Highlights

Knowledge-driven two-stage memetic algorithm for energy-efficient flexible job shop scheduling with machine breakdowns

Cong Luo, Wenyin Gong, Chao Lu

- A two-stage evolution framework is proposed for EMBFJSP.
- A rescheduling strategy is applied for machine breakdowns.
- The population is initialized by three problem-specific heuristics.
- Four knowledge-driven variable neighborhood search operators are proposed.
- Two types of energy-saving strategies are designed.

Knowledge-driven two-stage memetic algorithm for energy-efficient flexible job shop scheduling with machine breakdowns

Cong Luo[a], Wenyin Gong[a,*], Chao Lu[a,*]

[a] *School of Computer Science, China University of Geosciences, Wuhan 430074, China.*

Abstract

This paper focuses on the multi-objective energy-efficient flexible job shop scheduling problem with machine breakdowns. To mitigate the impact of machine breakdowns, a rescheduling strategy is implemented in the scheduling process. In addition to sequencing the operations, the current problem is to determine the appropriate allocation of the machine and the proper speed of the machine to minimize both makespan and total energy consumption simultaneously. A mixed integer linear programming model is established to describe the considered problem. With the aim of effectively solving this problem, a knowledge-driven two-stage memetic algorithm (KTMA) is proposed. In the first stage, a hybrid initialization strategy that combines three problem-specific heuristics is applied to generate a high-quality initial population. Then, a knowledge-driven variable neighborhood search approach is designed for quickly converging and fully exploiting the solution space of the KTMA. In the second stage, two energy-saving strategies are designed to further reduce the total energy consumption. Extensive experiments carried out to compare the KTMA with some well-known algorithms confirm that the proposed KTMA can efficiently solve this problem.

Keywords: Energy-efficient flexible job shop scheduling, machine breakdowns, multi-objective optimization, memetic algorithm, knowledge.

1. Introduction

Over the past few decades, economic globalization has provided a strong impetus for world economic growth, resulting in many uncertain challenges for traditional manufacturing. Nowadays, some concepts place higher demands on traditional manufacturing, for example, Industry 4.0, Carbon Neutrality, and Intelligent Manufacturing (Serrano-Ruiz et al., 2022; Lu et al., 2021b). The manufacturing industry is the foundation and backbone of the country's production capacity and national economy, and its level of development is often a reflection of a country's comprehensive national power. As an important part of the

*Corresponding author

Email addresses: 1202121530@cug.edu.cn (Cong Luo), wygong@cug.edu.cn (Wenyin Gong), luchao@cug.edu.cn (Chao Lu)

national industry and the leading sector of economic growth, the manufacturing industry of China has maintained a sustainable and stable development trend (Liang et al., 2022). However, as the pressure of international competition climbs and the cost of labor increases, the traditional manufacturing industry is in urgent need of transformation and upgrading to seek a new path of development.(Li et al., 2022a; Zhang et al., 2021).

On the other hand, with the frequent occurrence of environmental pollution, extreme weather, energy shortage, and other problems, more and more people are raising awareness of energy-saving and emission reduction (Lu et al., 2021a; Wei et al., 2022). In the past, the scheduling problem usually focuses on only the factors, including makespan, machine workload, and production costs, but often overlooked the importance of green energy-saving measures. Thus, green scheduling (Gong et al., 2021; Li et al., 2020; Zheng & Wang, 2018) is proposed, which focuses on energy-saving and consumption reduction and has become a hot issue for scholars around the world. The efficiency of the scheduling scheme will have a direct and significant impact on energy consumption and environmental emissions. For sustainable development, energy-saving strategies for job shop scheduling problems must receive sufficient attention. However, in a real-world scheduling environment, various unexpected situations will occur, for example, machine breakdowns (Wu et al., 2018), job arrivals (Gao et al., 2019b; Caldeira et al., 2020), delivery date changes (Hidri et al., 2019), and other uncertain factors (Afsar et al., 2022; Luo et al., 2022), etc. Therefore, it is very necessary to investigate dynamic scheduling schemes to handle the actual situation.

As the most common type of job shop scheduling problem (JSP) (Bhatt & Chauhan, 2015), the flexible job shop scheduling problem (FJSP), defined as the NP-hard problem, has been widely studied by many scholars (Gao et al., 2019a). Nowadays, in the existing literature on FJSP, most of the literature studies the static scheduling problems, but rarely studies the dynamic scheduling problems. However, it is necessary to take full account of the actual manufacturing environment and the customer requirements to propose a reliable schedule scheme to deal with various dynamic events. Therefore, research on the dynamic flexible job shop scheduling problem (DFJSP) has aroused the interest of many scholars (Lei et al., 2022; Ferreira et al., 2022; Mohan et al., 2019). Liang et al. (2020) investigated the dynamic scheduling problem for the arrival of new jobs and established an improved probabilistic neural network model. They applied a genetic algorithm to optimize the smoothing factor for the purpose of improving model performance. Zaharie et al. (2017) investigated the uncertain events for delivery date changes and proposed an integer programming model for accepting, delaying, or rejecting the ordered products which aims at obtaining the best long-term and short-term results. Wang et al. (2022) considered six dynamic events to simulate a realistic production environment. They applied deep reinforcement learning techniques to a real-time processing framework with the combined scheduling rules to further optimize the obtained solutions. In the actual scheduling environment, machine breakdowns frequently occur as a type of dynamic event. Therefore, we mainly investigated the flexible job shop scheduling with machine breakdowns in this paper. When machine breakdowns occur, reasonable rescheduling of the scheduling sequence is critical and required. However, most studies on the dynamic event of machine breakdowns are only dedicated to finding a new robust scheduling scheme, thus omitting

the need for rescheduling. Duan & Wang (2022) studied machine breakdown problems considering the factors of system reusability and task recurrence and proposed a dynamic event response strategy instead of rescheduling. They developed a multi-objective particle swarm algorithm combined with the dynamic event response strategy to establish an optimization model which is evaluated by the indicators of reusability and reproducibility. Abedi et al. (2020) focused on the energy-efficient job-shop scheduling problem with deteriorating machines. In order to reduce the impact of the deteriorating machines, they proposed a multi-population, multi-objective memetic algorithm that aims to determine the proper machine speeds, thus reducing the wear and tear of the machine. Zhang et al. (2022) applied the convolutional neural network model to the two-stage framework to address flexible job shop scheduling problem considering machine breakdown. They trained a prediction model using CNN in the first stage. Then, in the second stage, they used the model trained in the first stage to predict the results and thus evaluate the robustness.

In the background of energy-saving and consumption reduction, more and more scholars are focusing on green scheduling and designing effective energy-saving strategies as a way to promote the synergistic development of intelligent manufacturing and green production. Regarding the energy-efficient flexible job shop scheduling problem, Gong et al. (2022) proposed a two-stage memetic algorithm to reduce the times of machine restarts and designed a strategy based on operation-block moving to further optimize the objective of total energy consumption. Duan & Wang (2021) proposed a non-dominated genetic ranking algorithm and developed two types of energy-saving strategies considering idle time and speed level for the flexible job shop problem with machine breakdowns. Pan et al. (2022) considered a fuzzy flexible job shop scheduling problem, they constructed a bi-population evolutionary algorithm with a feedback mechanism to reduce the consumption of energy. Zhao et al. (2022) established a distributed no-idle flow-shop scheduling model related to an energy-saving strategy and proposed a self-learning discrete Jaya algorithm with a green neighborhood search strategy.

Memory algorithms (MAs) have been successfully utilized in various variants of the scheduling problem due to their excellent global and local search capabilities (Wang & Wang, 2021; Lou et al., 2022). Many scholars have proposed to incorporate problem knowledge into MAs to solve various types of multi-objective energy-efficient scheduling problems (Li et al., 2022b; Lu et al., 2022). Based on the above description, this study proposed a knowledge-driven two-stage memetic algorithm for solving the energy-efficient flexible job shop scheduling problem with machine breakdowns (EMBFJSP) to optimize makespan and **TEC** simultaneously, which includes the following five innovation points: (i) a two-stage evolution framework is utilized to address EMBFJSP; (ii) a rescheduling strategy is applied to reschedule the scheduling sequence when machine breakdowns occur. (iii) a hybrid initialization strategy consisting of three heuristics is constructed for generating a high-quality population. (iv) a knowledge-driven variable neighborhood search approach that includes four problem-specific neighborhood structures is presented to fully exploit the solution space. (v) two types of energy-saving strategies are designed to further reduce **TEC** effectively without increasing makespan. Meanwhile, to better solve EMBFJSP, we established a mixed-integer linear programming (MILP) model for the EMBFJSP.

The remainder of this paper is organized as follows. The problem description and modeling are described in Section 2. Moreover, details of our proposed KTMA are explained in Section 3. After that, experimental results and discussion are presented in Section 4. Finally, conclusions and future works are introduced in Section 5.

2. Problem description and modeling

2.1. Problem description

The EMBFJSP is a complex combinatorial optimization problem that requires allocating limited resources to several jobs under certain constraints in order to optimize both the makespan and the **TEC** objectives simultaneously. The makespan and the **TEC** are two conflicting objectives, and the decrease of the makespan is often accompanied by the increase of the **TEC**. The trade-off between makespan and **TEC** can be effectively reflected by the Pareto front, and Figure 1 shows a Pareto front for this EMBFJSP. Meanwhile, it is not only necessary to assign machines to operations, but also to select the appropriate machine processing speed and obtain the optimal processing sequence of jobs, and the start time and completion time of each job should be determined. Therefore, the EMBFJSP is also a complicated scheduling problem. The EMBFJSP should solve the three sub-problems: machine allocation, machine speed selection, and job processing sequence optimization. For this problem, there is a set of n jobs $J = \{J_1, J_2, ..., J_n\}$ and a set of m machines $M = \{M_1, M_2, ..., M_m\}$. Each job J_i has a sequence of n_i operations $O_i = \{O_{i,1}, O_{i,2}, ..., O_{i,n_i}\}$ to be processed one after another according to the precedence constraints. More precisely, the EMBFJSP is addressed by the following three steps: (i) allocate a machine from the available machine set for each operation; (ii) select a suitable processing speed level for each machine, it is worth noting that different processing speeds of the machines have different processing time; (iii) arrange the processing sequence of all operations on each machine.

