{a'} Q(s',a')$.

During each iteration, the agent encounters an exploration and exploitation dilemma, where it needs to make a tradeoff between selecting the action with the maximum Q value so far, and giving a chance to execute other actions. Many selection methods are proposed (Karimi-Mamaghan et al., 2022), in which ε-greedy policy make a good balance using parameter ε. The detailed procedure is:

Table 4
The initial Q-table.

States	Actions				
	a_1	a_2	a_3	⋯	a_5
s_1	0	0	0	⋯	0
s_2	0	0	0	⋯	0
s_3	0	0	0	⋯	0
⋮	⋮	⋮	⋮	⋱	⋮
s_5	0	0	0	⋯	0

$$A(s) = \begin{cases} \text{any one}, & \text{with probability } \varepsilon \\ \max_{a'} Q(s',a'), & \text{with probability } 1 - \varepsilon \end{cases} \tag{22}$$

With the higher ε, the agent intends to select random actions to explore more in the search space, while the lower ε enhances the exploitation capability by selecting the action with the best performance so far.

5. The proposed QVNS-NSGA-II

It is well-known that the basic HFSPs are NP-Hard problems. Due to the trade-off of problem complexity and computing efficiency, they are widely solved by meta-heuristic algorithms, such as genetic algorithm, and tabu search (Lee and Loong, 2019). In addition to the basic machine allocation and job sequencing decisions in the HFSPs, the studied EEHFSP needs to make decisions in turn on/off decisions of uniform parallel machines under TOU considering multiple conflicting objectives.

Thus, this paper proposes a Q-learning and GVNS driven NSGA-II. The well-known NSGA-II is adopted as the multiple-objective optimization framework (Deb et al., 2002). We incorporate GVNS into the NSGA-II framework to improve the neighborhood exploitation ability of NSGA-II, which avoids being trapped into local optimum. Generally, GVNS conducts neighborhood change in a random way. We further propose a Q-learning driven GVNS, taking advantage of the learning knowledge of Q-learning.

Most of the metaheuristics designed for EEHFSP consider neither EES nor knowledge from the previous search. The main novelty of the proposed QVNS-NSGA-II in this paper lies in the Q-learning boosted GVNS based on search history and current search status. The Q-learning process enables AOS in selecting neighborhood structures and local search operators, which enhances the performance of QVNS-NSGA-II. Furthermore, for the purpose of saving energy, an EES operator and problem-specific local search operators are first combined. To the best of

our knowledge, this paper is among the first to propose Q-learning and GVNS driven NSGA-II for EEHFSP. The general flow chart of the proposed QVNS-NSGA-II is given in Fig. 6.

5.1. The NSGA-II framework

5.1.1. Encoding and decoding

Encoding aims to transform the complete schedule into chromosomes. The encoding scheme for the EEHFSP problem needs to contain the following decisions: 1) Allocate jobs to machines at each stage. 2) Sequence the assigned jobs on each machine. 3) Determine turn on/off of uniform parallel machines at each stage. Decoding is the inverse process of encoding, which turns chromosomes into solutions. It also can determine 1) The standby/reset state of idle machines, and 2) The start time and completion time of jobs under TOU.

A native and direct way to encode the scheme includes multiple segments of allocation, sequencing and turn-on/turn-off decisions for all stages (Naderi et al., 2010, Yue et al., 2023). The complicated chromosomes will incur high computational cost and poor algorithm performance (Ruiz and Maroto, 2006). For the sake of simplicity and efficiency, we adopt an effective encoding scheme for the EEHFSP problem. The encoding scheme only encodes job permutation $\pi = \{1, 2, \ldots, n\}$ at the first stage and generates job permutation of later stages using List scheduling (LS). LS is widely employed for decoding HFSP in literature (Luo et al., 2013; Ruiz and Maroto, 2006; Yu et al., 2018).

LS leverages the earliest completion time (ECT) rule to dispatch jobs onto machines. Note that in uniform parallel machine environment, ECT will generate different schedules from the first available machine (FAM) rule. Even if a job is allocated to the first available machine, its completion time is likely to be larger than that of ECT rule because uniform parallel machines run at different speeds (Naderi et al., 2010).

Under LS, every time jobs are taken one by one in the list and assigned to a machine that can complete the job the earliest. This coincides with the minimization of *TT*. The job lists at later stages are updated based on the completion times of jobs at the previous stage.

Let π_k denotes the job permutation at stage k and $\pi_k(q)$ denotes the job at position q in π_k. To convert a chromosome into a feasible schedule, the procedure of decoding is shown below.

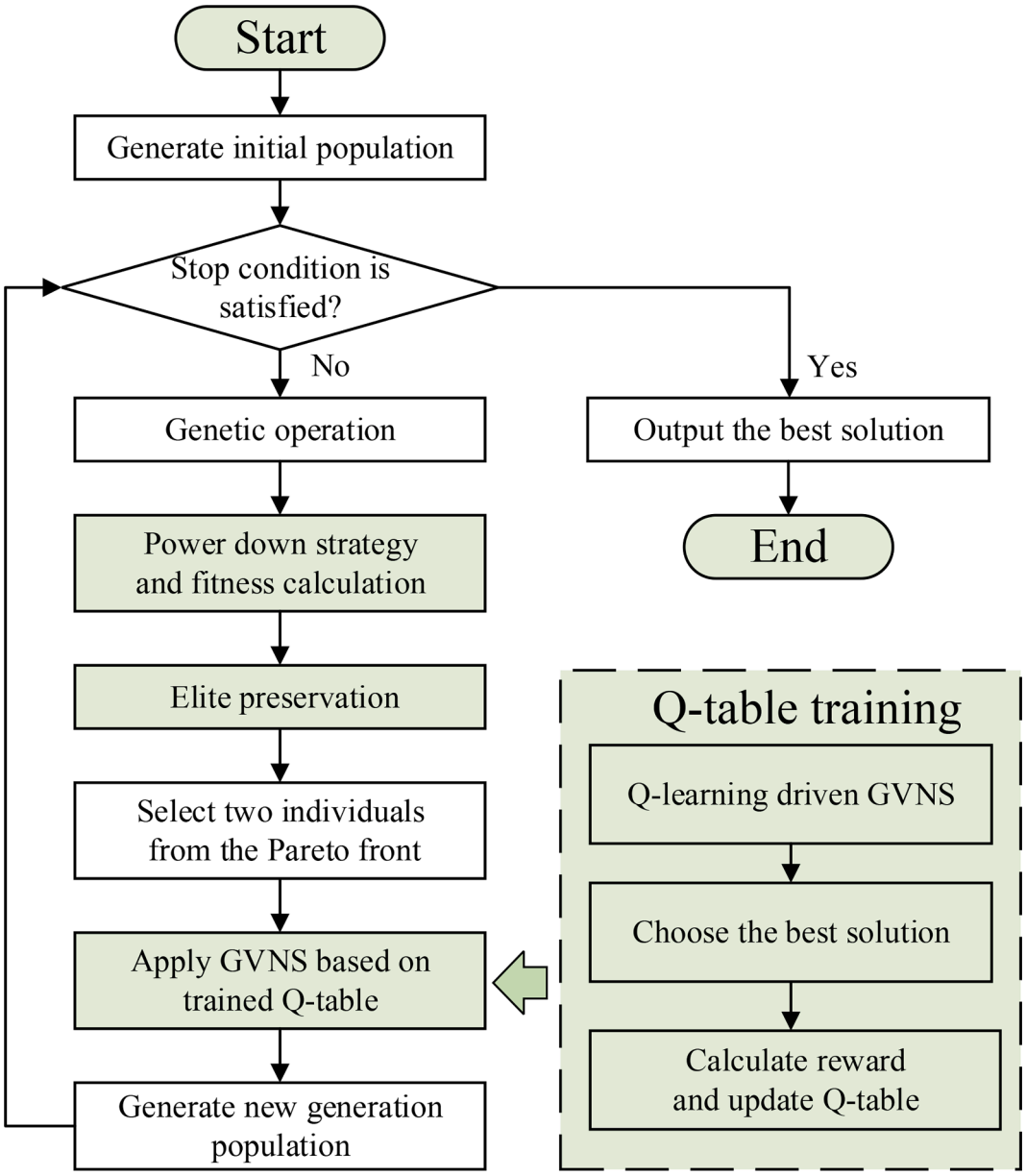


Fig. 6. Flow chart of the QVNS-NSGA-II.

According to Naderi et al. (2010), the complexity of decoding is calculated as $O(n^2\sum_{k=1}^{m} l_k)$, where l_k is the number of parallel machines at stage k.

Algorithm 2: LS decoding

1: **Input:** The sequence of chromosome π_1
2: **Output:** a feasible schedule
3: **For** $q = 1, 2, ..., n$ **do**
4: Assign the job $\pi_1(q)$ to the machine that can complete the job the earliest
5: **End for**
6: **For** $k = 2, ..., m$ **do**
7: Sort the jobs in non-decreasing order of their completion time at the previous stage $k-1$ and create a new sequence π_k
8: **For** $q = 1, 2, ..., n$ **do**
9: Assign the job $\pi_k(q)$ to the machine that can complete the job the earliest
10: **End for**
11: **End for**

5.1.2. Initialization

The NSGA-II begins with a population of initial individuals. The quality of the initial population is of vital importance to the algorithm's performance. The NEH from Nawaz et al. (1983) was recognized as the most successful heuristic in PFSP, and it is well adapted to HFSP (Naderi et al., 2010). Inspired by the work of Pan et al. (2014), we design a specific initialization method based on NEH for EEHFSP. We extend NEH with different optimization objectives to generate initial individuals. This objective can be any one of *TT*, *TEC* and *CTC*. The detailed procedure is presented in **Algorithm 3**.

Algorithm 3: NEH heuristic

1: **Input:** A set of jobs n
2: **Output:** a feasible schedule π
3: Generate a job sequence π_R by decreasing processing time of jobs
4: Take the first two jobs and schedule them as π to minimize a certain objective
5: **For** $i = 3, ..., n$ **do**
6: Take job $\pi_R(i)$ and insert it into all the possible places
7: Update π the individual with the lowest objective value
8: **End for**

The initialization includes two steps.

(1) Step 1: Yield three individuals based on **Algorithm 2**. One individual use *TT* as the minimization objective of NEH, one individual use *TEC*, and the remaining one uses *CTC*.
(2) Step 2: The rest individuals in the population are generated randomly.

This approach provides the algorithm with three excellent individuals in different directions of search space, which avoids an early convergence for the lack of diversity and slow convergence due to the poor quality of random solutions (Pan et al., 2014).

In **Algorithm 3**, we have $n(n+1)/2-1$ insertions in total (Nawaz et al., 1983) and each insertion requires an evaluation. The complexity of NEH heuristic is $O(n^3\sum_{k=1}^{m} l_k)$, where $\sum_{k=1}^{m} l_k$ denotes the total number of machines (Taillard, 1990).

5.1.3. Genetic operation

The genetic operation, namely *crossover* and *mutation*, act as the backbone to explore solution space and escape from local optimum during evolution. Considering the orderliness of the job sequence, hereby, *two-point order crossover* (2p-OXO) and *swap sequence mutation* are employed.

The 2p-OXO has been widely employed in FSP, which proved to have excellent performance (Lu et al., 2022; Pan et al., 2022). As shown in Fig. 7, we used an improved variation of classic 2p-OXO, wherein the cut points in both parents can be at different positions (Syswerda, 1991).

Swap mutation is a simple and effective mutation operator (Lu et al., 2022; Zhao et al., 2021b). Hereby, we extend the swap segment from one unit to a sequence. One of the offspring is randomly selected for mutation (Fig. 8).

5.1.4. Elite preservation

Elite preservation strategy aims to retain the best individuals (i.e., elites). To evaluate the quality of multi-objective solutions, fast non-dominated sorting (FNS) and crowding-distance calculation are adopted in Appendix A. The detailed procedure refers to Deb et al. (2002).

The *Pareto Dominance Operators* for two solutions are defined. Operators $\prec$ and $\succ$ mean "inferior to" and "superior to", respectively. In minimized optimization problems, the smaller the objective value, the better the solution, and vice versa.

Definition 1. (Deb et al., 2002): *If a solution* x_1 *Pareto dominates another solution* x_2, *it subjects to*:

$$f_j(x_1) \prec f_j(x_2), \forall j \in \{1, 2, 3, ...\} \tag{23}$$

where $f_j(\cdot)$ denote the objective function.

Based on this Pareto dominance operator, FNS assigns each solution

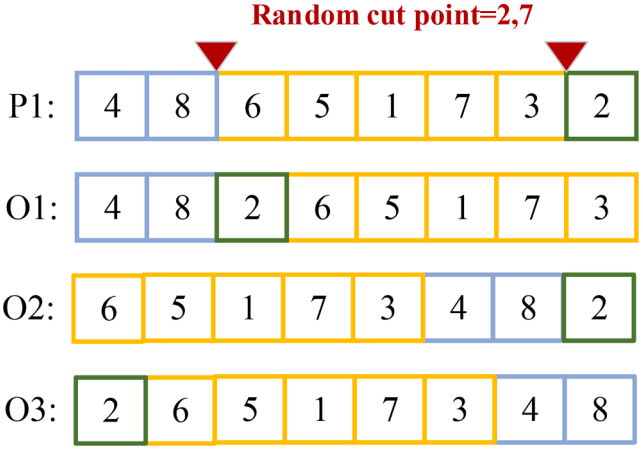


Fig. 8. Swap sequence mutation.