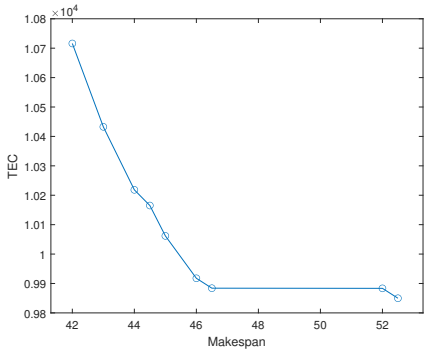


Figure 1: A Pareto front of the EMBFJSP.

Machine breakdown as the most common type of dynamic event, often appears in the actual manufacturing environment (Soofi et al., 2021). When machine breakdowns occur, the faulty machine should be repaired and can not process any jobs until the repair process is complete. Therefore, machine breakdown must satisfy three critical factors: the breakdown machine, the condition

that the machine breakdown occurs, and the time to repair the breakdown machine. There is a direct relationship between the workload of a machine and the probability of machine breakdown; that is, a machine that has been used for a long time tends to be more likely to break down than others. When machine breakdowns occur, the breakdown machine will no longer process any operations until the faulty machine is repaired. All unstarted operations will be rescheduled according to the rescheduling strategy.

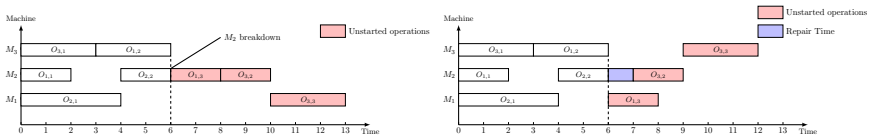
In short, the assumptions of EMBFJSP are described below:

- All jobs and all machines are ready at time zero.
- Any operations can be processed by machines that are available at any speed level.
- Each operation can only be processed by one available machine with the selected speed level at any time.
- Each machine can only process one operation at a time.
- An operation cannot be processed until its preceding operations are completed.
- Once started, no interruptions are allowed until the operation processing is complete unless the machine breakdown occurs.
- When the machine breakdown occurs, the faulty machine cannot process the operations and shut down instantly.
- In the case of a machine breakdown, the machine is repaired within a determined period of time.

2.2. Rescheduling strategy

Based on the previous description, how rescheduling the unstarted operations has a significant effect on the scheduling sequence. Machine breakdown is directly related to the processing time of the machines, therefore, when machine breakdowns occur, the most important thing is how to reschedule the affected jobs to make the least occurrence of machine breakdowns. Thus, the core idea of the rescheduling strategy is to evenly distribute the affected jobs so that each machine has a similar processing time, which will reduce the probability of machine breakdowns and also reduce the makespan. In this study, we design an effective rescheduling strategy to restore the previous scheduling sequence when machine breakdowns occur. The rescheduling strategy aims to reschedule the operations that have not yet started; that is, all unstarted operations in the original scheduling sequence will be rescheduled. Among them, the unstarted operations are allocated to the candidate machines according to the minimum machine workload rule, which aims to balance the workload to reduce makespan and effectively reduce the occurrence of machine breakdown. It is worth noting that when the job is being processed and its processing machine breakdown occurs, the job needs to wait until the processing machine is repaired before continuing to complete subsequent processing. During this period, the job will not be assigned to another machine for processing.

To explain it better, an example is shown in Figure 2. As shown in the original scheduling sequence in (a) of Figure 2, M_2 breakdown at time six and $O_{1,3}$, $O_{3,2}$ and $O_{3,3}$ are the unstarted operations. The previous scheduling sequence has remained, $O_{1,3}$, $O_{3,2}$ and $O_{3,3}$ select available candidate machine sets in turn according to the minimum workload rule. Before rescheduling, the workload of M_1, M_2, and M_3 are 4, 5, and 6, respectively. Therefore, $O_{1,3}$ is assigned to M_1, and the workload of M_1 is updated to 6. Then, based on the minimum workload rule, $O_{3,2}$ is assigned to M_2, and the workload of M_2 is updated to 7. Finally, $O_{3,3}$ is assigned to M_3, and the workload of M_3 is updated to 9. It is worth noting that, before rescheduling, the repair time of the machine should be counted in the workload, and after the operations are assigned to the corresponding machines, the workload of these machines should be updated.



(a) Original scheduling sequence

(b) New scheduling sequence based on rescheduling strategy

Figure 2: Examples of the rescheduling strategy.

2.3. MILP model for EMBFJSP

To model the EMBFJSP, we employed a MILP model. Furthermore, in the subsequent experiments, we adopted the CPLEX solver to solve this MILP model as a way to verify the correctness of our proposed algorithm.

Before modeling for the EMBFJSP, the notations of EMBFJSP are defined as follows:

Indices:

i, i': index of jobs, $i, i' = 1, 2, ..., n$

j, j': index of operations, $j, j' = 1, 2, ..., n_i$

k: index of machines, $k = 1, 2, ..., m$

q: index of machine speeds, $q = 1, 2, .., r$

l: index of position on the machine, $l = 1, 2, .., h$

Parameters:

n: the total number of jobs

n_i: the total number of operations of job i

m: the total number of machines

r: the total number of machine speed levels

h: the total number of positions of machine k

$O_{i,j}$: the jth operation of job i

$V_{i,j,k}$: the processing speed of $O_{i,j}$ on machine k, which contains three speed levels

$TPM_{k,q}$: the turn on/off power of machine k at speed q

$PM_{k,q}$: the processing power of machine k at speed q

$IPM_{k,q}$: the idle power of machine k at speed q

$T_{i,j,k}$: the basic time of operation $O_{i,j}$ on machine k

RT_k: the repair time of breakdown machine k

ST_k: the total turn on/off time of machine k
L: a large enough integer
Decision variables:
$\mathbf{S}_{i,j}$: the start time of operation $O_{i,j}$
$\mathbf{C}_{i,j}$: the completion time of operation $O_{i,j}$
$\mathbf{B}_{k,l,q}$: the start time of machine k on position l at speed q
$\mathbf{F}_{k,l,q}$: the finish time of machine k on position l at speed q
$\mathbf{P}_{i,j,k,q}$: the processing time of operation $O_{i,j}$ on machine k at speed q
$\mathbf{E_t}$: the total turn on/off energy consumption
$\mathbf{E_w}$: the total processing energy consumption
$\mathbf{E_i}$: the total idle energy consumption
TEC: the total energy consumption
$\mathbf{C_{max}}$: the makespan
$\mathbf{X}_{i,j,k,q}$: if operation $O_{i,j}$ is processed on machine k at speed q, $\mathbf{X}_{i,j,k,q} = 1$; otherwise, $\mathbf{X}_{i,j,k,q} = 0$
$\mathbf{Z}_{k,q}$: if machine k processes at speed q, $\mathbf{Z}_{k,q} = 1$; otherwise, $\mathbf{Z}_{k,q} = 0$
$\mathbf{U}_{i,j,k,q,l}$: if the position l of machine k at speed q is selected for operation $O_{i,j}$, $\mathbf{U}_{i,j,k,q,l} = 1$; otherwise, $\mathbf{U}_{i,j,k,q,l} = 0$
$\mathbf{W}_{i,j,k}$: If machine k breakdown while processing operation $O_{i,j}$, $\mathbf{W}_{i,j,k} = 1$; otherwise, $\mathbf{W}_{i,j,k} = 0$
The objectives of EMBFJSP consist of makespan and **TEC**, which are described in detail below.
(1) *Makespan criterion:* Makespan is often regarded as an efficiency indicator in scheduling problems. In other words, the magnitude of the makespan value can indicate the productivity of the enterprise. Thus, the objective of makespan $\mathbf{C_{max}}$ in EMBFJSP can be defined as follows:

$$\min F_1 = \mathbf{C_{max}} = \max \mathbf{C}_{i,j}, \forall i = 1, ..., n; \forall j = 1, ..., n_i \tag{1}$$

(2) **TEC** *criterion:* **TEC** criterion is not only a critical green indicator of the environment but also an economic indicator of an enterprise, which can reflect the contribution to the environment and the economic benefits of the enterprise. The **TEC** contains three components: turn on/off energy consumption, processing energy consumption, and idle energy consumption. Therefore, the **TEC** objective of EMBFJSP can be defined as follows:

$$\min F_2 = \mathbf{TEC} = \mathbf{E_t} + \mathbf{E_w} + \mathbf{E_i} \tag{2}$$

$$\mathbf{E_t} = \sum_{k=1}^{m} \sum_{q=1}^{r} TPM_{k,q} \cdot ST_k \cdot \mathbf{Z_{k,q}} \tag{3}$$

$$\mathbf{E_w} = \sum_{i=1}^{n} \sum_{j=1}^{n_i} \sum_{k=1}^{m} \sum_{q=1}^{r} PM_{k,q} \cdot P_{i,j,k,q} \cdot \mathbf{X}_{i,j,k,q} \tag{4}$$

$$\mathbf{E_i} = \sum_{k=1}^{m} \sum_{l=2}^{h} \sum_{q=1}^{r} IPM_{k,q} \cdot (\mathbf{B}_{k,l,q} - \mathbf{F}_{k,l-1,q}) \tag{5}$$

In conclusion, the MILP model of EMBFJSP is described as follows:

$$\begin{cases} \min F_1 = \mathbf{C_{max}} \\ \min F_2 = \mathbf{TEC} \end{cases} \tag{6}$$