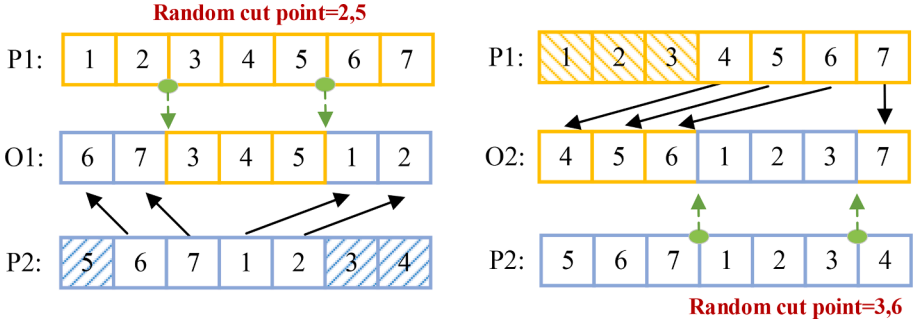


Fig. 7. Two-point order crossover.

a rank representing the non-dominated level or Pareto fronts through the comparison with other solutions. Each front is dominated by the preceding one, for example, rank 1 is the best level (F_1); rank 2 is the second level (F_2), and so on.

FNS designs two entities for each solution x, namely dominance count and the set of solutions dominated by x, decreasing the computational complexity of the naive sorting approach from $O(MN^3)$ to $O(MN^2)$, where M denotes the number of objectives and N denotes the number of individuals in a population.

Elite preservation first combines the parent population P_t and offspring population Q_t to form $P_t \cup Q_t$. Next, all solutions in $P_t \cup Q_t$ are sorted into different Pareto fronts using FNS. The next population P_{t+1} is filled with F_1, F_2 to the last front F_l in turn until the required population number N is exceeded. To eliminate the number of individuals to exactly N, the *crowding-distance* is introduced.

Crowding-distance indicates an estimate of the proximity or density of solutions surrounding a particular solution in the population. A solution with a higher crowding-distance suggests a less crowded neighbor region, which is preferred for diversity preservation. To preserve just N individuals for P_{t+1}, we sort the solutions in F_l using the *Crowded-Comparison Operator* $\prec_n$ and $\succ_n$. These operators use both rank and crowding-distance, which help to find a Pareto front with evenly distributed solutions.

Definition 2. *If a solution i is superior to j, it subjects to* (Deb et al., 2002):

$$i \prec_n j : if(i_{rank} < j_{rank}) or (i_{rank} = j_{rank} and i_{distance} > j_{distance}) \tag{24}$$

where i_{rank} and $i_{distance}$ are the rank and crowding-distance of solution i, respectively.

Here we present a demonstration of elite preservation in Fig. 9.

5.2. Power down strategy

In the original NSGA-II, LS decoding method considers only time-related objective *TT*. In order to reduce both *TEC* and *CTC*, we resort to power down mechanism to identify which idle machine should be shut down.

We exemplify the power down mechanism with the Gantt charts in Fig. 10. The green dashed box represents that the machine stays idle between two jobs, while the grey dashed box represents the shutdown and reset states of the machine.

As we can see, the power down mechanism turns down the machine m_{12}, m_{22}, m_{32}, m_{33} for a while, reducing the electricity cost without affecting *TT*. The mechanism determines whether it saves costs to shut down the machine or keep idle. Meanwhile, the machine will not be shut down if there is no available time to reset.

The power down mechanism dynamically adjusts the states of machines to reduce amounts of energy. We calculate the *inter-arrival time* between the q-th job and the $q+1$-th job on m_{ik} using the following Eq. (25). Let π_{ik} denote the job sequence on machine m_{ik}, and $\pi_{ik}(q)$ denotes the job at position q in m_{ik}.

$$TI_{ik\sigma_{ik}(q)\sigma_{ik}(q+1)} = C_{\sigma_{ik}(q+1)k} - \frac{p_{\sigma_{ik}(q+1)k}}{v_{ik}} - C_{\sigma_{ik}(q)k}, i \in M_k, k \in S \tag{25}$$

Eq. (26) defines the *break-even time* of each machine m_{ik}.

$$TB_{ik} = \frac{t_{reset} e^r_{ik}}{e^s_{ik}}, i \in M_k, k \in S \tag{26}$$

TB_{ik} of each machine m_{ik} is the threshold value when reset energy consumption equals standby energy consumption. When the condition $TI_{ik\sigma_{ik}(q)\sigma_{ik}(q+1)} > TB_{ik}$ is satisfied, the machine m_{ik} should be shut down, otherwise, it would consume more energy in standby state than in reset state. Note that each machine m_{ik} has its corresponding break-even time TB_{ik}.

In order to implement this operator in QVNS-NSGA-II, a detailed procedure is introduced. The time complexity of **Algorithm 6** is $O(nm)$ since we have a total of nm operations.

Algorithm 6: Power down mechanism

Input: a feasible schedule π
Output: an improved schedule π' for energy saving
For each machine m_{ik} **do**
For $q < |\pi_{ik}|$ **do**# the q-th job on machine m_{ik} is not the last one
If $TI_{ik\pi_{ik}(q)\pi_{ik}(q+1)} \geqslant \max\{TB_{ik}, t_{reset}\}$ **then**
Shut down the machine m_{ik} when the job $\pi_{ik}(q)$ is finished; that means for the time $C_{\pi_{ik}(q)k} < t \leqslant S_{\pi_{ik}(q+1)k} - t_{reset}$, let $x^t_{ik} = 0 y^t_{ik} = 0$. m_{ik} is supposed to turn on before the job $\pi_{ik}(q+1)$ begins; that means $S_{\pi_{ik}(j+1)k} - t_{reset} < t \leqslant S_{\pi_{ik}(j+1)k}$, let $x^t_{ik} = 0, y^t_{ik} = 1$.
Else
Keep the machine idle during inter-arrival time $TI_{ik\pi_{ik}(q)\pi_{ik}(q+1)}$,
So $x^t_{ik} = 1, y^t_{ik} = 0, C_{\pi_{ik}(q)k} < t < S_{\pi_{ik}(q+1)k}$
End if
End for
End for

5.3. Neighborhood structures and local search

As mentioned in Section 4.2, GVNS is composed of a shaking procedure and a local search procedure. Both procedures require a predefined neighborhood structure set, for clarity, we use the term "neighborhood structure" in the shaking procedure and "local search operator" in the local search procedure, specifically.

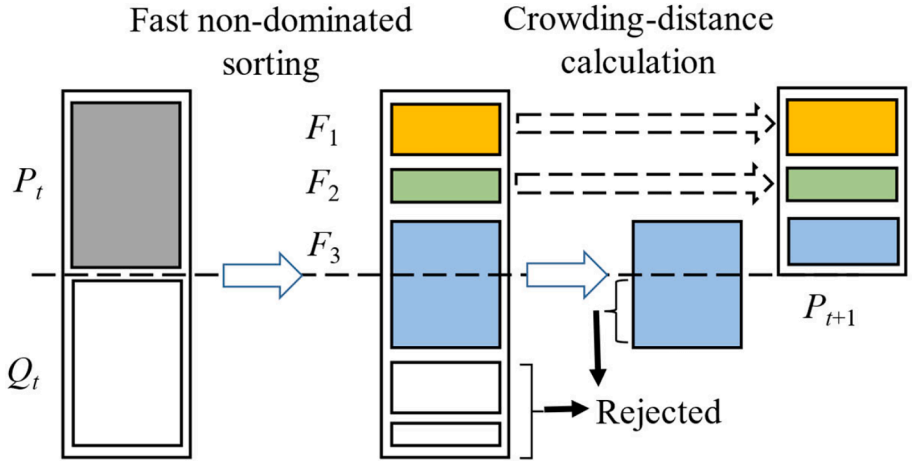


Fig. 9. Elite preservation.

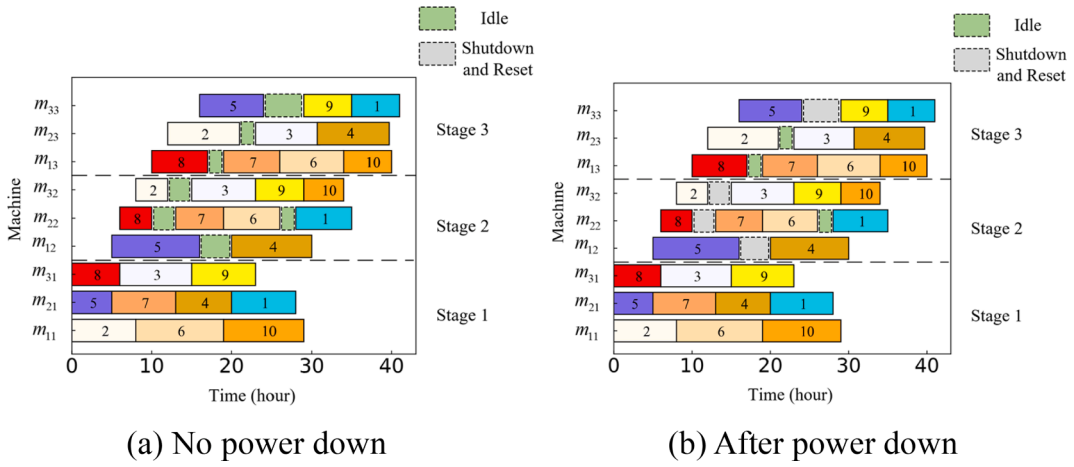


Fig. 10. The effect of power down mechanism.

5.3.1. Neighborhood structure

In the shaking procedure, we consider a simple perturbation operator as our neighborhood structure, namely Destruction-Construction (DC). This operator is widely used in different metaheuristics such as Iterated Greedy (Karimi-Mamaghan et al., 2023) and Adaptive Large Neighborhood Search (Gendreau and Potvin, 2019). The original perturbation operator includes two parts: the destruction phase which removes d jobs randomly from job sequence π, and the construction phase to repair the sequence using NEH heuristic (Ruiz and Stützle, 2007).

The number of jobs possible to be removed ranges from 1 to a limit $d_{\max}$, which equals the number of possible neighborhood structures (Karimi-Mamaghan et al., 2023). To avoid expensive evaluation cost caused by iteration through all possible insertions, we adapt the construction phase, using a first-improvement pivot rule (Naderi et al., 2010). The inner search process will be terminated either an improved solution is found or iterations are over a limit *Max_iter*.

The detailed pseudocode is shown as follows.

Algorithm 7: Pareto-based Destruction-Construction

Input: a feasible schedule π, the number of removed jobs $d \in [1, d_{\max}]$, maximum iterations without improvement *Max_iter*.
Output: an improved schedule π'
Define an empty set π_R to store the reinserted jobs, $\pi' = \pi$
For $i = 1$ to d **do**
  Remove one job from π' and insert into π_R
End for # Destruction phase
Define the number of no improvement as n_n, and set $n_n = 0$, $i = 1$
For $1 \leqslant i \leqslant d$ **do**
  Set $n_n = 0$, improve = False
  While$n_n <$ *Max_iter* **do**
    Insert job $\pi_R(i)$ into the π' randomly without repetition to get π''
    If $\pi'' \prec \pi'$ **then # Pareto-dominance operator**
      Set $\pi'=\pi''$, improve = True
      break # jump out of the while loop
    Else
      Set $n_n = n_n + 1$
    End if
  End while
  If not improve **then**
    Insert job $\pi_R(i)$ into the π' randomly
  End if
End for # Construction phase

5.3.2. Local search operator

Local search operators are critical to improving solutions, however, general single-objective methods cannot be directly applied to the trade-offs of *TT*, *TEC* and *CTC*. Insertion and pairwise swap are widely used in the literature, and we modify the classic methods to problem-specific ones. We adopt knowledge-based operators critical-path-based local search (Zhao et al., 2022b) to avoid blind search while at the same time further improve the solutions (Wang and Wang, 2016). Besides, two effective local search operators, namely three-point permutation and three-segment permutation, are selected (Asefi et al., 2014).

Fig. 11 illustrates an example of the critical path that is pointed out by arrows.

Jobs in the critical path are defined as critical jobs J_C (i.e., job 8, 3, 9, 1), and the other jobs are called non-critical jobs J_R. When the critical path is identified, select one job from J_C and one job from J_R, and the following local search can be defined:

(1) Critical swap (CSwap): Swap the position of J_C and J_R in π, see Fig. 12(a).
(2) Critical insertion (CInsr): Insert J_C to the position just after J_R in π, see Fig. 12(b).
(3) Critical inverse (CInv): Inverse the jobs between J_C and J_R, see Fig. 13.
(4) Three-point permutation (TPP): Randomly choose three adjacent genes at any position and perform all of the possible permutations, see Fig. 14(a). The best solution is then used.
(5) Three-segment permutation (TSP): Randomly choose two cut points on the chromosome and split it into three distinct segments. Then perform permutations of the three segments, and generate a total of 5 possible new permutations, see Fig. 14(b). Finally, evaluate these solutions and select the best one.

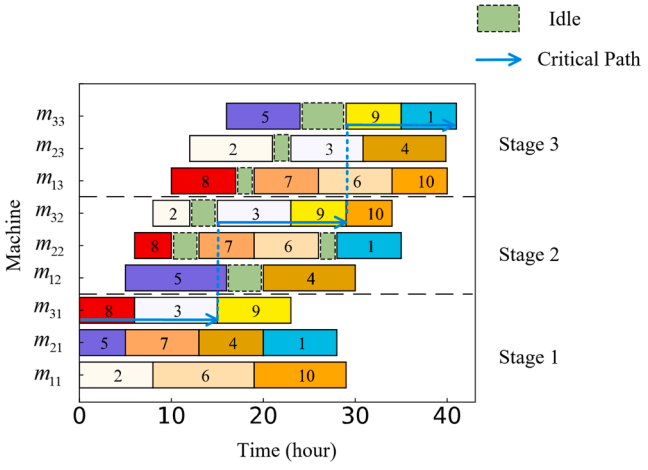


Fig. 11. Critical path.

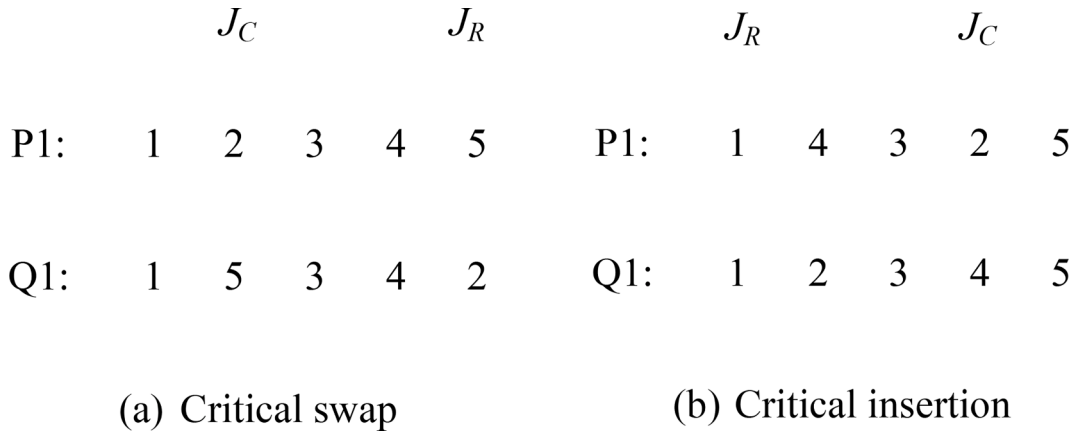


Fig. 12. Critical swap and Critical insertion.

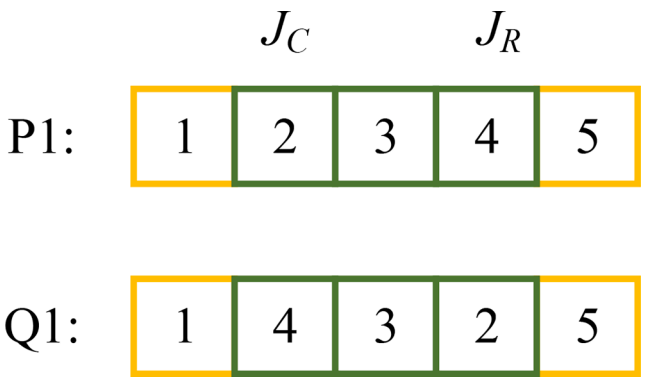


Fig. 13. Critical inverse.

5.4. Q-learning driven GVNS

In the proposed QVNS-NSGA-II algorithm, GVNS is driven by Q-learning to select the most appropriate neighborhood structure and local search operator during the evolution process. The aforementioned Q-learning process requires a set of states/actions and reward function. Fig. 15 gives an overview of the QVNS.

5.4.1. States and actions

The set of states and actions define the environment that the agent can perceive and take response to. Zhao et al. (2021) and Li et al. (2022) set the individuals as states to construct Q-table, leading to a tremendous state space that can hardly be explored by training. Different from theirs, we set the neighborhood structures and local search operators as states. Indeed, the state space is limited and effective to reflect a solution.

Since the shaking and local search procedures are separate, we use $S1$ and $S2$ to denote their sets of states respectively.

- State $S1$: All possible states describing the shaking procedure. States are $d_{\max}$ neighborhood structures in Section 5.3.1. $S1 = \{1, 2, \ldots, d_{\max}\}$.

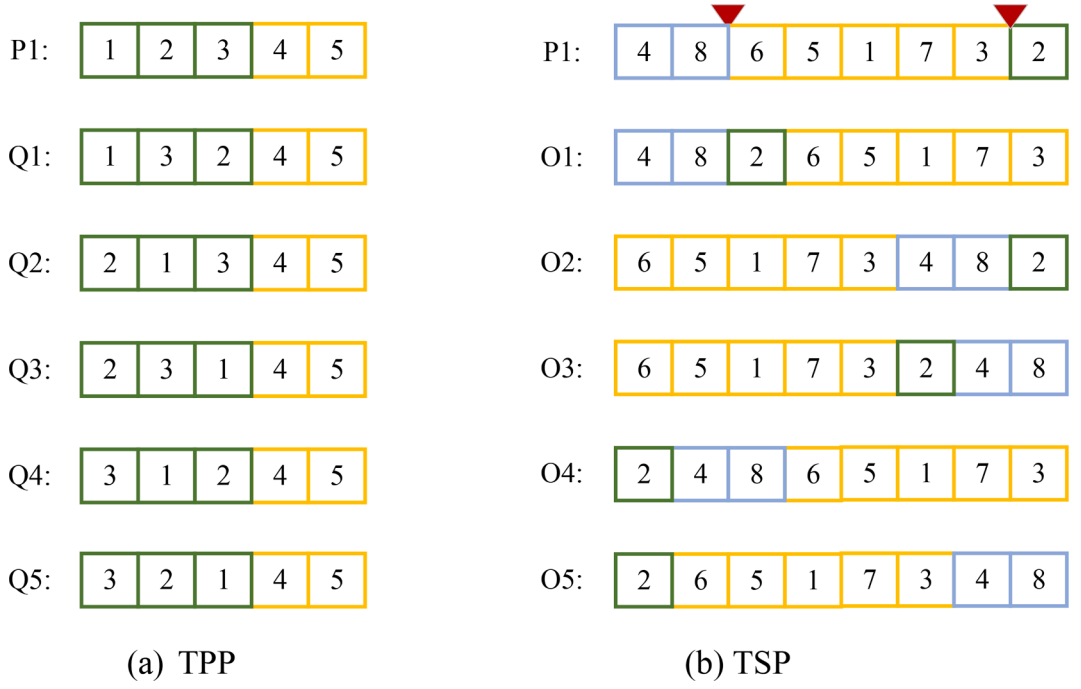


Fig. 14. Three-point permutation and three-segment permutation.

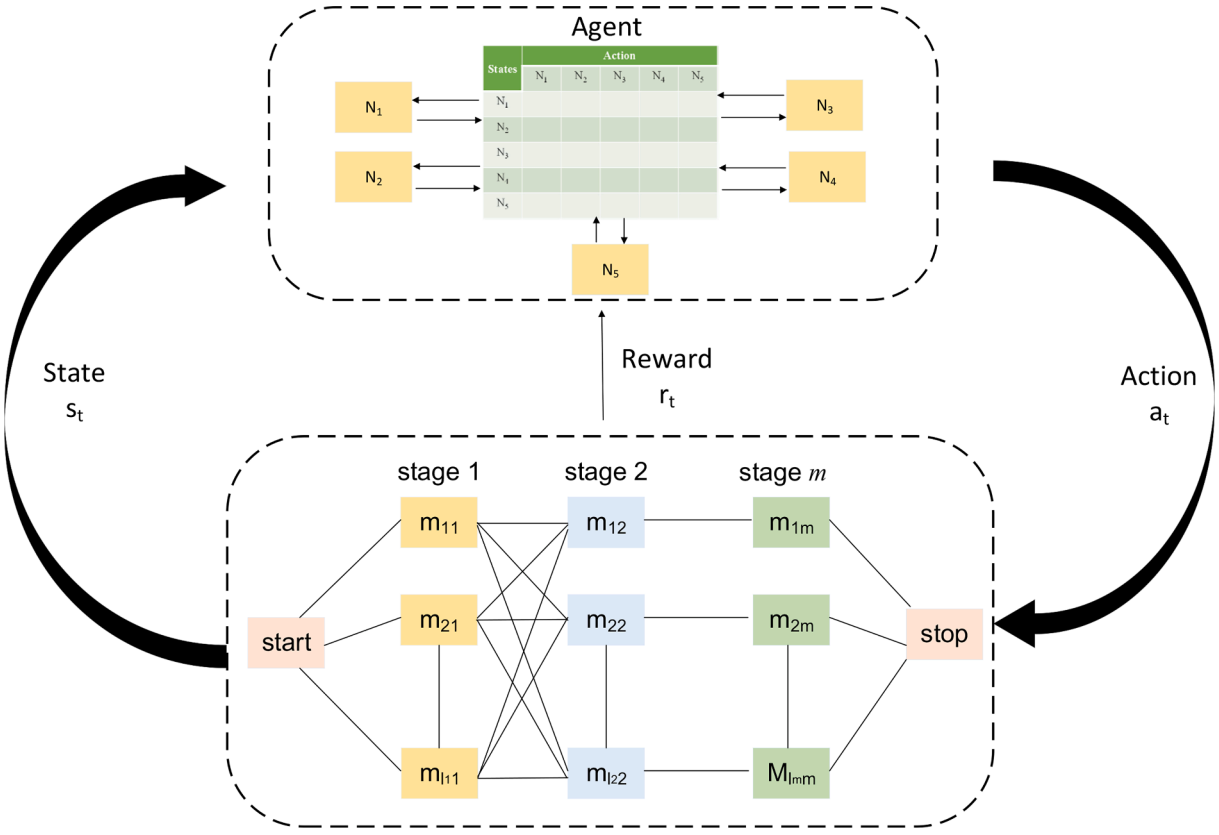


Fig. 15. The illustration of QVNS.

- State $S2$: All possible states describing the local search procedure. States are five local search operators in Section 5.3.2. $S2 = \{\text{CSwap}, \text{CInsr}, \text{CInv}, \text{TPP}, \text{TSP}\}$.

The action set is as same as the corresponding state set, which means $A1 = S1$ and $A2 = S2$. Fig. 16 presents an example of state set $S2$ and actions set $A2$, where the arrow indicates a transfer from one state to another state ("go-to").

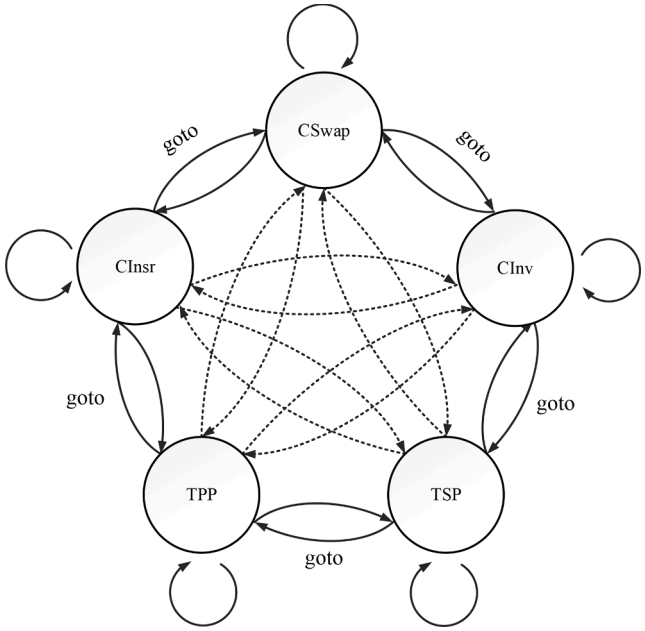


Fig. 16. States $S2$ and Actions $A2$ of Q-learning.

The current action of an agent depends on the last neighborhood it adopts. After executing a shaking or local search, a reward is calculated based on the improvement of solutions. Then Q-value is updated, instructing which neighborhood to transfer.

5.4.2. Reward function

During the learning process, each operator should be allocated a reward, which is immediate feedback from the application of an operator.

Durgut et al. (2021) proposed a reward function that overcomes the degeneration or disruption caused by the immediate result. However, it is only applicable to single objective evaluation. In order to consider multiple objectives in reward assignments, we adapt their reward function to a multi-objective version by normalization and summation. For a minimization problem, the reward is defined as:

$$r_t = \sum_k \frac{f_{k\min}}{f_k(x_t)} \left(\frac{f_k(x_t) - f_k(x_{t+1})}{f_{k\max} - f_{k\min}} \right) \tag{27}$$

where x_t is the initial solution at time t, and x_{t+1} is the new solution after applying a neighborhood structure or local search operator. $f_k(\bullet)$ denote the k-th objective function, $f_{k\max}$ and $f_{k\min}$ denote the worst objective value and the best objective value found so far, respectively.

The reward ensures that if a solution is close to the best objective, it will get larger r_t; the more a new solution is improved, the larger its reward.

5.4.3. The procedure of the proposed QVNS

Algorithm 8 presents the details of GVNS with Q-learning process, which consists of 2 steps. Step 1 is a shaking procedure as well as Q1 training. After the execution of shaking, it evaluates the improvement of solutions and updates Q1 to select the next action-state. Step 2 incorporates Q-learning into VND local search, applying all operators in S2

without repetition and updating Q2 to select the next action-state.