Subject to:

$$\begin{aligned} &P_{i,j,k,q} = T_{i,j,k}/V_{i,j,k}, \\ &\forall i = 1, ..., n; \forall j = 1, ..., n_i; \forall k = 1, ..., m; \forall q = 1, ..., r \end{aligned} \tag{7}$$

$$\begin{aligned} &\sum_{i=1}^{n} \sum_{j=1}^{n_i} \mathbf{U}_{i,j,k,q,l} \geqslant \sum_{i'=1}^{n} \sum_{j'=1}^{n_{i'}} \mathbf{U}_{i',j',k,q,(l+1)}, \\ &\forall k = 1, ..., m; \forall q = 1, ..., r; \forall l = 1, ..., h-1 \end{aligned} \tag{8}$$

$$\begin{aligned} &\sum_{k=1}^{m} \sum_{q=1}^{r} \sum_{l=1}^{h} \mathbf{U}_{i,j,k,q,l} = 1, \\ &\forall i = 1, ..., n; \forall j = 1, ..., n_i \end{aligned} \tag{9}$$

$$\begin{aligned} &\sum_{i=1}^{n} \sum_{j=1}^{n_i} \mathbf{U}_{i,j,k,q,l} \leqslant 1, \\ &\forall k = 1, ..., m; \forall q = 1, ..., r; \forall l = 1, ..., h \end{aligned} \tag{10}$$

$$\begin{aligned} &\mathbf{S}_{i,j} + \sum_{k=1}^{m} \sum_{q=1}^{r} (P_{i,j,k,q} \cdot \mathbf{X}_{i,j,k,q} + RT_k \cdot \mathbf{W}_{i,j,k}) \leqslant \mathbf{S}_{i,j+1}, \\ &\forall i = 1, ..., n; \forall j = 1, ..., n_i - 1 \end{aligned} \tag{11}$$

$$\mathbf{S}_{i,j+1} \geqslant \mathbf{C}_{i,j}, \forall i = 1, ..., n; \forall j = 1, ..., n_i - 1 \tag{12}$$

$$\begin{aligned} &\mathbf{B}_{k,l+1,q} - \mathbf{B}_{k,l,q} \geqslant \sum_{i=1}^{n} \sum_{j=1}^{n_i} (P_{i,j,k,q} \cdot \mathbf{U}_{i,j,k,q,l}), \\ &\forall k = 1, ..., m; \forall q = 1, ..., r; \forall l = 1, ..., h-1 \end{aligned} \tag{13}$$

$$\begin{aligned} &\mathbf{B}_{k,l,q} \geqslant \mathbf{S}_{i,j} - L \cdot (1 - \mathbf{U}_{i,j,k,q,l}), \\ &\forall i = 1, ..., n; \forall j = 1, ..., n_i; \forall k = 1, ..., m; \forall q = 1, ..., r; \forall l = 1, ..., h \end{aligned} \tag{14}$$

$$\begin{aligned} &\mathbf{B}_{k,l,q} \leqslant \mathbf{S}_{i,j} + L \cdot (1 - \mathbf{U}_{i,j,k,q,l}), \\ &\forall i = 1, ..., n; \forall j = 1, ..., n_i; \forall k = 1, ..., m; \forall q = 1, ..., r; \forall l = 1, ..., h \end{aligned} \tag{15}$$

$$\mathbf{C}_{i,j} = \mathbf{S}_{i,j} + \sum_{k=1}^{m}\sum_{q=1}^{r}(P_{i,j,k,q} \cdot \mathbf{X}_{i,j,k,q} + RT_k \cdot \mathbf{W}_{i,j,k}), \quad \forall i = 1, ..., n; \forall j = 1, ..., n_i \tag{16}$$

$$\mathbf{C_{max}} = \max \mathbf{C}_{i,n_i}, \quad \forall i = 1, ..., n \tag{17}$$

$$\mathbf{S}_{i,j} \geqslant 0; \mathbf{C}_{i,j} \geqslant 0, \forall i = 1, ..., n; \forall j = 1, ..., n_i; \tag{18}$$

$$\mathbf{B}_{k,l,q} \geqslant 0; \mathbf{F}_{k,l,q} \geqslant 0, \forall k = 1, ..., m; \forall l = 1, ..., h; \forall q = 1, 2, ..., r \tag{19}$$

$$\mathbf{X}_{i,j,k,q}, \mathbf{Z}_{k,q}, \mathbf{U}_{i,j,k,q,l}, \mathbf{W}_{i,j,k} \in \{0, 1\}, \quad \forall i = 1, ..., n; \forall j = 1, ..., n_i; \forall k = 1, ..., m; \forall q = 1, ..., r; \forall l = 1, ..., h \tag{20}$$

Here, Formula (6) is the objectives, including makespan and **TEC**. Formula (7) is to calculate the actual processing time, the faster the processing speed, the shorter the actual processing time. Formula (8) ensures that the positions of each machine are sequentially assigned to the operations. Formula (9) ensures that each operation can only correspond to one position of each machine. Formula (10) ensures that each position of the machines can process one operation at most. Formula (11) and Formula (12) guarantee the constraint relationship between the operations of the job, for the adjacent operations, the successor operation can only be started processing after the predecessor operation is completed. Formula (13) guarantees that a machine can only process one operation at any given time, as each operation needs to occupy a position on the machine to be processed. Formula (14) and Formula (15) define the relationship between the start time of machines and the start time of operations. Formula (16) defines the completion time of each operation. Formula (17) defines the makespan. Formula (18) - Formula (20) determine the limitations on the range of values. There are $n(1 + 4n_i) + 3mrh + nn_imr(1 + 2h)$ constraints, $2nn_i + 2mh$ continuous variables, and $nn_imr + n^2n_i^2 + mr + nn_imrh + nn_im$ binary decision variables.

3. The proposed algorithm: KTMA

3.1. Framework of KTMA

Our proposed KTMA is based on the two-stage framework which is stated in Algorithm 1, and it consists of two stages: the first stage is to find as many optimal solutions as possible, and the second stage is to further optimize the **TEC** criteria based on the optimal solutions found in the first stage. In the first stage, a memetic algorithm framework is applied to accelerate the convergence performance of the KTMA, thus obtaining a sufficient number of non-dominated solutions. Afterward, a hybrid initialization strategy that consists of three heuristic methods is utilized for generating a high-quality population. Then, a knowledge-driven variable neighborhood search strategy that adjusts

the position of operations on the critical path is designed to fully explore the solution space to improve the convergence performance efficiently. In the second stage, an efficient energy-saving strategy is designed to further optimize the total energy consumption objective without deteriorating the makespan objective by delaying the start time of operations and turning off specific machines. In the following sections, we will introduce the components of each stage step by step.

Algorithm 1 The Framework of KTMA

Input: PS (population size), G (maximum number of iterations), P_c (crossover rate), P_m (mutation rate)
Output: $\mathcal{PF}$ (Pareto solution set)

// First stage
$Gen \leftarrow 1$, $t \leftarrow 0$, $\mathcal{PF} \leftarrow \emptyset$.
$\mathcal{P}_0 \leftarrow$ Initialization(PS).
while $Gen \leqslant G$ **do**
  $MatePool_t \leftarrow$ TournamentSelection($\mathcal{P}_t$).
  $\mathcal{C}_t \leftarrow$ GeneticOperation($MatePool_t, P_c, P_m$).
  $\mathcal{P}_t \leftarrow \mathcal{P}_t \cup \mathcal{C}_t$.
  $\mathcal{S}_t \leftarrow$ Randomly select $PS/4$ individuals for variable neighborhood search.
  $\mathcal{Q}_t \leftarrow$ VariableNeighborhoodSearch($\mathcal{S}_t$).
  $\mathcal{P}_t \leftarrow \mathcal{P}_t \cup \mathcal{Q}_t$.
  $\{F_1, F_2, ..., F_{last}, F_n\} \leftarrow$ FastNonDominatedSort($\mathcal{P}_t$).
  $\mathcal{P}_t \leftarrow$ DeleteDuplicateIndividuals($\mathcal{P}_t$).
  $k \leftarrow 1$, $\mathcal{P}_{t+1} \leftarrow \emptyset$.
  while $|\mathcal{P}_{t+1}| + |F_k| \leqslant PS$ **do**
    $\mathcal{P}_{t+1} \leftarrow \mathcal{P}_{t+1} \cup F_k$.
    $k \leftarrow k + 1$.
  end while
  $W \leftarrow PS - |\mathcal{P}_{t+1}|$.
  $F_{last} \leftarrow$ CrowdingDistance(F_{last}).
  $\mathcal{P}_{t+1} \leftarrow \mathcal{P}_{t+1} \cup F_{last}(1:W,:)$.
  $\mathcal{PF} \leftarrow$ UpdatePF($\mathcal{PF}$).
end while
// Second stage
for *i = 1 to* $|\mathcal{PF}|$ **do**
  $\mathcal{PF}(i) \leftarrow$ DelayedStartTimeStrategy($\mathcal{PF}(i)$).
  $\mathcal{PF}(i) \leftarrow$ MachineTurnoffStrategy($\mathcal{PF}(i)$).
end for

3.2. Encoding and decoding

This section describes how to implement encoding and decoding. Three vectors are applied to represent a chromosome: the operation sequence vector (OSV), the machine sequence vector (MSV), and the speed sequence vector (SSV), which can be presented in Figure 3. For OSV, each gene i is sequentially encoded by the job number. The numbering sequence between jobs indicates the processing sequence of jobs. The jth appearance of the job number indicates the jth operation $O_{i,j}$ of job J_i. For MSV, all operations are arranged in

ascending order that are allocated by machines from left to right. For SSV, the genes of MSV and the genes of SSV maintain a one-to-one mapping, which indicates the processing speed of the machine for processing $O_{i,j}$. Moreover, the length of OSV, MSV, and SSV is equivalent to the total number of operations.

When decoding, each operation will start decoding as early as possible according to the constraints between machines and jobs. First, all operations are divided into a processing sequence according to their order in the OSV. Second, each operation in the processing sequence is sequentially allocated to the selected machine number from the MSV. Finally, the selected machines are assigned to the given processing speed from the SSV.

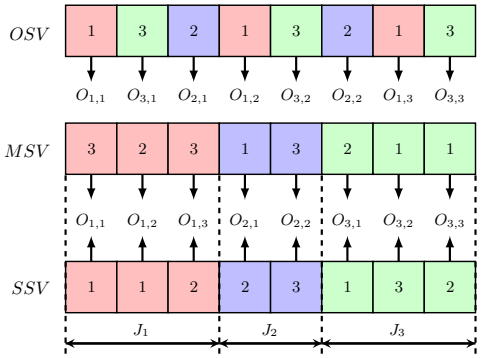


Figure 3: Encoding scheme.