Algorithm 8: **GVNS with Q-Learning process**

1: **Input:** parameters learning factor α, discount factor γ, Epsilon-greedy factor ε, initial solution x_0, state set $S1$, $S2$, action set $A1$, $A2$, the number of episodes E, Maximum iterations without improvement *Max-iter*
2: **Output:** an improved solution x, trained Q-table
3: Initialize Q-table Q1, Q2 as zero matrices, Q1=$|S1 \times A1|$, Q2=$|S2 \times A2|$
4: Initialize the global best solution $x^* = x_0$
5: Remember the global best objective f_{k0}^* and local optimum $f_k(x_0)$,$k = 1,2,3$
6: Select an action $a_1 = d$ at random from $A1$, and set the initial state $s_1 = d$
7: Select an action $a_2 = N$ at random from $A2$, and set the initial state $s_2 = N$
8: **For** $t = 1 : E$ **do # For each episode**
9: **#Step 1: Shaking procedure and training Q1**
10: x = Pareto-based Destruction-Construction (x_0, a_1, Max_iter)
11: **If** $x \prec x_0$ **do**
12: **# Update the Q1 and action-state for the next episode**
13: Calculate the reward r using equation (27)
14: Q1 = Q-learning $(\alpha, \gamma, \varepsilon, s_1, a_1, r)$ using equation (21)
15: s_1, a_1= Epsilon-greedy (Q1, s_1, a_1) using (22)
16: $x_0 = x$, remember the local optimum $f_k(x_0) = f_k(x)$
17: **End if**
18: **If** $x_0 \prec x^*$ **do # Identify the global best solution**
19: $x^* = x_0$, remember the global best objective $f_{kt}^* = \min\left(f_{k(t-1)}^*, f_k(x_0)\right)$
20: **End if**
21: **#Step 2: VND Local search procedure and training Q2**
22: **While** $S2 \neq \varnothing$ **# Apply every local search operator without repetition**
23: x = Local search (x_0, a_2)
24: **If** $x \prec x_0$ **do**
25: **# Update the Q2 and action-state for the next episode**
26: Calculate the reward r using equation (27)
27: Q2 = Q-learning $(\alpha, \gamma, \varepsilon, s_2, a_2, r)$ using equation (21)
28: $x_0 = x$, remember the local optimum $f_k(x_0) = f_k(x)$
29: **Else # When there is no improvement, choose the next action**
30: Remove s_2, a_2 from $S2$,$A2$
31: s_2, a_2= Epsilon-greedy (Q2, s_2, a_2)
32: **End if**
33: **If** $x_0 \prec x^*$ **do # Identify the global best solution**
34: $x^* = x_0$, remember the global best objective $f_{kt}^* = \min(f_{kt}^*, f_k(x_0))$
35: **End if**
36: **End while**
37: **End for**

The complexity of the proposed QVNS is analyzed as follows.

The complexity of the shaking procedure. According to Karimi-Mamaghan et al. (2023), the complexity of destruction and construction (lines 9–10) is $O\left(d + dn\sum_{k=1}^{m} l_k\right)$ as we randomly pick d jobs and reinsert them into n possible positions. $\sum_{k=1}^{m} l_k$ denotes the total number of machines. The worst-case complexity of Q-learning process from lines 12–16 is $O(d_{\max})$ because there are at most $d_{\max}$ updates using equation (21).

The complexity of the local search procedure. The only difference between local search and shaking is in Line 24. Since we have five local search operators, they require $O\left(n\sum_{k=1}^{m} l_k\right)$ when calculating the critical path. The complexity of Q-learning update from lines 26–29 is $O(5)$.

The complexity of the proposed QVNS. Considering we have E episodes in total, the total complexity is: $O\left(E\left(d + dn\sum_{k=1}^{m} l_k + d_{\max} + n\sum_{k=1}^{m} l_k + 5\right)\right)$. Therefore, the complexity of QVNS is $O\left(n\sum_{k=1}^{m} l_k\right)$.

6. Computational experiments and results

This section conducts a series of computational experiments to testify the proposed algorithm. First, performance indicators are defined to measure the algorithm's performance. Since EEHFSP lacks standard benchmark instances, test instances are randomly generated. Then, parameter tuning experiments are conducted to determine the key parameters for the proposed QVNS-NSGA-II. Third, QVNS-NSGA-II is compared to classical NSGA-II (Deb et al., 2002) and two state-of-the-art multi-objective evolutionary algorithms (MOEA), namely improved Jaya (Pan et al., 2022) and modified MOEA/D (Wang et al., 2021). Finally, sensitivity analysis experiments are conducted to present managerial insights.

6.1. Experiment settings

To the best of our knowledge, there is limited previous research with benchmark instances on the EEHFSP problem. Test instances are randomly generated. Each test instance is denoted as the "number of jobs-number of stages-number of machines at each stage" (Luo et al., 2013). For example, a test instance with 10 jobs, 3 stages, and 4 machines at each stage is denoted as "10-3-4". The detailed experiment settings are given below:

(i) Due date d_j is determined by the following formula (Ding et al., 2021):

$$d_j = \max\left(0, U\left[P\left(1 - \tau - \frac{R}{2}\right), P\left(1 - \tau + \frac{R}{2}\right)\right]\right) \tag{28}$$

where P, τ and R represents the lower bound of makespan, tardiness factor and due date factor respectively. denotes the nearest integer function. $\tau \in \{0.2, 0.4\}$ and $R \in \{0.6, 1.0\}$.

(ii) The TOU electricity function is given (Development and Reform Commission of Jiangsu Province, 2021):

$$f(t) = \begin{cases} 25024n \leqslant t < 24n + 8 \\ 100024n + 8 \leqslant t < 24n + 11 \\ 60024n + 11 \leqslant t < 24n + 17 \\ 100024n + 17 \leqslant t < 24n + 22 \\ 60024n + 22 \leqslant t < 24n + 24 \end{cases} \text{(CNY/MWh)}, \quad n \in \{0, 1, 2, 3, \ldots\}.$$

The other parameters are shown in Table 5.

The quality of MOEA depends on convergence and diversity. The former reflects the distance between the obtained front and the optimal Pareto front, and the latter requires a more even distribution of solutions. Here we adopt the following indicators to compare these two aspects:

(i) Coverage metric (CM) (Ding et al., 2016): This indicator indicates the percentage of solutions in the Pareto set B dominated by at least one solution in the Pareto set A. The closer the CM value is to 1, the better set A is. CM is calculated by:

$$C(A, B) = \frac{|\{b \in B | \exists a \in A : a \succ bora = b\}|}{|B|} \tag{29}$$

The value $C(A, B)$ reflects the dominance relationship between two solution sets. If all the solutions of B are dominated by some solution of A, then $C(A, B) = 1$. Since some solutions in A and B are not dominated by each other, $C(A, B)$ and $1 - C(B, A)$ are not necessarily equal.

(ii) Number of Pareto solutions (NPS): NPS is equal to the number of non-dominated solutions of the Pareto front. A larger NPS indicates a more diverse Pareto front.

(iii) Spacing matrix (SM) (Wang et al., 2017): SM aims to evaluate

Table 5
Test instance parameter settings.

Factors	Levels
Number of jobs	10, 20, and 50
Number of stages	3, 5
Number of machines at each stage	3, 4, 6
Processing time of each operation	$U[5, 10]$(hour)
Power of machine	$U[5, 10](10^5$ W)
Processing speed	{1.2, 1.0, 0.8} (Mansouri et al., 2016)
Conversion rate	{1.5, 1.0, 0.6} (Mansouri et al., 2016)
Standby power of machine	2
Reset power of machine	4
Carbon emission coefficient	0.2 (ton/MWh)
Price of carbon emissions	30 (CNY/ton)
Emission Allowance	1 ton/(job·stage)

the diversity of the obtained Pareto solutions (uniformly distributed in the front). A smaller SM suggests a more even spread of the solutions in a Pareto front. SM can be calculated by the following equation

$$\mathrm{SM} = \sqrt{\frac{\sum_{i=1}^{A}(d_i - \overline{d})^2}{|A|}} \tag{30}$$

where d_i is the distance measure, which is the minimum value of the sum of the absolute difference in normalized objective function values between the i-th solution and any other solution in the obtained non-dominated set A.

$$d_i = \min_{j \in A \wedge i \neq j} \left(\sum_{k=1}^{M} |f_k^{'}(i) - f_k^{'}(j)| \right) \tag{31}$$

$\overline{d_i}$ is the mean value of d_i, and it is calculated by:

$$\overline{d} = \frac{\sum_{i=1}^{A} d_i}{|A|} \tag{32}$$

$f_k^{'}(\cdot)$ denotes the normalized objective value of k-th objective of individual x. $f_{k\max}$ and $f_{k\min}$ represent the maximum and minimum value of the objective function f_k in all tests.

$$f_k^{'}(x) = \frac{f_k(x) - f_{k\min}}{f_{k\max} - f_{k\min}}, k = 1, 2, 3 \tag{33}$$

6.2. Parameter tuning

The QVNS-NSGA-II contains five significant parameters: population size *Psize*, crossover rate *Pc*, mutation rate *Pm*, stop condition of QVNS *Q-iter* and CPU time ($CT \times n \times m$ *second*). *CT* denotes cycle time; n and m are the numbers of jobs and stages. Three-level Taguchi method DOE experiments (Pan et al., 2022) of these parameters are correspondingly conducted using a moderate-scaled instance "20–3-3". Each parameter is regarded as a factor, and three factor levels are considered for each factor, see Table 6.

The orthogonal array $L_{27}(3^5)$, listed in Table 6, is selected with five factors, each at three levels. Without loss of generality, the other parameters are set, i.e., the number of generations $Max\_Gen = 100$, learning factor $\alpha = 0.1$, epsilon factor $\varepsilon = 0.1$, discount factor $\lambda = 0.1$, maximum DC depth$d_{max} = 8$. The performance indicator SM is used as the response indicator (Table 7).

The main effect plot of parameters is shown in Fig. 17. It can be observed that *CT* and *Q-iter* have more significant effects than *Pc*. The parameters of the QVNS-NSGA-II are set as population size $Psize = 120$, crossover probability $Pc = 0.8$, mutation probability $Pm = 0.3$, $Q\text{-}iter = 3$, and $CT = 0.8$.

For a fair comparison, all four algorithms adopt the crossover and mutation method. All algorithms have the same population size (*Psize*) of 120 and termination condition that $CT \times n \times m$ is met.

In NSGA-II (Deb et al., 2002), crossover rate(*Pc*) and mutation rate (*Pm*) are set as 0.8 and 0.3. In improved Jaya (Pan et al., 2022), power down EES is set as energy-efficient strategy. As for modified MOEA/D (Wang et al., 2021), critical-path swap is used as the local search operator for the objective of makespan, and power down EES is adopted to reduce TEC. The parameters are strictly followed from the original paper: neighborhood size is set as 10 and crossover rate is set as 0.85.

All the experiments are coded in Python 3.9 and are executed on a laptop computer with Intel i5-11300H 3.10 GHz and 16 GB RAM. Each instance of an algorithm is run five times to obtain the average values (mean) and standard deviation (std) used for evaluation.

Table 6
Levels of parameters.

Factors	Factor levels		
	1	2	3
Psize	80	100	120
Pc	0.6	0.8	0.9
Pm	0.1	0.3	0.5
Q-iter	2	3	5
CT	0.5	0.8	1.2

Table 7
The orthogonal array of DOE.

No.	*Psize*	*Pc*	*Pm*	*Q-iter*	*CT*	SM
1	80	0.6	0.1	2	0.5	0.091
2	80	0.6	0.1	2	0.8	0.076
3	80	0.6	0.1	2	1.2	0.067
4	80	0.8	0.3	3	0.5	0.085
5	80	0.8	0.3	3	0.8	0.067
6	80	0.8	0.3	3	1.2	0.065
7	80	0.9	0.5	5	0.5	0.097
8	80	0.9	0.5	5	0.8	0.073
9	80	0.9	0.5	5	1.2	0.071
10	100	0.6	0.1	2	0.5	0.070
11	100	0.6	0.1	2	0.8	0.059
12	100	0.6	0.1	2	1.2	0.068
13	100	0.8	0.3	3	0.5	0.082
14	100	0.8	0.3	3	0.8	0.052
15	100	0.8	0.3	3	1.2	0.064
16	100	0.9	0.5	5	0.5	0.097
17	100	0.9	0.5	5	0.8	0.083
18	100	0.9	0.5	5	1.2	0.078
19	120	0.6	0.1	2	0.5	0.076
20	120	0.6	0.1	2	0.8	0.074
21	120	0.6	0.1	2	1.2	0.064
22	120	0.8	0.3	3	0.5	0.074
23	120	0.8	0.3	3	0.8	0.054
24	120	0.8	0.3	3	1.2	0.084
25	120	0.9	0.5	5	0.5	0.081
26	120	0.9	0.5	5	0.8	0.057
27	120	0.9	0.5	5	1.2	0.081

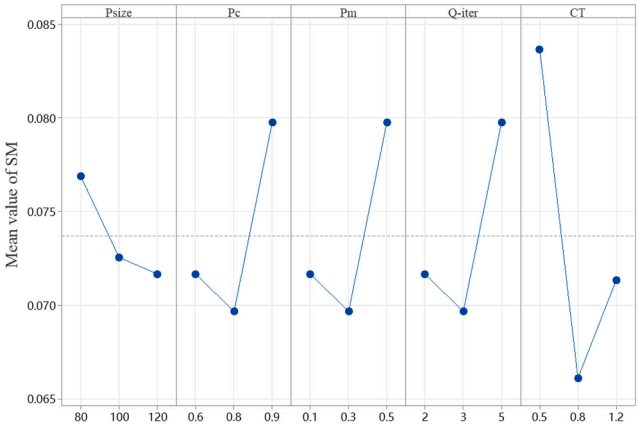


Fig. 17. Main effect plot of parameter tuning.

6.3. Algorithm comparison and analysis

In this section, QVNS-NSGA-II is compared to classical NSGA-II (Deb et al., 2002) and two state-of-the-art MOEAs, namely improved Jaya (Pan et al., 2022) and modified MOEA/D (Wang et al., 2021). The comparison results of NPS, SM and CM are listed in Tables 8–10. The **bold** values represent the best results among the three algorithms. Hit rate records the number of times the algorithm performed the best in all instances.

Table 8 reports the coverage metric between the four MOEAs, for simplicity, each algorithm is represented by an initial letter (i.e., $C(Q, J)$ for QVNS-NSGA-II and Jaya). As seen from the results, the proposed QVNS-NSGA-II has overwhelming superiority in all instances. This

Table 8
The comparison result based on CM.

	C(*Q*, *N*)		*C*(*N*, *Q*)		*C*(*Q*, *J*)		*C*(*J*, *Q*)		*C*(*Q*, *M*)		*C*(*M*, *Q*)	
	mean	std	Mean	std	mean	std	mean	std	mean	std	mean	std
10–3-3	**0.89**	**0.16**	0.00	0.00	**1.00**	**0.00**	0.00	0.00	**1.00**	**0.00**	0.00	0.00
10–3-4	**0.88**	**0.11**	0.00	0.00	**1.00**	**0.00**	0.00	0.00	**1.00**	**0.00**	0.00	0.00
10–3-6	**0.98**	**0.01**	0.00	0.00	**1.00**	**0.00**	0.00	0.00	**1.00**	**0.00**	0.00	0.00
10–5-3	**0.96**	**0.05**	0.00	0.00	**0.99**	**0.01**	0.00	0.00	**1.00**	**0.00**	0.00	0.00
10–5-4	**0.84**	**0.14**	0.00	0.00	**0.98**	**0.03**	0.00	0.00	**1.00**	**0.00**	0.00	0.00
10–5-6	**0.98**	**0.02**	0.00	0.00	**1.00**	**0.00**	0.00	0.00	**1.00**	**0.00**	0.00	0.00
20–3-3	**0.79**	**0.17**	0.00	0.00	**1.00**	**0.00**	0.00	0.00	**1.00**	**0.00**	0.00	0.00
20–3-4	**0.98**	**0.03**	0.00	0.00	**1.00**	**0.00**	0.00	0.00	**1.00**	**0.00**	0.00	0.00
20–3-6	**0.57**	**0.12**	0.00	0.00	**1.00**	**0.00**	0.00	0.00	**1.00**	**0.00**	0.00	0.00
20–5-3	**0.81**	**0.19**	0.00	0.00	**1.00**	**0.00**	0.00	0.00	**1.00**	**0.00**	0.00	0.00
20–5-4	**0.63**	**0.22**	0.00	0.00	**1.00**	**0.00**	0.00	0.00	**1.00**	**0.00**	0.00	0.00
20–5-6	**0.84**	**0.13**	0.00	0.00	**1.00**	**0.00**	0.00	0.00	**1.00**	**0.00**	0.00	0.00
50–3-3	**0.75**	**0.15**	0.00	0.00	**1.00**	**0.00**	0.00	0.00	**1.00**	**0.00**	0.00	0.00
50–3-4	**0.75**	**0.10**	0.00	0.00	**1.00**	**0.00**	0.00	0.00	**1.00**	**0.00**	0.00	0.00
50–3-6	**0.58**	**0.11**	0.00	0.00	**1.00**	**0.00**	0.00	0.00	**1.00**	**0.00**	0.00	0.00
50–5-3	**0.80**	**0.18**	0.00	0.00	**1.00**	**0.00**	0.00	0.00	**1.00**	**0.00**	0.00	0.00
50–5-4	**0.95**	**0.06**	0.00	0.00	**1.00**	**0.00**	0.00	0.00	**1.00**	**0.00**	0.00	0.00
50–5-6	**0.85**	**0.12**	0.00	0.00	**1.00**	**0.00**	0.00	0.00	**1.00**	**0.00**	0.00	0.00
Hit rate	**18/18**		0/18		**18/18**		0/18		**18/18**		0/18	

Table 9
The comparison result based on NPS.

Instance	NSGA-II		Jaya		MOEA/D		QVNS-NSGA-II	
	mean	std	mean	std	mean	std	mean	std
10-3-3	27	8	6	2	10	2	**38**	7
10-3-4	31	10	5	2	8	4	**41**	11
10-3-6	**89**	5	20	7	13	6	77	3
10-5-3	37	15	8	2	7	1	**52**	7
10-5-4	37	14	9	2	7	2	**47**	5
10-5-6	80	8	8	3	17	7	**91**	4
20-3-3	24	6	5	1	10	5	**41**	9
20-3-4	33	5	8	4	10	3	**38**	8
20-3-6	20	8	6	2	12	3	**51**	7
20-5-3	14	6	6	4	5	2	**33**	8
20-5-4	23	6	3	3	7	2	**40**	6
20-5-6	34	11	6	3	5	2	**37**	10
50-3-3	**25**	10	9	4	6	1	24	4
50-3-4	24	11	5	4	7	3	**28**	8
50-3-6	21	4	5	2	5	2	**29**	8
50-5-3	**40**	33	7	3	6	2	29	3
50-5-4	**35**	12	9	3	9	3	24	9
50-5-6	25	7	6	3	6	3	**29**	10
Hit rate	4/18		0/18		0/18		**14/18**	

proves QVNS-NSGA-II has great capability to find solutions with better convergence than the other three MOEAs. Such a situation can be explained by the fact that QVNS-NSGA-II leverages new NEH heuristic and Q-learning-driven GVNS to exploit more promising solution space with high efficiency.

Table 9 presents NPS comparison between the NSGA-II, improved Jaya, modified MOEA/D, and QVNS-NSGA-II. It can be seen that QVNS-NSGA-II outperforms the other three algorithms in finding diverse solutions in most cases. The proposed QVNS-NSGA-II can even obtain three to five times more Pareto solutions than Jaya and MOEA/D.

When QVNS-NSGA-II does not perform the best in NPS, QVNS-NSGA-II has a much lower standard deviation of NPS than NSGA-II. This indicates better robustness of our algorithm. For example, in instance 50–5-3, QVNS-NSGA-II has a much lower standard deviation (3) of NPS than NSGA-II (33).

Table 10 summarizes the SM comparison results of four MOEAs. QVNS-NSGA-II still outperforms the other three algorithms in the majority of instances (10/18).

To further illustrate the comparison results of the four algorithms, the boxplot of SM is given below. In Fig. 18, the SM values of QVNS-NSGA-II are averagely lower with a narrower interval range than the other three MOEAs.

6.4. Statistical test and visualization of solutions

To make the comparison results convincing statistically, paired-sample *t*-tests are conducted to eliminate stochastic error (Ding et al., 2016). The term "*t*-test (*A*, *B*)" in the first column suggests the conducted paired-sample *t*-test between algorithm *A* and *B*. *p*-value results from the hypothesis tests are presented in Table 11. The significance level is set as 95% ($\alpha = 0.05$). Note that the *t*-test (*A*, *B*) on CM compares the difference between *C* (*A*, *B*) and *C* (*B*, *A*).

The results show QVNS-NSGA-II significantly outperforms the other three algorithms in terms of CM, SM and NPS in the statistical sense. Specifically, QVNS-NSGA-II performs better in searching diverse and high-quality Pareto front solutions than NSGA-II, improved Jaya and modified MOEA/D.

Furthermore, the 3D scatter plots of the Pareto fronts by four MOEAs are also given in Fig. 19. The axes represent three objective functions, namely *TT*, *TEC* and *CTC*, respectively. The confidence level is set as 95% to obtain a 3D confidence ellipsoid (green), which describes the Pareto front in 3D space. We take 20–5-4 and 50–3-6 as examples of medium- and large-scale instances.

From Fig. 19, the Pareto solutions generated by QVNS-NSGA-II are significantly superior to others. The confidence ellipsoid is close to the coordinate origin, which visually reflects the convergence to the optimal front. Also, QVNS-NSGA-II provides much more diverse solutions that can help decision-makers to select based on preference.

The reason why QVNS-NSGA-II outperforms NSGA-II, Jaya and MOEA/D lies in the QVNS procedure. GVNS takes advantage of knowledge from Q-learning to achieve AOS in the DC phase and local search phase.

RL-driven AOS lightens the burden of blind search, enabling appropriate switches among DC neighborhood structures and local search operators without following a trajectory. This provides the algorithm with higher exploration and exploitation capabilities.

Considering the good balance of convergence and even distribution of the solutions, QVNS-NSGA-II is highly recommended to solve the multi-objective hybrid flow shop scheduling problem.

6.5. Sensitivity analysis

6.5.1. Trade-offs between three objectives

We further investigate the relationship between the three objectives. Fig. 20 and Fig. 21 show the trade-offs between the three objectives of

Table 10
The comparison of MOEAs based on SM.

Instance	NSGA-II		Jaya		MOEA/D		QVNS-NSGA-II	
	mean	std	mean	std	mean	std	mean	std
10-3-3	9.74E-02	3.13E-02	2.76E-01	2.82E-01	2.02E-01	7.49E-02	**7.77E-02**	2.37E-02
10-3-4	**1.22E-01**	1.11E-02	6.32E-01	4.97E-01	1.89E-01	1.50E-01	1.34E-01	6.78E-02
10-3-6	6.77E-02	1.19E-02	4.11E-01	2.80E-01	**5.81E-02**	6.86E-02	6.52E-02	1.33E-02
10-5-3	1.59E-01	5.95E-02	2.44E-01	2.02E-01	1.50E-01	8.85E-02	**6.43E-02**	1.72E-02
10-5-4	9.34E-02	3.24E-02	1.33E-01	1.91E-02	2.37E-01	3.84E-02	**8.52E-02**	1.99E-02
10-5-6	**4.07E-02**	1.56E-02	3.21E-01	2.09E-01	2.40E-01	9.64E-02	5.17E-02	7.29E-03
20-3-3	1.90E-01	2.06E-01	1.28E-01	7.51E-02	1.23E-01	4.59E-02	**1.23E-01**	6.05E-02
20-3-4	1.40E-01	4.48E-02	2.32E-01	1.40E-01	1.45E-01	7.14E-02	**7.90E-02**	1.42E-02
20-3-6	1.51E-01	1.00E-01	2.58E-01	1.53E-01	1.54E-01	1.08E-01	**1.12E-01**	2.41E-02
20-5-3	**8.46E-02**	1.72E-02	2.26E-01	1.74E-01	2.02E-01	2.14E-01	1.24E-01	3.06E-02
20-5-4	1.26E-01	6.35E-02	**6.42E-02**	9.47E-02	1.08E-01	3.20E-02	8.12E-02	3.44E-02
20-5-6	8.94E-02	2.85E-02	3.32E-01	5.47E-02	1.03E-01	8.04E-02	**8.41E-02**	3.44E-02
50-3-3	1.23E-01	9.41E-02	2.58E-01	1.47E-01	**7.18E-02**	5.38E-02	7.83E-02	1.64E-02
50-3-4	1.11E-01	3.88E-02	2.42E-01	1.31E-01	6.89E-02	1.55E-02	7.49E-02	2.79E-02
50-3-6	8.75E-02	4.68E-02	3.65E-01	7.45E-02	9.12E-02	1.24E-01	**8.35E-02**	2.28E-02
50-5-3	**8.74E-02**	4.47E-02	4.70E-01	3.36E-01	1.36E-01	9.41E-02	1.45E-01	4.86E-02
50-5-4	1.29E-01	4.97E-02	3.05E-01	2.53E-01	1.48E-01	7.64E-02	**1.05E-01**	4.05E-02
50-5-6	9.24E-02	2.64E-02	4.33E-01	4.28E-01	1.15E-01	5.65E-02	**7.90E-02**	2.96E-02
Hit rate	4/18		1/18		2/18		**10/18**	

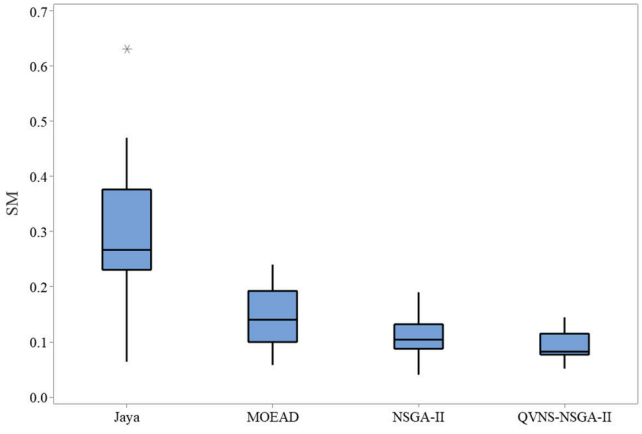


Fig. 18. The boxplot of SM results.

Table 11
Paired-sample *t*-tests for MOEAs on CM, SM and NPS ($\alpha = 0.05$, *p*-value).

	CM	SM	NPS
t-test (QVNS-NSGA-II, NSGA-II)	0.000	0.044	0.014
t-test (QVNS-NSGA-II, Jaya)	0.000	0.000	0.000
t-test (QVNS-NSGA-II, MOEA/D)	0.000	0.000	0.000

instance "10–3-3" in 3D and 2D respectively.

Fig. 20 demonstrates that *TT*, *TEC* and *CTC* in a Pareto solution always conflict with each other. Thus, it is not intuitive for a decision maker to choose an appropriate solution from a group of Pareto solutions considering the conflicts between the three objectives.