3.3. Hybrid initialization strategy

Since the quality of the initial population often affects the speed of convergence of the algorithm to find satisfactory solutions, the initialization strategy becomes an important part of the evolutionary process. Therefore, the core purpose is to design an effective initialization strategy to generate a high-quality initial population. Generally, a simple single initialization strategy cannot obtain a high-quality population. Therefore, the hybrid initialization strategy is proposed to solve this problem, which is described as follows:

Minimum global machine workload heuristic ($\mathcal{H}_1$): Select the machine with the smallest total machine workload from its candidate machine set, which can reduce makespan by balancing machine workload.

Minimum processing time heuristic ($\mathcal{H}_2$): Select the machine with the smallest processing time from its candidate machine set to assign to each operation $O_{i,j}$, which can effectively reduce total energy consumption.

Lower machine processing speed heuristic ($\mathcal{H}_3$): Randomly select a machine and lower the machine processing speed by one level, which will reduce the idle time to a certain extent. If the machine processing speed is at level 1, the processing speed remains constant.

To ensure that the initial population does not fall into the local optimum, as well as to maintain the diversity of the population, a random initialization strategy is applied to the hybrid initialization strategy.

Random initialization strategy ($\mathcal{R}$): Randomly initialize the population to maintain the diversity performance of the initial population.

The process of the hybrid initialization strategy is as follows: Firstly, the OSV and the SSV are randomly generated using the strategy $\mathcal{R}$ to maintain the diversity performance of the population. Then the MSV is generated by utilizing the aforementioned four initialization strategies, which are defined the probabilities as 0.3, 0.2, 0.2, and 0.3. It is worth noting that when using the heuristic $\mathcal{H}_3$, the MSV randomly selects a machine, and the SSV also changes accordingly.

3.4. Genetic operator

The genetic operator is the critical step to perturb the population to generate new individuals, thus achieving an optimal search to explore the feasible space. For crossover operators, an improved precedence operation crossover (IPOX) operator is employed for OSV and a multipoint crossover (MPX) operator is employed for MSV and SSV (Wang et al., 2010). Examples of the IPOX and MPX operators are presented in Figure 4 (a) and Figure 4 (b), respectively. The procedure is described as follows:

For the IPOX operator:

Step 1: Divide all jobs into two sub-job sets $JobSet_1$ and $JobSet_2$ randomly.

Step 2: Select all the jobs belonging to $JobSet_1$ from parent P_1 to move into offspring O_1, select all the jobs belonging to $JobSet_2$ from parent P_2 to move into offspring O_2, and keep their positions unchanged.

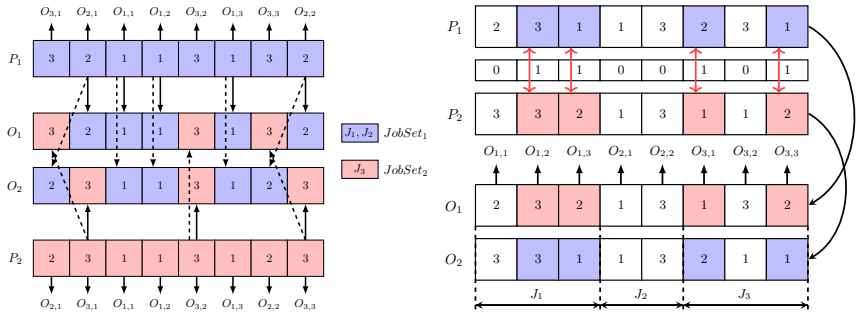
Step 3: Select all the jobs belonging to $JobSet_1$ from parent P_1 to move into offspring O_2, select all the jobs belonging to $JobSet_2$ from parent P_2 to move into offspring O_1, and keep their sequence unchanged.

For the MPX operator:

Step 1: Randomly generate a sequence consisting of integers 0 and 1 that equals the length of the chromosome.

Step 2: Find the position TP with the value 1 in the 0-1 sequence.

Step 3: Swap the machines or machine speeds in parent P_1 and parent P_2 corresponding to position TP and the other machines or machine speeds in parent P_1 and parent P_2 remain unchanged.



(a) IPOX operator (b) MPX operator

Figure 4: Examples of the crossover operators.

For mutation operators, a swapping mutation (SM) operator is employed for OSV and a multi-point mutation (MPM) operator is employed for MSV and SSV (Deng et al., 2017). Examples of the SM and MPM operators are presented in Figure 5 (a) and Figure 5 (b), respectively. The procedure is described as follows:

For the SM operator:

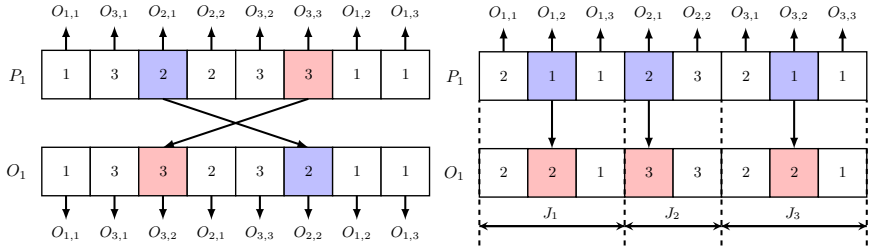
Step 1: Select two positions TP_1 and TP_2 in parent P_1 randomly.

Step 2: Exchange the jobs of positions TP_1 and TP_2 in parent P_1 to generate offspring O_1.

For the MPM operator:

Step 1: Randomly select several positions TP in parent P_1.

Step 2: Randomly replace the machines or machine speeds of parent P_1 in these positions with the candidate machine set or candidate machine speed set to generate offspring O_1.



(a) SM operator (b) MPM operator

Figure 5: Examples of the mutation operators.

3.5. Knowledge-driven variable neighborhood search

Variable neighborhood search has become a very critical part of the scheduling problems. Especially, an effective design of a variable neighborhood search can greatly facilitate the search capabilities of the algorithm. By analyzing the characteristics of EMBFJSP, we added problem-specific knowledge to the variable neighborhood search and constructed four knowledge-driven variable neighborhood search operators as a way to enhance the convergence of the algorithm. The EMBFJSP aims to optimize two objectives: makespan and **TEC**. The makespan is mainly related to the operations on the critical path, and the **TEC** is mainly related to the idle time of the machines. Therefore, when designing the knowledge-driven variable neighborhood search operators, we mainly consider how to optimize the operation sequence on the critical path and how to reduce the idle time of the machines. The four knowledge-driven variable neighborhood search operators are designed as follows:

3.5.1. Neighborhood operator 1

Neighborhood operator 1 ($\mathcal{N}_1$) aims to adjust the position of the operation on the critical block, which contains continuous operations on the same machine on the critical path. The critical path can be defined as the longest path from the start node to the end node of the disjunctive graph. A more detailed description of the disjunctive graphs can be referred to (Zhang et al., 2007). Neighborhood operator 1 ($\mathcal{N}_1$) randomly selects the operations on the critical path and adjusts them to the head of the critical block. An example of $\mathcal{N}_1$ is shown in Figure 6 (a).

From Figure 6 (a), we can find the critical path (consisting of $O_{2,1}$, $O_{2,2}$, $O_{3,2}$, $O_{1,3}$ and $O_{3,3}$) and randomly select an operation $O_{3,2}$ and move it to the head

of the critical block (consisting of $O_{2,2}$, $O_{3,2}$ and $O_{1,3}$) satisfying its constraints. Meanwhile, $O_{3,3}$ can also move forward to connect its predecessor operation ($O_{3,2}$) and thus shorten the makespan of the original sequence.

3.5.2. Neighborhood operator 2

Neighborhood operator 2 ($\mathcal{N}_2$) is also based on the critical block and randomly selects the operations on the critical path and adjusts them to the tail of the critical block. An example of $\mathcal{N}_2$ is shown in Figure 6 (b).

From Figure 6 (b), we can find the critical path (consisting of $O_{2,1}$, $O_{2,2}$, $O_{1,3}$, $O_{3,2}$ and $O_{3,3}$) and randomly select an operation $O_{1,3}$ and move it to the tail of the critical block (consisting of $O_{2,2}$, $O_{1,3}$ and $O_{3,2}$) satisfying its constraints. At the same time, $O_{3,3}$ can also move forward to connect its predecessor operation ($O_{3,2}$) that can shorten the makespan of the original sequence.

3.5.3. Neighborhood operator 3

Neighborhood operator 3 ($\mathcal{N}_3$) aims to move the operations on the critical path, which randomly selects an operation on the critical path and inserts it into a different candidate machine. An example of $\mathcal{N}_3$ is shown in Figure 6 (c).

From Figure 6 (c), we can find the critical path (consisting of $O_{2,1}$, $O_{2,2}$, $O_{1,3}$, $O_{3,2}$ and $O_{3,3}$) and randomly select an operation $O_{1,3}$ and insert it to M_1. $O_{3,2}$ and $O_{3,3}$ can move ahead accordingly, resulting in a smaller makespan value than the original sequence.

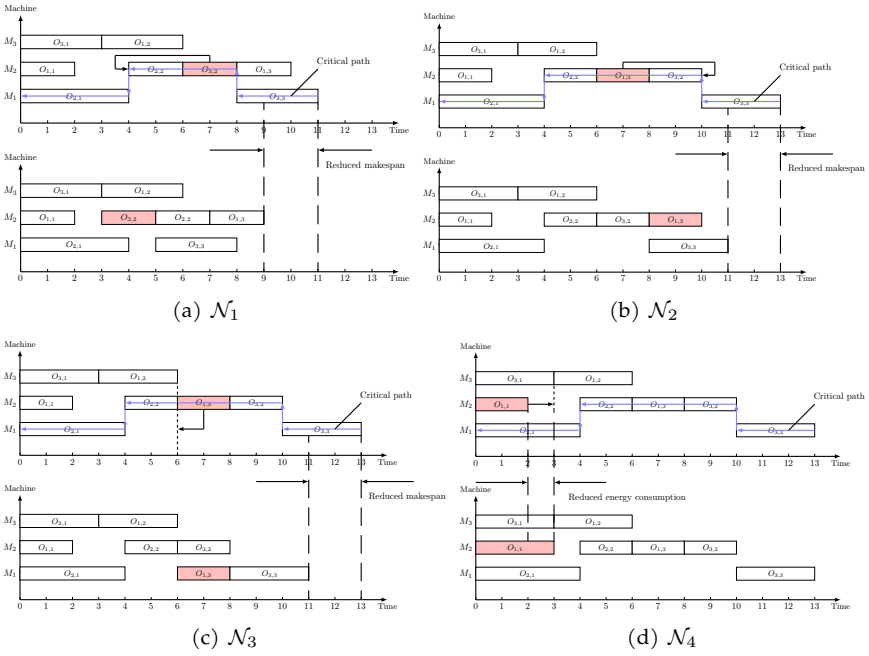
3.5.4. Neighborhood operator 4

Neighborhood operator 4 ($\mathcal{N}_4$) is based on the machine speeds which randomly select an operation on the non-critical path where subsequent idle time exists and lower the machine speed of its processing machine by one level. The energy consumption generated by processing time and the energy consumption generated by idle time are large in **TEC**. Therefore, by selecting operations on the non-critical path and reducing the processing speed of their processing machines, the power of the processing machines can be reduced and the idle time of the machines can also be reduced. Thus, the **TEC** can be effectively reduced. An example of $\mathcal{N}_4$ is shown in Figure 6 (d).

From Figure 6 (d), we randomly select $O_{1,1}$ on the non-critical path and lower the machine speed to satisfy its constraints. The processing speed of $O_{1,1}$ becomes slower, resulting in longer processing time, and the total energy consumption of this sequence after using the $\mathcal{N}_4$ is less than the original.

3.6. Energy-saving strategy

Energy-saving strategy is especially important in the energy-efficient scheduling problem. An effective energy-saving strategy can reduce **TEC** without increasing makespan, thus significantly increasing industry profits and efficiently protecting the environment. **TEC** depends mainly on the processing time and the idle time; therefore, we design two types of energy-saving strategies based on the previous.



(a) $\mathcal{N}_1$ (b) $\mathcal{N}_2$

(c) $\mathcal{N}_3$ (d) $\mathcal{N}_4$

Figure 6: Examples of the neighborhood operators.

3.6.1. *Delayed start time strategy*

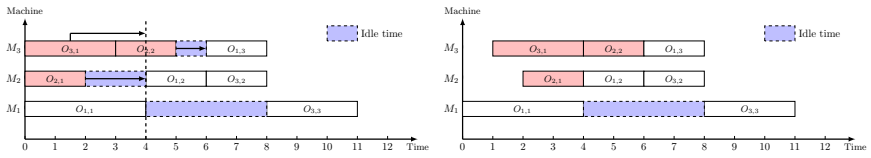
The delayed start time strategy (DSTS) is designed to reduce idle time in the scheduling sequence. By delaying the start time of the operations, the idle time of machines can be reduced, thus optimizing **TEC**. However, it is not possible to delay the start time of operations as long as there exists idle time, the operations with delayed start times must satisfy their constraints ; that is, adjacent operations of the same job must finish processing the predecessor operation before starting to process the successor operation, and a machine can only process one operation at a time. With the execution of the delayed start time strategy, although the OSV, MSV, and SSV cannot be changed, it can delay the start time of the jobs and obtain the start time and completion time of the jobs, thus reducing the idle time of the machines to minimize the **TEC** criteria. The detail of the delayed start time strategy is presented in Algorithm 2.

Algorithm 2 Delayed Start Time Strategy

Input: J (job sequence), O (operation sequence), Po (successor operation pointer), Pm (machine successor operation pointer), S (start time of operations), C (completion time of operations)
Output: S (start time of operations), C (completion time of operations)

for i = Length(O) to 1 **do**
  $Jc = J(i), Oc = O(i)$ //The current job and operation
  if $Po = \emptyset \wedge Pm \neq \emptyset$ **then**
    //The operation is the last operation of the job
    $Jm = J(Pm(i)), Om = O(Pm(i))$ //The successor job and operation on the machine
    $time = C(Jc, Oc) - S(Jc, Oc)$
    $C(Jc, Oc) = S(Jm, Om)$
    $S(Jc, Oc) = C(Jc, Oc) - time$
  else if $Po \neq \emptyset \wedge Pm \neq \emptyset$ **then**
    $Os = O(Po(i))$ //The successor operation
    $Jm = J(Pm(i)), Om = O(Pm(i))$
    $time = C(Jc, Oc) - S(Jc, Oc)$
    $C(Jc, Oc) = min(C(Jc, Os), C(Jm, Om))$
    $S(Jc, Oc) = C(Jc, Oc) - time$
  end if
end for

To better explain, an example is shown in Figure 7. As shown in the original scheduling sequence in (a) of Figure 7, 7 units of idle time exist and DSTS can be utilized to further reduce the idle time, where OSV = {1, 2, 3, 2, 1, 1, 3, 3}, MSV = {1, 2, 3, 2, 3, 3, 2, 1}, and SSV = {1, 3, 3, 2, 2, 2, 3, 2}. Traversing the OSV from backward to forward, $O_{2,2}$ can be delayed without violating the constraints. After $O_{2,2}$ is delayed, $O_{3,1}$ and $O_{2,1}$ have the condition that can be delayed, so $O_{3,1}$ and $O_{2,1}$ are delayed accordingly. The scheduling sequence using DSTS is shown in Figure 7 (b), where the idle time can be reduced to 4 unit time; thus, the **TEC** can be reduced.



(a) Original scheduling sequence (b) Scheduling sequence using DSTS

Figure 7: Examples of the delayed start time strategy.

3.6.2. Machine turn off strategy

After the execution of DSTS, we designed the machine turn off strategy (MTOS) to further optimize **TEC** criteria by reducing machine idle time. When a machine has finished processing an operation and is waiting to process the next operation, it will incur idle time. If the energy consumed when the machine is turned off and on is less than the energy consumed when the machine is

idle, turn the machine off; otherwise, keep the machine in an idle state. Through the machine turn off strategy, it is possible to determine which machines can be turned off during idle time, and to determine the time of shutdown, thus reducing the idle time of the machines to optimize the **TEC** objective.

4. Experimental results

In this section, sufficient experiments were designed to evaluate the performance of KTMA. In addition, all algorithms are coded in MATLAB R2020b on the Intel Core i7-7700 CPU @ 3.60GHz with 8G RAM.

4.1. Experimental instances

To verify the performance of our proposed KTMA, the benchmark example is selected from (Wu & Sun, 2018) and referenced the DFJSP case from (Li et al., 2021) with modifications to its machine breakdown part. Table S-I presents an example of benchmark Ins01, which contains 10 jobs, 6 machines, and 6 maximum number of operations. Since there are 3 levels of machine processing speeds, each level has a different power, the processing power, the idle power and the turn on/off power are also different. As shown in Table 1, columns 2–7 listed the processing power and the idle power for each level. In addition, the last column listed the turn on/off power.

Table 1: The power distribution for each machine.

	Level 1		Level 2		Level 3		
	processing	idle	processing	idle	processing	idle	turn on/off
M_1	31.21	6.76	49.06	19.16	57.00	26.07	85.84
M_2	35.90	2.89	48.80	10.01	56.39	21.92	88.63
M_3	32.26	6.72	48.18	14.62	50.34	27.38	80.31
M_4	33.85	6.95	42.61	14.24	50.69	22.43	99.68
M_5	35.83	1.68	45.94	14.61	53.20	29.17	83.34
M_6	32.52	2.55	40.23	17.70	55.31	22.69	82.12
M_7	32.90	2.24	44.25	13.22	56.54	27.66	87.45
M_8	36.17	6.68	43.13	17.85	54.08	21.89	83.96
M_9	32.65	8.44	41.61	14.71	58.20	22.87	89.79
M_{10}	38.24	3.44	41.79	10.36	57.18	20.91	86.79
M_{11}	39.83	7.81	44.23	11.76	59.69	25.76	99.03
M_{12}	37.30	6.75	40.94	17.22	55.31	26.83	98.41
M_{13}	33.44	5.67	45.99	14.73	53.25	27.47	81.05
M_{14}	35.84	6.02	44.71	11.53	51.06	24.26	94.76
M_{15}	31.08	3.87	46.96	13.41	56.11	26.66	85.38

4.2. Performance metrics

To fully evaluate the performance of the proposed KTMA, three evaluation indicators are adopted, which are the hypervolume (HV) metric (While et al., 2006), generation distance (GD) metric (Zitzler et al., 2000) and Spread metric (Deb et al., 2002).

The HV metric is usually utilized to evaluate the comprehensive performance of the multi-objective optimization algorithm (MOEAs), and the MOEAs with higher HV values tend to have better comprehensive performance. HV is calculated as follows:

$$HV(P,R) = \bigcup_{x \in P}^{P} v(x,R) \tag{21}$$

where P indicates the PF obtained by the MOEAs, and R indicates the reference point which is often set to $(1.01, 1.01)$. x indicates all non-dominated solution in PF that has been normalized and v indicates the volume of the hypercube enclosed by x and R.

The GD metric is used to evaluate the convergence performance of the MOEAs, and the MOEAs with lower GD values tend to have better convergence performance. GD is calculated as follows:

$$GD(P,P^*) = \frac{\sqrt{\sum_{y \in P} \min_{x \in P^*} dis(x,y)^2}}{|P|} \tag{22}$$

where P indicates the non-dominated solutions set of the MOEAs and P^* indicates the reference PF obtained by all MOEAs. $dis(x,y)$ indicates the Euclidean distance between the point $x \in P^*$ and the point $y \in P$.