Fig. 21 shows 2D scatter plots concerning the combination of two objectives. It can be seen from Fig. 21(a) and Fig. 21(b) that *TEC* and *CTC* decrease greatly with the increase of *TT*. The reason is that a loose *TT* makes the use of slow-speed machines possible, which consumes less power than high-speed ones. Another reason is that the allocation of jobs onto the off-peak periods with low electricity prices leads to an increase in *TT*.

The decrease tendency can be divided into 2 stages: rapid decrease stage and mild decrease stage. In the rapid decrease stage, *TEC* and *CTC* go down sharply when *TT* increases. In the mild decrease stage, *TEC* and *CTC* decline moderately when *TT* grows. Thus, in the rapid decrease stage, *TEC* and *CTC* can be improved significantly at the expense of a slight deterioration of *TT*.

It is observed from Fig. 21(c) that *CTC* and *TEC* have a somewhat positive correlation because they are energy consumption-related. However, the increase of *TEC* is not necessarily followed by the increase of *CTC* due to the effect of TOU mechanism. The decision-maker is supposed to select a schedule with a lower *CTC* given the same *TEC*.

6.5.2. *Effect of different TOU tariffs*

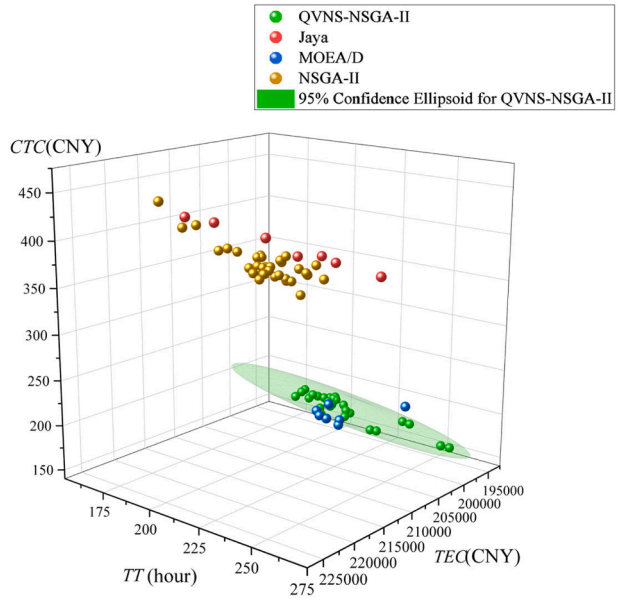
TOU mechanisms aim to encourage electricity users including manufacturers to adjust the temporal schedule of their electricity demands. However, TOU price varies with the season depending on seasonal demand differences and local electricity system capabilities. It is usually defined in advance for periods of a day, week, month or year (Shrouf et al., 2014). For example, China has issued documents to enhance effect of TOU tariffs including measures that adopting CPP in summer, changing TOU based on seasons (National Province Development and Reform Commision, 2021). Hence, it is important for managers to figure out how the TOU tariffs affect the scheduling results. A moderate-scaled instance "20–3-3" is employed to obtain Pareto optimal solutions under six different TOU tariffs in terms of *TT*, *TEC* and *CTC*.

Six TOU tariffs are formulated for sensitivity analysis in Fig. 22. TOU tariff (1) maintains the same price all the time as a control group. TOU tariff (4) refers to Development and Reform Commission of Jiangsu Province (2021) for spring and fall seasons. TOU tariffs (2) and (3) are the variants of (4) using a lower off-peak price (200CNY/MWh) and a lower on-peak price (750CNY/MWh), respectively. Both (5) and (6) employ CPP policy considering the electricity demands in summer and winter are much higher than those of spring and fall.

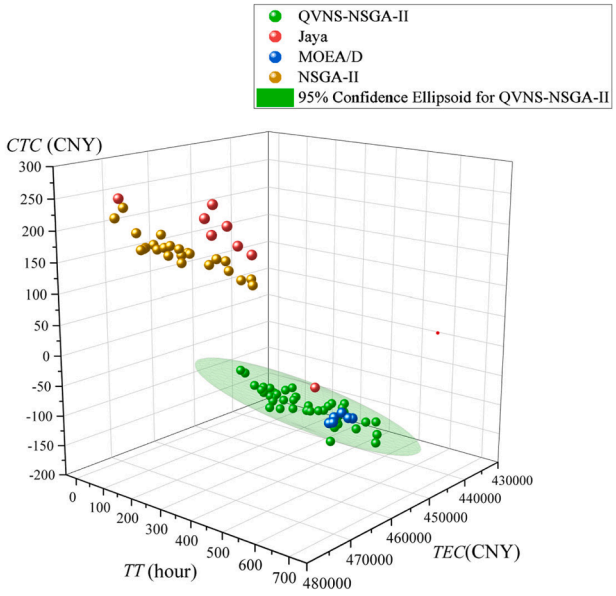
In Fig. 23, *TT* sees no apparent trend by adjusting the price and the period of TOU tariffs. There is no significant difference regarding the total tardiness *TT* by different TOU tariffs, indicating that adopting the TOU mechanisms can hardly affect product delivery.

It is observed from Fig. 24 that TOU tariffs have a significant impact on *TEC*. TOU tariffs (2) and (3) lead to much lower *TEC* than no TOU tariff (1) does. This means the use of TOU tariffs (2) and (3) can have a positive effect on electricity cost-saving for manufacturers. Statistically, the *TEC* under TOU tariff (4) is similar to the *TEC* under no TOU tariff (1). The *TEC* under TOU tariff (4) is much larger than the *TECs* under TOU tariffs (2) and (3) because the prices of electricity periods increase.

Similarly, there is a significant increase in *TEC* under TOU tariff (4) as the CPP policy is employed. In a broader sense, TOU tariffs vary with seasons. The fluctuation of *TEC* inspires manufacturers to allocate more



(a) The comparison among four algorithms of 20-5-4



(b) The comparison among four algorithms of 50-3-6

Fig. 19. 3D scatter plot of four MOEAs with confidence ellipsoids.

orders into seasons with TOU tariffs (2), (3) and (4) in order to save *TEC*.

In addition, *CTC* remains the same under different TOU tariffs. This means the use of TOU mechanisms has a limited impact on *CTC* savings for manufacturers (Fig. 25).

7. Conclusion

This paper investigates an energy-efficient hybrid flow shop scheduling problem with production- and environment-related objectives (total tardiness *TT*, total energy cost *TEC* and carbon trading cost *CTC*) simultaneously. A novel mixed-integer nonlinear programming model is presented, which considers uniform parallel machines and practical EES in both energy-supply and -demand sides, i.e., time-of-use tariffs and power down strategy, respectively. Then the properties of the problem

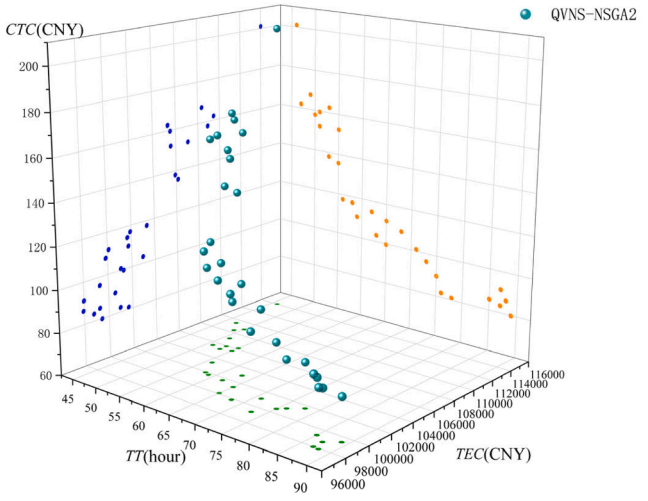


Fig. 20. Trade-offs between 3 objectives of instance "10-3-3".

are analyzed.

To solve the problem, we integrate RL, particularly Q-learning algorithm, into GVNS to achieve adaptive operator selection in the shaking and local search phase. Q-learning leverages the learned experience of Q-tables to select the most appropriate operator from a set of efficacious neighborhood structures and problem-specific local search operators. Then we first combine the Q-learning driven GVNS, a multi-objective NEH heuristic, and NSGA-II and propose a new algorithm named QVNS-NSGA-II. To the best of our knowledge, this is among the first research that combines metaheuristics with RL to solve EEHFSP.

We conduct a comprehensive set of experiments to evaluate the performance of our proposed algorithm. We compare the well-tuned QVNS-NSGA-II with a classic metaheuristic NSGA-II and two state-of-the-art metaheuristics, namely improved Jaya and modified MOEA/D. The experiment results show that the proposed algorithm can find more diverse Pareto solutions with high quality. This contributes to the Q-learning driven GVNS to prevent the search from being trapped into local optima. We can conclude that the proposed algorithm outperforms the three metaheuristics significantly for EEHFSP.

In addition, some management insights are gained from sensitivity analysis. As *TT* increases, *TEC* and C*TC* witness rapid decrease followed by mild decrease. Decision-makers can make a compromise between the three objectives. Besides, the analysis of different TOU tariffs demonstrates that TOU tariffs and CPP policy have a great impact on *TEC*, however, it can hardly affect *TT* and *CTC*. Manufacturers should allocate more orders into seasons with specific TOU tariffs that can save *TEC* when *TT* and *CTC* are stable.

Limitations and future directions are summarized. This paper designs the same action set and state set, both of which are neighborhood structures/ local search operators. However, this representation cannot take into account the quality of the solution or the iteration of the metaheuristic. The definition of action and state sets can be further investigated in the future. Besides, future research can investigate the integration of reinforcement learning in more phases of metaheuristics, such as initialization, parameter tuning, and fitness calculation. It is also interesting to employ more advanced and state-of-the-art RL techniques for EEHFSP.

CRediT authorship contribution statement

Peize Li: Methodology, Software, Writing – original draft. **Qiang Xue:** Data curation, Investigation. **Ziteng Zhang:** Visualization. **Jian Chen:** Conceptualization, Writing – review & editing. **Dequn Zhou:** Validation, Supervision.

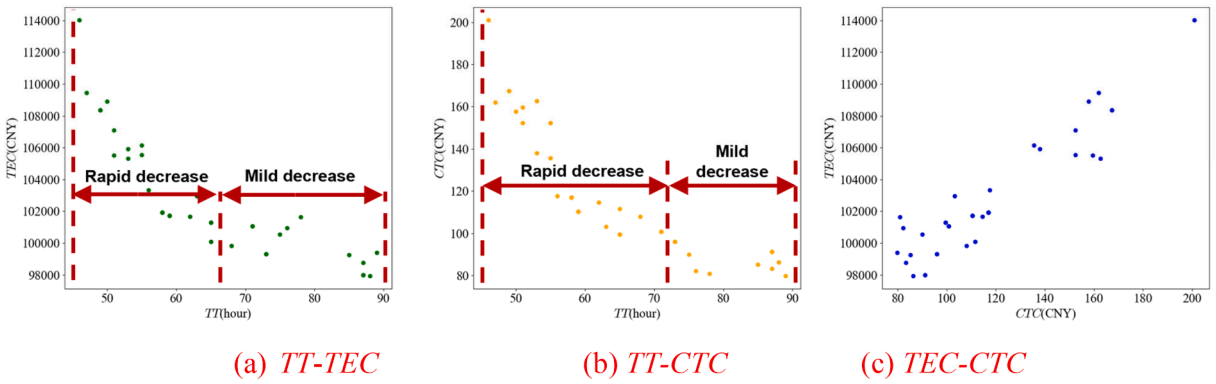


Fig. 21. Trade-offs between any 2 objectives of instance "10-3-3".

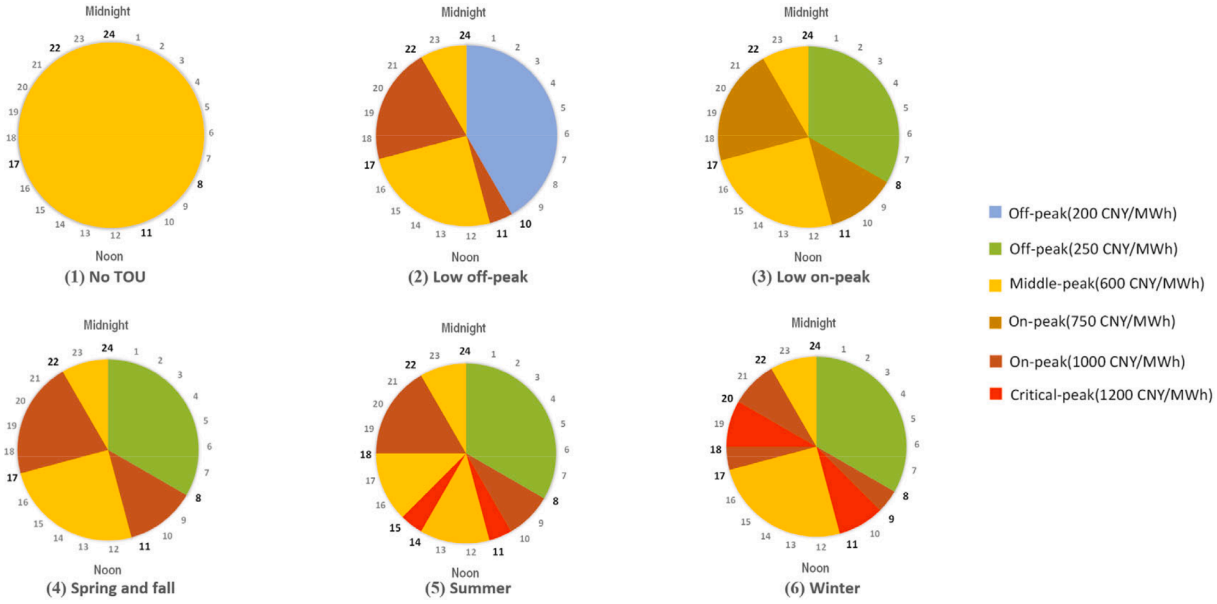


Fig. 22. Six TOU tariffs with different periods and prices.