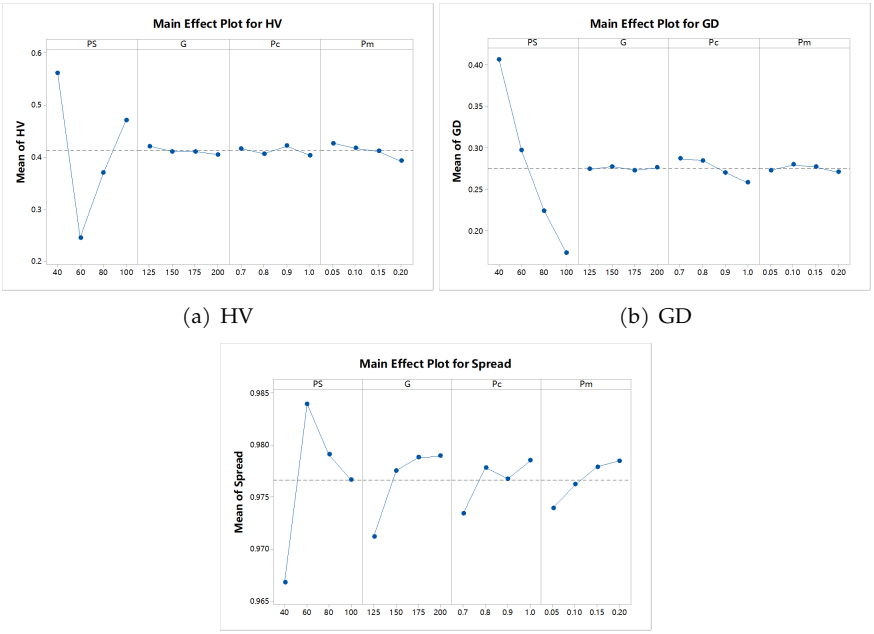
The Spread metric is applied to evaluate the diversity performance of the MOEAs, and the MOEAs with lower Spread values tend to have better diversity performance. Spread is calculated as follows:

$$Spread = \frac{d_l + d_f + \sum_{i=1}^{N-1} |d_i - \overline{d}|}{d_l + d_f + (N-1)\overline{d}} \tag{23}$$

where d indicates the Euclidean distance and $\overline{d}$ indicates the average distance of $d_i, i = 1, 2, ..., (N-1)$. d_l and d_f indicate the Euclidean distances between the extreme points and the boundary points of the obtained non-dominated set.

4.3. Parameter calibration

The importance of parameter configuration to the proposed KTMA in solving EMBFJSP cannot be overstated. In the proposed KTMA, four critical parameters should be calibrated, which are population size PS, maximum number of iterations G, crossover probability P_c, and mutation probability P_m. To obtain the best setting for these parameter combinations, a Taguchi method of design-of-experiments (DOE) (Roy & K, 2001) is adopted. More precisely, each parameter has four different levels; that is, $PS = \{40, 60, 80, 100\}$, $G = \{125, 150, 175, 200\}$, $P_c = \{0.7, 0.8, 0.9, 1.0\}$ and $P_m = \{0.05, 0.10, 0.15, 0.20\}$ and an orthogonal array $L_{16}(4^4)$ was employed in this parameter calibration experiment. To ensure a fair comparison of experiments, each parameter combination ran 20 times independently. Figure 8 presents the main effects plots of four parameters for three metrics. Generally, the combination of parameters with a higher HV value has better performance. On the contrary, the combination of parameters with a lower HV value and a lower Spread value has better performance. From the experimental results and the comprehensive observation, it is clear that the best parameter combination is $PS = 100$, $G = 125$, $P_c = 0.9$, and $P_m = 0.05$.



(a) HV (b) GD

(c) Spread

Figure 8: Main effects plots of three metrics: (a) HV, (b) GD, and (c) Spread.

4.4. *MILP model validation*

To verify the correctness of the MILP model, the CPLEX 12.6.3 solver was applied to obtain the optimal solutions for the optimization objectives of the EMBFJSP. Since the CPLEX solver can only obtain the value of one objective at a time, and EMBFJSP is a bi-objective problem, therefore the MILP model needs to be transformed into a single-objective problem and solved in two times. As the scale of the EMBFJSP increases, the constraints increase dramatically, it is impractical to obtain the optimal solution set in a finite time. Therefore, we utilized the CPLEX solver to solve the smallest size instance to verify the correctness of the MILP model. The size of the smallest instance is as follows: the number of jobs is 10, the number of machines is 6, and the maximum number of operations is 6. Not only that, some larger instances like Ins05 and Ins10 were also selected to validate the correctness of the MILP model, which consisted of 15 jobs, 4 machines, 9 maximum number of operations, and 20 jobs, 15 machines, 14 maximum number of operations, respectively. Our proposed KTMA also executed this instance as a comparison.

In order to compare the optimization performance of the CPLEX and the KTMA, the Relative Percentage Difference (RPD) (Hatami et al., 2015) metric is employed for evaluation, which is calculated as follows:

$$RPD = \frac{R_1 - R_2}{R_2} \times 100\% \tag{24}$$

where R_1 denotes the objective values obtained from the KTMA, and R_2 denotes the objective values obtained from the CPLEX.

Additionally, the CPU time consumed by the CPLEX and the KTMA is also employed for the comparison. Table 2 records the results.

From Table 2, the solutions obtained by the CPLEX are better than that obtained by the KTMA because the CPLEX uses the branch-and-bound algorithm, which can find the optimal solutions with fast average speed. However, the results obtained by the CPLEX can verify the correctness of our proposed KTMA. Meanwhile, compared to the CPLEX, our proposed KTMA consumes less CPU time.

Table 2: Comparison results of the CPLEX solver and the KTMA on the selected instances.

Instance	makespan		TEC		RPD		CPU time	
	CPLEX	KTMA	CPLEX	KTMA	makespan	TEC	CPLEX	KTMA
Ins01	39.00	45.50	9.83E+03	1.02E+04	16.67%	3.76%	90.25	70.92
Ins05	231.75	257.50	4.32E+04	4.51E+04	11.11%	4.40%	267.83	131.98
Ins10	453.50	482.00	1.67E+05	1.71E+05	6.28%	2.40%	970.19	667.70

4.5. Effectiveness of each improvement part of KTMA

To verify the effectiveness of each improvement part of the proposed algorithm, three variants of KTMA are compared as follows: KTMA1 denotes KTMA without the hybrid initialization strategy, KTMA2 denotes KTMA without the knowledge-driven variable neighborhood search operation and KTMA3 denotes KTMA without the energy-saving strategy. To ensure the fairness of the comparison, the KTMA and its all variants are run 20 times in all instances. The statistical values of HV, GD, and Spread of all variant algorithms over 20 times runs independently in all instances are listed in Table 3, Table 4 and Table 5, where the best values are marked in **bold**. We can see that in most instances, the performance of KTMA is not inferior to that of all variants of KTMA for HV, GD, and Spread metrics. Moreover, Table 6 lists the Friedman rank test results for all variants of the algorithm with a confidence level $\alpha = 0.05$. Therefore, based on the comprehensive performance of the three metrics in the table, KTMA has the highest overall ranking, which indicates that the KTMA outperforms its four variants.

Table 3: HV statistical values of all variant algorithms in all instances.

Instances	KTMA1		KTMA2		KTMA3		KTMA	
	mean	std	mean	std	mean	std	mean	std
Ins01	0.5257	0.1761	0.4072	0.1867	0.6259	0.1332	**0.6911**	0.1290
Ins02	0.3418	0.1287	0.2242	0.1696	0.6428	0.1836	**0.6502**	0.1849
Ins03	0.2429	0.2054	**0.5415**	0.3262	0.3152	0.2004	0.3998	0.2193
Ins04	**0.4006**	0.2017	0.2146	0.1597	0.3052	0.1346	0.3715	0.1457
Ins05	0.3650	0.2942	0.1915	0.1279	0.4707	0.3336	**0.4707**	0.3336
Ins06	0.4273	0.2425	0.3794	0.2775	0.5605	0.2927	**0.6231**	0.3180
Ins07	0.1818	0.1891	**0.3354**	0.3526	0.2068	0.2370	0.2074	0.2376
Ins08	**0.4240**	0.2925	0.2483	0.2027	0.2062	0.2486	0.3014	0.3183
Ins09	0.3159	0.2405	**0.4233**	0.2418	0.2065	0.2009	0.2964	0.2496
Ins10	0.3225	0.2306	**0.5233**	0.2307	0.2543	0.1867	0.4471	0.2107

Table 4: GD statistical values of all variant algorithms in all instances.

Instances	KTMA1		KTMA2		KTMA3		KTMA	
	mean	std	mean	std	mean	std	mean	std
Ins01	0.1081	0.0628	0.1536	0.0624	0.0864	0.0440	**0.0558**	0.0351
Ins02	0.2703	0.0881	0.3547	0.1439	0.1304	0.0889	**0.1176**	0.086864
Ins03	0.3717	0.2423	**0.1365**	0.1142	0.3349	0.1890	0.2735	0.1720
Ins04	0.1493	0.1017	0.2617	0.1394	0.1987	0.0779	**0.1442**	0.0630
Ins05	0.3831	0.2924	0.4865	0.1789	0.2643	0.2618	**0.2641**	0.2615
Ins06	0.1872	0.0933	0.1182	0.0726	0.1067	0.0797	**0.0645**	0.0764
Ins07	0.4788	0.2518	**0.1993**	0.1552	0.4330	0.2199	0.4264	0.2197
Ins08	**0.1842**	0.1295	0.2517	0.1349	0.4316	0.2558	0.2618	0.2222
Ins09	0.3731	0.1678	**0.2850**	0.1407	0.5358	0.3145	0.4276	0.2876
Ins10	0.3072	0.1227	**0.2074**	0.0831	0.3563	0.1590	0.2391	0.1366

Table 5: Spread statistical values of all variant algorithms in all instances.