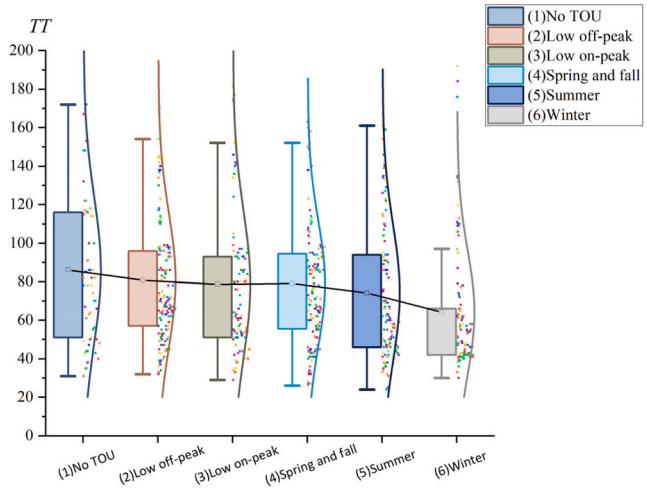


Fig. 23. Box plots of *TT* by different TOU tariffs.

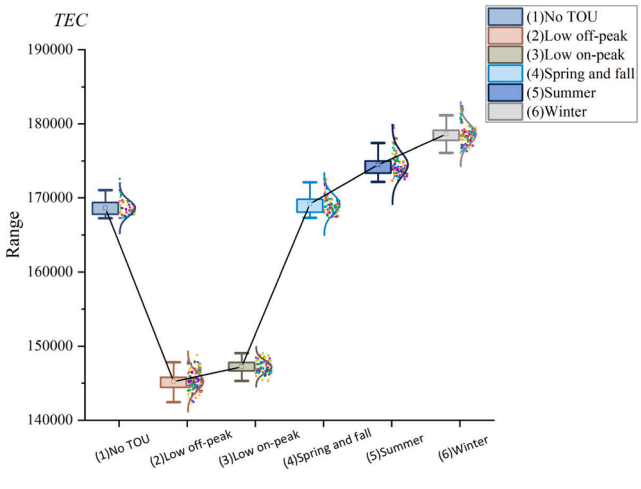


Fig. 24. Box plots of *TEC* by different TOU tariffs.

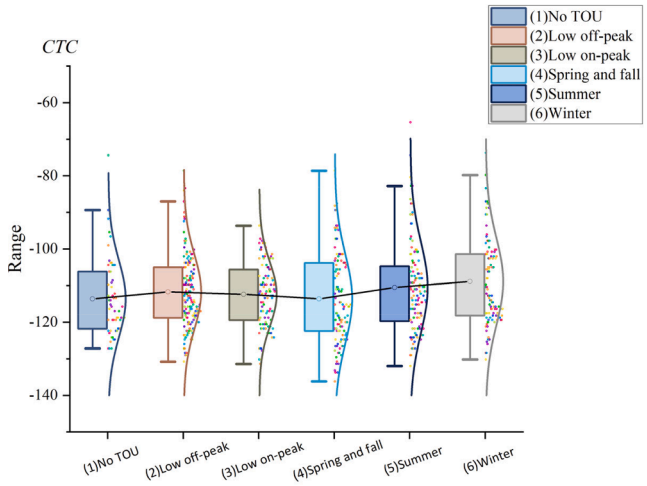


Fig. 25. Box plots of *CTC* by different TOU tariffs.

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.

Acknowledgements

This research was supported by the National Natural Science Foundation of China (52075259), the Qing Lan project, the China Postdoctoral Science Foundation (2021T140320, 2019M661839), and Sichuan Province Engineering Technology Research Center of Broadband Electronics Intelligent Manufacturing.

Appendix A. Fast non-dominated sorting and crowding distance calculation

Algorithm 4: Fast non-dominated sorting

Input: a population P
Output: Pareto fronts and ranks
For each $p \in P$ **do**
Define S_p as the set of solutions that the solution p dominates, n_p as the number of solutions which dominate the solution p
Set $S_p = \emptyset, n_p = 0$
For each $q \in P$ **do**
If $p \succ q$ **then**
Set $S_p = S_p \cup \{q\}$
Else if $p \prec q$ **then**
Set $n_p = n_p + 1$
End if
If $n_p = 0$ **then**
Set $p_{rank} = 1$ # p_{rank}. is the rank of p
Set $F_1 = F_1 \cup \{p\}$ #F_1 is the (first) Pareto front
End if
End for
Set $i = 1$
While $F_i \neq \emptyset$
Set $Q = \emptyset$
For each $p \in F_i$
For each $q \in S_p$
$n_q = n_q - 1$
If $n_q = 0$ **then**# q belongs to the next front
Set $q_{rank} = i + 1$
Set $Q = Q \cup \{q\}$
End if
End for
End for
Set $i = i + 1$
Set $F_i = Q$
End while

Algorithm 5: Crowding-distance calculation

Input: a Pareto-front F_i
Output: crowding-distance of a solution in Pareto-front F_i
Let l denote the number of solutions in the Pareto-front F_i
For each k **do**
Set $F_i[k]_{\text{distance}} = 0$
End for
For each objective f_j **do**

(*continued on next page*)

(*continued*)

Algorithm 5: Crowding-distance calculation

8: Sort individuals in ascending order using each objective value, $F_i = sort\left(F_i, f_j\right)$
9: Set $F_i[1]_{\text{distance}} = F_i[l]_{\text{distance}} = \infty$
10: **For** $i = 2$ to $(l-1)$ **do**
11: Set $F_i[i]_{\text{distance}} = F_i[i]_{\text{distance}} + \left(f_j(F_i[i+1]) - f_j(F_i[i-1])\right) \Big/ \left(f_j^{\max} - f_j^{\min}\right)$
12: **End for**
13: **End for**

References

Alejandro Rossit, D., Tohme, F., Frutos, M., 2018. The non-permutation flow-shop scheduling problem: a literature review. Omega 77, 143–153. https://doi.org/10.1016/j.omega.2017.05.010.

An, X., Si, G., Xia, T., Wang, D., Pan, E., Xi, L., 2023. An energy-efficient collaborative strategy of maintenance planning and production scheduling for serial-parallel systems under time-of-use tariffs. Appl. Energy 336, 120794. https://doi.org/10.1016/j.apenergy.2023.120794.

Asefi, H., Jolai, F., Rabiee, M., Araghi, M.E.T., 2014. A hybrid NSGA-II and VNS for solving a Bi-objective no-wait flexible flowshop scheduling problem. Int. J. Adv. Manuf. Technol. 75 (5–8), 1017–1033. https://doi.org/10.1007/s00170-014-6177-9.

Bengio, Y., Lodi, A., Prouvost, A., 2021. Machine learning for combinatorial optimization: a methodological tour D'horizon. Eur. J. Oper. Res. 290 (2), 405–421. https://doi.org/10.1016/j.ejor.2020.07.063.

Cai, J., Lei, D., Li, M., 2021. A shuffled frog-leaping algorithm with memeplex quality for bi-objective distributed scheduling in hybrid flow shop. Int. J. Prod. Res. 59 (18), 5404–5421. https://doi.org/10.1080/00207543.2020.1780333.

Cai, J.C., Lei, D.M., Wang, J., Wang, L., 2023. A novel shuffled frog-leaping algorithm with reinforcement learning for distributed assembly hybrid flow shop scheduling. Int. J. Prod. Res. 61 (4), 1233–1251. https://doi.org/10.1080/00207543.2022.2031331.

Chen, T.L., Cheng, C.Y., Chou, Y.H., 2020b. Multi-objective genetic algorithm for energy-efficient hybrid flow shop scheduling with lot streaming. Ann. Oper. Res. 290 (1–2), 813–836. https://doi.org/10.1007/s10479-018-2969-x.

Chen, J., Ning, T., Xu, G. and Liu, Y. (2022). A memetic algorithm for energy-efficient scheduling of integrated production and shipping. Int. J. Computer Integrated Manufact., 10.1080/0951192X.2022.2025618: 1-23. 10.1080/0951192X.2022.2025618.

Chen, J.F., Wang, L., Peng, Z.P., 2019. A Collaborative optimization algorithm for energy-efficient multi-objective distributed no-idle flow-shop scheduling. Swarm Evol. Comput. 50, 100557 https://doi.org/10.1016/j.swevo.2019.100557.

Chen, W., Wang, J., Yu, G., 2022b. Energy-efficient scheduling for an energy-intensive industry under punitive electricity price. J. Clean. Prod. 373, 133851 https://doi.org/10.1016/j.jclepro.2022.133851.

Chen, R.H., Yang, B., Li, S., Wang, S.L., 2020a. A self-learning genetic algorithm based on reinforcement learning for flexible job-shop scheduling problem. Comput. Ind. Eng. 149, 106778 https://doi.org/10.1016/j.cie.2020.106778.

Cheng, L., Tang, Q., Zhang, L., Zhang, Z., 2022. Multi-objective Q-learning-based hyper-heuristic with bi-criteria selection for energy-aware mixed shop scheduling. Swarm Evol. Comput. 69, 100985 https://doi.org/10.1016/j.swevo.2021.100985.

Cui, W., Lu, B., 2021. Energy-aware operations management for flow shops under tou electricity tariff. Comput. Ind. Eng. 151, 106942 https://doi.org/10.1016/j.cie.2020.106942.

Dai, M., Tang, D., Giret, A., Salido, M.A., Li, W.D., 2013. Energy-efficient scheduling for a flexible flow shop using an improved genetic-simulated annealing algorithm. Rob. Comput. Integr. Manuf. 29 (5), 418–429. https://doi.org/10.1016/j.rcim.2013.04.001.

Deb, K., Pratap, A., Agarwal, S., Meyarivan, T., 2002. A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE Trans. Evol. Comput. 6 (2), 182–197. https://doi.org/10.1109/4235.996017.

Development and Reform Commission of Jiangsu Province (2021). Notice of the Provincial Development and Reform Commission on Further Improvement of the Time-of-Use Tariff Mechanism. Nanjing, China, http://fzggw.jiangsu.gov.cn/art/2021/12/27/art_72382_10232339.html.

Ding, J., Schulz, S., Shen, L., Buscher, U., Lue, Z., 2021. Energy aware scheduling in flexible flow shops with hybrid particle swarm optimization. Comput. Oper. Res. 125, 105088 https://doi.org/10.1016/j.cor.2020.105088.

Ding, J.Y., Song, S.J., Wu, C., 2016. Carbon-efficient scheduling of flow shops by multi-objective optimization. Eur. J. Oper. Res. 248 (3), 758–771. https://doi.org/10.1016/j.ejor.2015.05.019.

Dong, J., Ye, C., 2022. Green scheduling of distributed two-stage reentrant hybrid flow shop considering distributed energy resources and energy storage system. Comput. Ind. Eng. 169, 108146 https://doi.org/10.1016/j.cie.2022.108146.

Durgut, R., Aydin, M.E., Atli, I., 2021. Adaptive Operator selection with reinforcement learning. Inf. Sci. 581, 773–790. https://doi.org/10.1016/j.ins.2021.10.025.

European Commission (2023). Emissions Cap and Allowances. https://climate.ec.europa.eu/eu-action/eu-emissions-trading-system-eu-ets/emissions-cap-and-allowances_en.

Gahm, C., Denz, F., Dirr, M., Tuma, A., 2016. Energy-efficient scheduling in manufacturing companies: a review and research framework. Eur. J. Oper. Res. 248 (3), 744–757. https://doi.org/10.1016/j.ejor.2015.07.017.

Gao, K.Z., Huang, Y., Sadollah, A., Wang, L., 2020. A review of energy-efficient scheduling in intelligent production systems. Complex & Intelligent Syst. 6 (2), 237–249. https://doi.org/10.1007/s40747-019-00122-6.

Garey, M.R., Johnson, D.S., Sethi, R., 1976. The complexity of flowshop and jobshop scheduling. Math. Oper. Res. 1 (2), 117–129. https://doi.org/10.1287/moor.1.2.117.

Michel Gendreau and Potvin, J.-Y. (2019). Handbook of Metaheuristics (3rd Edition). Switzerland, Springer Cham. 10.1007/978-3-319-91086-4.

Ghorbani Saber, R., Ranjbar, M., 2022. Minimizing the total tardiness and the total carbon emissions in the permutation flow shop scheduling problem. Comput. Oper. Res. 138, 105604 https://doi.org/10.1016/j.cor.2021.105604.

Goli, A., Ala, A., Hajiaghaei-Keshteli, M., 2023. Efficient multi-objective meta-heuristic algorithms for energy-aware non-permutation flow-shop scheduling problem. Expert Syst. Appl. 213, 119077 https://doi.org/10.1016/j.eswa.2022.119077.

Gupta, J.N.D., 1988. Two-stage, hybrid flowshop scheduling problem. J. Oper. Res. Soc. 39 (4), 359–364. https://doi.org/10.1057/jors.1988.63.

Hansen, P., Mladenovic, N., 2001. Variable neighborhood search: principles and applications. Eur. J. Oper. Res. 130 (3), 449–467. https://doi.org/10.1016/s0377-2217(00)00100-4.

Hansen, P., Mladenovic, N., Moreno Perez, J.A., 2010. Variable neighbourhood search: methods and applications. Ann. Oper. Res. 175 (1), 367–407. https://doi.org/10.1007/s10479-009-0657-6.

Ho, M.H., Hnaien, F., Dugardin, F., 2022. Exact method to optimize the total electricity cost in two-machine permutation flow shop scheduling problem under time-of-use tariff. Comput. Oper. Res. 144, 105788 https://doi.org/10.1016/j.cor.2022.105788.

IEA, 2021. World Energy Balances. International Energy Agency, Paris, France.

Jiang, S.-L., Xu, C., Zhang, L., Ma, Y., 2023. A decomposition-based two-stage online scheduling approach and its integrated system in the hybrid flow shop of steel industry. Expert Syst. Appl. 213, 119200 https://doi.org/10.1016/j.eswa.2022.119200.

Karimi-Mamaghan, M., Mohammadi, M., Jula, P., Pirayesh, A., Ahmadi, H., 2020. A learning-based metaheuristic for a multi-objective agile inspection planning model under uncertainty. Eur. J. Oper. Res. 285 (2), 513–537. https://doi.org/10.1016/j.ejor.2020.01.061.

Karimi-Mamaghan, M., Mohammadi, M., Meyer, P., Karimi-Mamaghan, A.M., Talbi, E.-G., 2022. Machine learning at the service of meta-heuristics for solving combinatorial optimization problems: a state-of-the-art. Eur. J. Oper. Res. 296 (2), 393–422. https://doi.org/10.1016/j.ejor.2021.04.032.

Karimi-Mamaghan, M., Mohammadi, M., Pasdeloup, B., Meyer, P., 2023. Learning to select operators in meta-heuristics: an integration of Q-learning into the iterated greedy algorithm for the permutation flowshop scheduling problem. Eur. J. Oper. Res. 304 (3), 1296–1330. https://doi.org/10.1016/j.ejor.2022.03.054.

Khalaf, A.F., Wang, Y., 2018. Energy-cost-aware flow shop scheduling considering intermittent renewables, energy storage, and real-time electricity pricing. Int. J. Energy Res. 42 (12), 3928–3942. https://doi.org/10.1002/er.4130.

Lee, T., Loong, Y.-T., 2019. A review of scheduling problem and resolution methods in flexible flow shop. Int. J. Ind. Eng. Comput. 10, 67–88. https://doi.org/10.5267/j.ijiec.2018.4.001.

Lei, D., Zheng, Y., 2017. Hybrid flow shop scheduling with assembly operations and key objectives: a novel neighborhood search. Appl. Soft Comput. 61, 122–128. https://doi.org/10.1016/j.asoc.2017.07.058.

Li, K., Fialho, A., Kwong, S., Zhang, Q., 2013. Adaptive operator selection with bandits for a multiobjective evolutionary algorithm based on decomposition. IEEE Trans. Evol. Comput. 18 (1), 114–130. https://doi.org/10.1109/TEVC.2013.2239648.

Li, H., Gao, K., Duan, P.-Y., Li, J.-Q., Zhang, L., 2022. An improved artificial bee colony algorithm with q-learning for solving permutation flow-shop scheduling problems. IEEE Trans. Syst., Man, and Cybernetics: Syst. https://doi.org/10.1109/tsmc.2022.3219380.

Li, M., Wang, G.-G., 2022. A review of green shop scheduling problem. Inf. Sci. 589, 478–496. https://doi.org/10.1016/j.ins.2021.12.122.

Lu, C., Liu, Q., Zhang, B., Yin, L., 2022. A pareto-based hybrid iterated greedy algorithm for energy-efficient scheduling of distributed hybrid flowshop. Expert Syst. Appl. 204, 117555 https://doi.org/10.1016/j.eswa.2022.117555.

Luo, H., Du, B., Huang, G.Q., Chen, H., Li, X., 2013. Hybrid flow shop scheduling considering machine electricity consumption cost. Int. J. Prod. Econ. 146 (2), 423–439. https://doi.org/10.1016/j.ijpe.2013.01.028.

Mansouri, S.A., Aktas, E., Besikci, U., 2016. Green scheduling of a two-machine flowshop: trade-off between makespan and energy consumption. Eur. J. Oper. Res. 248 (3), 772–788. https://doi.org/10.1016/j.ejor.2015.08.064.

Mladenovic, N., Hansen, P., 1997. Variable neighborhood search. Comput. Oper. Res. 24 (11), 1097–1100. https://doi.org/10.1016/s0305-0548(97)00031-2.

Mouzon, G., Yildirim, M.B., 2008. A framework to minimise total energy consumption and total tardiness on a single machine. Int. J. Sustain. Eng. 1 (2), 105–116. https://doi.org/10.1080/19397030802257236.

Mouzon, G., Yildirim, M.B., Twomey, J., 2007. Operational methods for minimization of energy consumption of manufacturing equipment. Int. J. Prod. Res. 45 (18–19), 4247–4271. https://doi.org/10.1080/00207540701450013.

Naderi, B., Ruiz, R., Zandieh, M., 2010. Algorithms for a realistic variant of flowshop scheduling. Comput. Oper. Res. 37 (2), 236–246. https://doi.org/10.1016/j.cor.2009.04.017.

National Province Development and Reform Commision (2021). Notice of the National Development and Reform Commission on Further Improvement of the Tou Tariff Mechanism. Beijing, China, http://www.gov.cn/zhengce/zhengceku/2021-07/29/content_5628297.htm.

Nawaz, M., Enscore, E.E., Ham, I., 1983. A heuristic algorithm for the M-machine, N-job flow-shop sequencing problem. Omega 11 (1), 91–95. https://doi.org/10.1016/0305-0483(83)90088-9.

Öztop, H., Tasgetiren, M. F., Kandiller, L. and Pan, Q. K. (2020). A Novel General Variable Neighborhood Search through Q-Learning for No-Idle Flowshop Scheduling. 2020 IEEE Congress on Evolutionary Computation (CEC), 1-8. 10.1109/CEC48606.2020.9185556.

Pan, Y., Gao, K., Li, Z., Wu, N., 2022. Solving biobjective distributed flow-shop scheduling problems with lot-streaming using an improved jaya algorithm. IEEE Trans. Cybern. https://doi.org/10.1109/tcyb.2022.3164165.

Pan, Q.-K., Wang, L., Li, J.-Q., Duan, J.-H., 2014. A novel discrete artificial bee colony algorithm for the hybrid flowshop scheduling problem with makespan minimisation. Omega 45, 42–56. https://doi.org/10.1016/j.omega.2013.12.004.

Pinedo, M. L. (2016). Scheduling: Theory, Algorithms, and Systems(5th Edition), Springer Cham.

Ribas, I., Leisten, R., Framinan, J.M., 2010. Review and classification of hybrid flow shop scheduling problems from a production system and a solutions procedure perspective. Comput. Oper. Res. 37 (8), 1439–1454. https://doi.org/10.1016/j.cor.2009.11.001.

Richard, S.S., Andrew, G.B., 2019. Reinforcement Learning: An Introdution, 2nd ed. The MIT Press.

Ruiz, R., Maroto, C., 2006. A genetic algorithm for hybrid flowshops with sequence dependent setup times and machine eligibility. Eur. J. Oper. Res. 169 (3), 781–800. https://doi.org/10.1016/j.ejor.2004.06.038.

Ruiz, R., Stützle, T., 2007. A simple and effective iterated greedy algorithm for the permutation flowshop scheduling problem. Eur. J. Oper. Res. 177 (3), 2033–2049. https://doi.org/10.1016/j.ejor.2005.12.009.

Shao, W., Shao, Z., Pi, D., 2022. Multi-local search-based general variable neighborhood search for distributed flow shop scheduling in heterogeneous multi-factories. Appl. Soft Comput. 125, 109138 https://doi.org/10.1016/j.asoc.2022.109138.

Shrouf, F., Ordieres-Meré, J., García-Sánchez, A., Ortega-Mier, M., 2014. Optimizing the production scheduling of a single machine to minimize total energy consumption costs. J. Clean. Prod. 67, 197–207. https://doi.org/10.1016/j.jclepro.2013.12.024.

Syswerda, G., 1991. Schedule Optimization Using Genetic Algorithms. Van Nostrand Reinhold, New York.

Taillard, E., 1990. Some efficient heuristic methods for the flow shop sequencing problem. Eur. J. Oper. Res. 47 (1), 65–74. https://doi.org/10.1016/0377-2217(90)90090-X.

Talbi, E.-G., 2021. Machine learning into Metaheuristics: a survey and taxonomy. ACM Comput. Surv. 54 (6), 1–32. https://doi.org/10.1145/3459664.

The State Council of the People's Republic of China (2021). China's 14th Five-Year Plan (2021-2025). Beijing, China, http://www.gov.cn/xinwen/2021-03/13/content_5592681.htm.

Wang, Z., Cai, B., Li, J., Yang, D., Zhao, Y., Xie, H., 2023b. Solving non-permutation flow-shop scheduling problem via a novel deep reinforcement learning approach. Comput. Oper. Res. 151, 106095 https://doi.org/10.1016/j.cor.2022.106095.

Wang, H., Fu, Y., Huang, M., Huang, G.Q., Wang, J., 2017. A NSGA-II based memetic algorithm for multiobjective parallel flowshop scheduling problem. Comput. Ind. Eng. 113, 185–194. https://doi.org/10.1016/j.cie.2017.09.009.

Wang, G., Li, X., Gao, L., Li, P., 2021. Energy-efficient distributed heterogeneous welding flow shop scheduling problem using a modified moea/D. Swarm Evol. Comput. 62, 100858 https://doi.org/10.1016/j.swevo.2021.100858.

Wang, Z.Y., Shen, L.S., Li, X.Y., Gao, L., 2023c. An improved multi-objective firefly algorithm for energy-efficient hybrid flowshop rescheduling problem. J. Clean. Prod. 385, 135738 https://doi.org/10.1016/j.jclepro.2022.135738.

Wang, S.-Y., Wang, L., 2016. An estimation of distribution algorithm-based memetic algorithm for the distributed assembly permutation flow-shop scheduling problem. IEEE Trans. Syst., Man, Cybernetics: Syst. 46 (1), 139–149. https://doi.org/10.1109/tsmc.2015.2416127.

Wang, S., Wang, X., Chu, F., Yu, J., 2020. An energy-efficient two-stage hybrid flow shop scheduling problem in a glass production. Int. J. Prod. Res. 58 (8), 2283–2314. https://doi.org/10.1080/00207543.2019.1624857.

Wang, Y.-J., Wang, G.-G., Tian, F.-M., Gong, D.-W., Pedrycz, W., 2023a. Solving energy-efficient fuzzy hybrid flow-shop scheduling problem at a variable machine speed using an extended NSGA-II. Eng. Appl. Artif. Intel. 121, 105977 https://doi.org/10.1016/j.engappai.2023.105977.

Watkins, C., 1989. Learning from Delayed Rewards. University of Cambridge, PhD.

Wu, X.Q., Che, A.D., 2020. Energy-efficient no-wait permutation flow shop scheduling by adaptive multi-objective variable neighborhood search. Omega 94, 102117. https://doi.org/10.1016/j.omega.2019.102117.

Yenisey, M.M., Yagmahan, B., 2014. Multi-objective permutation flow shop scheduling problem: literature review, classification and current trends. Omega 45, 119–135. https://doi.org/10.1016/j.omega.2013.07.004.

Yu, C., Semeraro, Q., Matta, A., 2018. A genetic algorithm for the hybrid flow shop scheduling with unrelated machines and machine eligibility. Comput. Oper. Res. 100, 211–229. https://doi.org/10.1016/j.cor.2018.07.025.

Yue, L., Wang, H., Mumtaz, J., Rauf, M., Li, Z., 2023. Energy-efficient scheduling of a two-stage flexible printed circuit board flow shop using a hybrid pareto spider monkey optimisation algorithm. J. Ind. Inf. Integr. 31, 100412 https://doi.org/10.1016/j.jii.2022.100412.

Zhao, F., He, X., Wang, L., 2021a. A two-stage cooperative evolutionary algorithm with problem-specific knowledge for energy-efficient scheduling of no-wait flow-shop problem. IEEE Trans. Cybern. 51 (11), 5291–5303. https://doi.org/10.1109/TCYB.2020.3025662.

Zhao, F., Zhang, L., Cao, J., Tang, J., 2021b. A cooperative water wave optimization algorithm with reinforcement learning for the distributed assembly no-idle flowshop scheduling problem. Comput. Ind. Eng. 153, 107082 https://doi.org/10.1016/j.cie.2020.107082.

Zhao, F., Di, S., Wang, L., 2022a. A hyperheuristic with Q-learning for the multiobjective energy-efficient distributed blocking flow shop scheduling problem. IEEE Trans. Cybern. https://doi.org/10.1109/tcyb.2022.3192112.

Zhao, F., Hu, X., Wang, L., Xu, T., Zhu, N., Zhu, N., 2022b. A reinforcement learning-driven brain storm optimisation algorithm for multi-objective energy-efficient distributed assembly no-wait flow shop scheduling problem. Int. J. Prod. Res. https://doi.org/10.1080/00207543.2022.2070786.

Zhao, F.Q., Liu, Y., Zhang, Y., Ma, W.M., Zhang, C., 2017. A hybrid harmony search algorithm with efficient job sequence scheme and variable neighborhood search for the permutation flow shop scheduling problems. Eng. Appl. Artif. Intel. 65, 178–199. https://doi.org/10.1016/j.engappai.2017.07.023.