Instances	KTMA1		KTMA2		KTMA3		KTMA	
	mean	std	mean	std	mean	std	mean	std
Ins01	0.9588	0.0579	0.9817	0.0571	**0.9095**	0.0806	0.9477	0.0883
Ins02	0.9564	0.0430	0.9731	0.0395	**0.8958**	0.0677	0.9150	0.0817
Ins03	0.9797	0.0184	**0.8422**	0.1584	0.9607	0.0424	0.9583	0.0490
Ins04	0.9614	0.0516	0.9824	0.0454	**0.9455**	0.0599	0.9714	0.0588
Ins05	0.9371	0.0790	0.9805	0.0145	**0.9137**	0.1450	0.9139	0.1450
Ins06	0.9843	0.0143	0.9919	0.0142	0.9722	0.0291	**0.9669**	0.0292
Ins07	0.9752	0.0369	**0.9583**	0.0766	0.9712	0.0480	0.9826	0.0204
Ins08	0.9845	0.0273	0.9939	0.0145	0.9886	0.0152	**0.9841**	0.0216
Ins09	0.9801	0.0151	0.9632	0.0260	0.9803	0.0278	**0.9356**	0.2153
Ins10	0.9674	0.0547	**0.9223**	0.0816	0.9738	0.0404	0.9552	0.1072

Table 6: Overall ranks through the Friedman test among all variants (a level of significant $\alpha = 0.05$).

MOEAs	HV		GD		Spread	
	rank	p-value	rank	p-value	rank	p-value
KTMA1	2.30	1.52E-05	2.90	1.24E-05	2.90	2.66E-03
KTMA2	2.30		2.50		2.90	
KTMA3	3.30		2.90		2.20	
KTMA	**2.10**		**1.70**		**2.00**	

4.6. Comparison and discussion among other algorithms

To further verify the performance of the proposed KTMA, we compared it with the classical MOEAs, including SPEA2 (Zitzler et al., 2001), NSGA-II (Deb et al., 2002) and MOEA/D (Zhang & Li, 2007), the current proposed MOEAs, including AdaW (Li & Yao, 2020) and TS-NSGA-II (Ming et al., 2022), a state-of-art algorithm named NSGA-III/ARV (An et al., 2022), a RL-based algorithm named LRVMA (Li et al., 2022b), and a GP-related algorithm named GPHH (Fan et al., 2021). The best parameter settings refer to the corresponding references. For SPEA2, NSGA-II and MOEA/D, $PS = 100$, $P_c = 0.9$ and $P_m = 0.05$. The archive size of SPEA2 is set to 50 and the neighborhood size of MOEA/D is set to 15. Furthermore, for a fair comparison, all comparison algorithms are compared in the same environment, which are coded in MATLAB R2020b on the Intel Core i7-7700 CPU @ 3.60GHz with 8G RAM and run 20 times independently with the same stopping criteria in all instances. ($G = 125$).

Table 7: HV statistical values of all comparison algorithms in all instances.

Instances	SPEA2	NSGA-II	MOEA/D	AdaW	TS-NSGA-II	NSGA-III/ARV	LRVMA	GPHH	KTMA
Ins01	0.8242	0.2677	0.1134	0.0465	0.0458	0.3832	0.1997	0.2952	**0.9432**
Ins02	0.6948	0.2477	0.1301	0.0509	0.0191	0.4628	0.1032	0.3111	**0.9458**
Ins03	0.3308	0.2157	0.1023	0.1042	0.0399	0.4259	0.1758	0.1933	**0.7374**
Ins04	0.7494	0.2131	0.0964	0.0503	0.0207	0.4119	0.1266	0.1543	**0.8789**
Ins05	0.2545	0.0929	0.0473	0.0318	0.0207	0.1089	0.0530	0.0711	**0.6467**
Ins06	0.2277	0.1077	0.0547	0.0408	0.0101	0.3140	0.0668	0.1158	**0.7624**
Ins07	0.2220	0.2112	0.1009	0.0815	0.0625	0.2731	0.1137	0.1477	**0.5439**
Ins08	0.4199	0.1755	0.0631	0.0532	0.0119	0.1465	0.0819	0.0885	**0.4793**
Ins09	0.2312	0.1608	0.0749	0.0323	0.0292	0.2136	0.0813	0.1613	**0.5609**
Ins10	0.1950	0.1789	0.0697	0.0407	0.1012	0.3289	0.1025	0.1341	**0.7212**

Table 8: GD statistical values of all comparison algorithms in all instances.

Instances	SPEA2	NSGA-II	MOEA/D	AdaW	TS-NSGA-II	NSGA-III/ARV	LRVMA	GPHH	KTMA
Ins01	0.0514	0.3310	0.5357	0.6365	0.5407	0.2295	0.4725	0.3096	**0.0123**
Ins02	0.1212	0.3841	0.4860	0.6973	0.6839	0.2297	0.5981	0.3861	**0.0216**
Ins03	0.3223	0.4140	0.5844	0.6202	0.6594	0.2533	0.5222	0.4724	**0.0955**
Ins04	0.0943	0.4057	0.5118	0.7352	0.7636	0.2618	0.5184	0.5457	**0.0207**
Ins05	0.4280	0.5742	0.6586	0.8747	0.8990	0.5290	0.7209	0.7096	**0.0939**
Ins06	0.3639	0.5265	0.6987	0.7620	0.9061	0.3369	0.6469	0.6135	**0.0335**
Ins07	0.4574	0.4083	0.6031	0.7185	0.7431	0.4077	0.6317	0.5441	**0.1725**
Ins08	0.2515	0.3879	0.5856	0.8264	0.9175	0.5554	0.6188	0.6945	**0.1322**
Ins09	0.4330	0.4603	0.6243	0.8467	0.8832	0.4204	0.6830	0.5504	**0.1737**
Ins10	0.4362	0.4393	0.5535	0.8158	0.7171	0.3624	0.6281	0.5636	**0.1131**

Table 9: Spread statistical values of all comparison algorithms in all instances.

Instances	SPEA2	NSGA-II	MOEA/D	AdaW	TS-NSGA-II	NSGA-III/ARV	LRVMA	GPHH	KTMA
Ins01	**0.9298**	0.9834	0.9772	0.9931	0.9974	0.9657	0.9642	0.9680	0.9537
Ins02	0.9778	0.9876	0.9883	0.9948	0.9990	0.9778	0.9866	0.9799	**0.9146**
Ins03	0.9872	0.9895	0.9828	0.9768	0.9910	0.9726	0.9671	0.9858	**0.9513**
Ins04	0.9580	0.9813	0.9901	0.9892	0.9981	0.9655	0.9749	0.9782	**0.9499**
Ins05	0.9888	0.9921	0.9950	0.9964	0.9960	0.9950	0.9928	0.9957	**0.9179**
Ins06	0.9920	0.9941	0.9908	0.9865	0.9992	0.9935	0.9868	0.9834	**0.9557**
Ins07	0.9897	0.9821	0.9887	0.9898	0.9923	0.9902	0.9843	0.9865	**0.9577**
Ins08	0.9911	0.9941	0.9937	0.9916	0.9985	0.9918	0.9919	0.9931	**0.9557**
Ins09	0.9906	0.9880	0.9870	0.9911	0.9979	0.9884	0.9893	0.9897	**0.9371**
Ins10	0.9917	0.9859	0.9901	0.9898	0.9806	0.9790	0.9842	0.9893	**0.9315**

The statistical values of HV, GD, and Spread of all comparison algorithms are listed in Table 7, Table 8 and Table 9, where the best values are marked in **bold**. As observed in those three tables, in almost all instances, the proposed KTMA significantly outperforms these comparison algorithms in terms of HV, GD, and Spread metrics. Moreover, Table 10 lists the Friedman rank test results for all comparison algorithms with a confidence level $\alpha = 0.05$. Through observation, KTMA ranks best in HV, GD, and Spread metrics, which shows the excellent convergence performance and diversity performance of KTMA. More than that, Figure S-1 - S-2, Figure S-3 - S-4 and Figure S-5 - S-6 present the boxplot comparison of HV, GD and Spread metrics of all comparison algorithms in all instances, where the values obtained by KTMA are better than those obtained by other comparison algorithms, highlighting the advantages of KTMA. Regarding the boxplot of the Spread metric, the box width of KTMA is greater than that of other comparison algorithms, which means that the stability of KTMA in the Spread metric still has room for improvement. Table 11 shows the CPU computation time of all comparison algorithms in all instances,

from which it can be concluded that NSGA-II and MOEA/D consume the least amount of CPU time for all the instances, implying that they run faster for the same number of iterations. Although our proposed KTMA is not superior in terms of CPU computation time metric, the comprehensive performance of our proposed KTMA is the best among all the compared algorithms, therefore, it is worthwhile to consume slightly more CPU computation time.

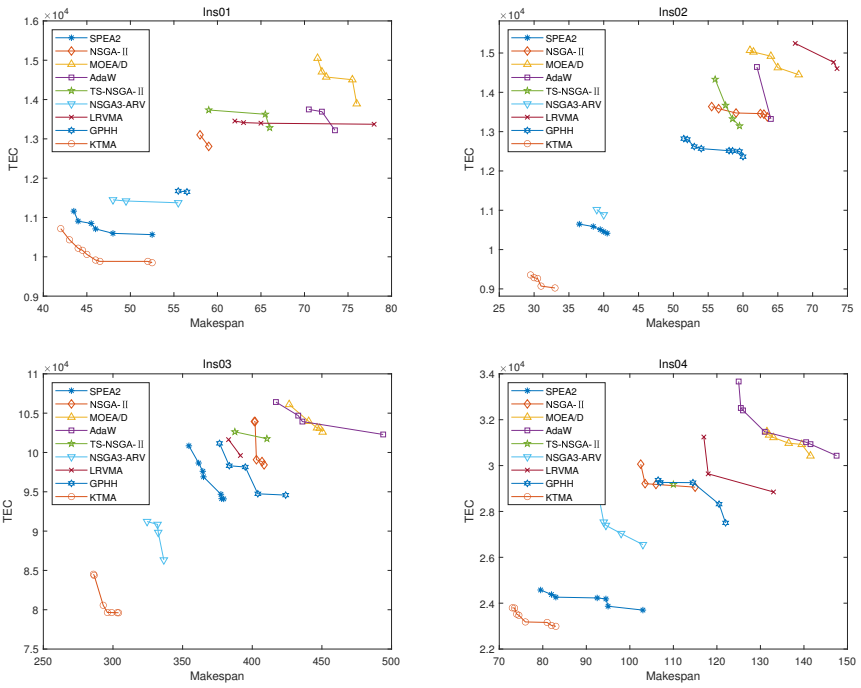


Figure 9: PF comparison results of all comparison algorithms in the instances of Ins01 - Ins04.

Table 10: Overall ranks through the Friedman test among all comparison algorithms (a level of significant $\alpha = 0.05$).

MOEAs	HV		GD		Spread	
	rank	p-value	rank	p-value	rank	p-value
SPEA2	2.00		2.00		4.40	
NSGA-II	3.90		3.90		5.60	
MOEA/D	7.30		5.90		5.80	
AdaW	7.60		8.30		6.40	
TS-NSGA-II	9.00	1.95E-13	8.70	3.93E-13	8.30	1.81E-06
NSGA-III/ARV	2.90		2.60		4.40	
LRVMA	5.70		6.50		3.90	
GPHH	5.40		5.50		5.10	
KTMA	**1.20**		**1.00**		**1.10**	

The excellent convergence and distribution of KTMA benefit from its design. First, a two-stage framework is utilized for improving the convergence performance of the algorithm and maintaining the diversity performance of the population. Second, a rescheduling strategy is applied when machine

breakdowns occur, which can balance the workload of machines, thus reducing makespan. Third, three problem-specific heuristics are proposed to generate a high-quality initial population, which can maintain population diversity. Then, four knowledge-driven variable neighborhood search operators are developed to improve convergence performance, which can fully explore the solution space to obtain better PF solutions. Finally, two types of energy-saving strategies are designed to further reduce **TEC** without increasing makespan, which can increase convergence vastly.

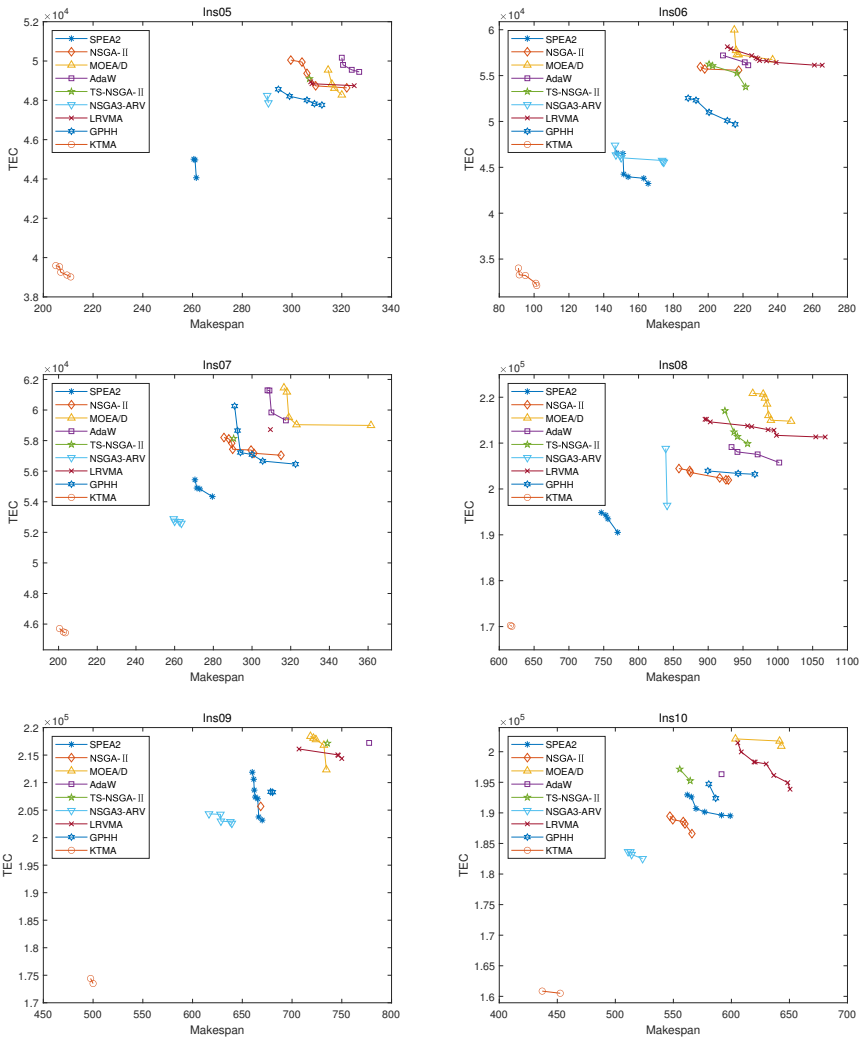


Figure 10: PF comparison results of all comparison algorithms in the instances of Ins05 - Ins10.

Figure 9 - 10 presents the comparison results of the Pareto front obtained by all comparison algorithms in all instances. Regarding the convergence and diversity of the Pareto front, KTMA can obtain better solutions than its competitors. Because KTMA has strong exploration ability, KTMA can explore better PF, while other comparison algorithms can only find inferior PF due to insuffi-

cient exploration solution space ability. Meanwhile, the Pareto front obtained by KTMA is closer to the lower left corner of the graph, which indicates that the non-dominated solutions obtained by KTMA are closer approximations toward real PF. Based on the observation and analysis of the above experimental results, our proposed KTMA can solve EMBFJSP effectively.

Table 11: The CPU computation time of all comparison algorithms in all instances.

Instances	SPEA2	NSGA-II	MOEA/D	AdaW	TS-NSGA-II	NSGA-III/ARV	LRVMA	GPHH	KTMA
Ins01	18.27	25.39	**15.80**	18.59	107.48	37.27	21.99	262.27	70.92
Ins02	18.98	26.00	**17.15**	18.93	113.45	34.10	20.97	215.41	81.14
Ins03	180.12	**121.81**	127.10	149.11	223.27	176.07	243.53	2533.56	261.17
Ins04	**21.60**	27.43	**21.60**	25.19	118.31	49.84	30.87	286.06	81.89
Ins05	63.86	56.49	**53.33**	65.37	170.57	107.99	92.80	982.02	131.98
Ins06	65.20	**41.25**	50.83	48.67	132.87	71.32	76.41	688.05	99.66
Ins07	80.79	**65.80**	70.68	73.56	162.79	95.34	111.32	1164.36	138.95
Ins08	407.03	**268.38**	303.37	338.96	406.64	569.42	601.31	6199.92	529.97
Ins09	527.44	**356.33**	375.52	434.51	511.96	624.26	777.97	7884.94	632.66
Ins10	482.93	**324.12**	351.60	424.50	482.47	582.85	677.82	7340.27	667.70

Figure 11 and Table 12 present the comparison results of all algorithms under different preferences. In the case of preference 1, the makespan objective and the **TEC** objective are of equal importance, and our proposed KTMA can obtain the better solution of (44.50, 1.02E+04), which dominates the solutions obtained by all the other comparison algorithms, and therefore, the proposed KTMA performs better in solving the actual EMBFJSP. In the case of preference 2, it focuses primarily on the makespan objective, and our proposed KTMA can obtain the optimal solution of 42.00, which is improved by 3.45%, 27.59%, 41.25%, 40.43%, 28.81%, 12.50%, 32.26%, and 24.32% respectively compared to other algorithms. Therefore, when focusing on improving the economic efficiency of the enterprises, the proposed KTMA can achieve better performance to improve productivity. In the case of preference 3, it focuses primarily on the **TEC** objective, and our proposed KTMA can obtain the optimal solution of 9.85E+03, which is improved by 7.08%, 23.05%, 29.14%, 25.38%, 25.94%, 13.60%, 26.49%, and 15.81% respectively compared to other algorithms. Therefore, when focusing on energy saving and emission reduction, the proposed KTMA can bring better green benefits to enterprises.

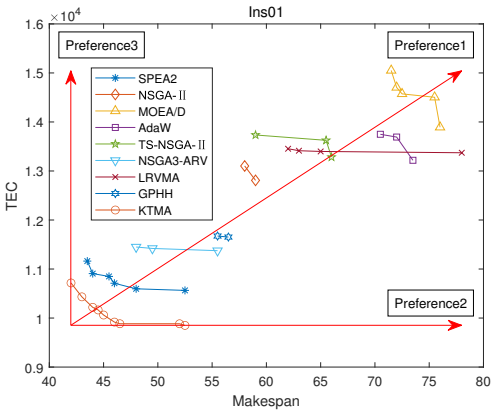


Figure 11: Comparison figure of all algorithms with different preferences.

Table 12: The optimal objective values of all comparison algorithms under different preferences.

		SPEA2	NSGA-II	MOEA/D	AdaW	TS-NSGA-II	NSGA-III/ARV	LRVMA	GPHH	KTMA
Preference1	C_{max}	48.00	59.00	75.50	70.50	66.00	55.50	65.00	55.50	**44.50**
	TEC	1.06E+04	1.28E+04	1.45E+04	1.37E+04	1.33E+04	1.14E+04	1.34E+04	1.17E+04	**1.02E+04**
Preference2	C_{max}	43.50	58.00	71.50	70.50	59.00	48.00	62.00	55.50	**42.00**
Preference3	**TEC**	1.06E+04	1.28E+04	1.39E+04	1.32E+04	1.33E+04	1.14E+04	1.34E+04	1.17E+04	**9.85E+03**

5. Conclusion

This paper proposed a knowledge-driven two-stage memetic algorithm for energy-efficient flexible job shop scheduling with machine breakdowns, aiming at optimizing makespan and **TEC** simultaneously. A two-stage framework is utilized to enhance convergence performance and diversity performance. Meanwhile, a rescheduling strategy is developed to be applied to reschedule the scheduling sequence when machine breakdowns occur. In the first stage, a hybrid initialization strategy is proposed to obtain a high-quality initial population. Then, a knowledge-driven variable neighborhood search is represented that combines four problem-specific operators to accelerate the convergence speed and fully exploit the solution space. In the second stage, an energy-saving strategy, including two types of strategies, is designed to further reduce **TEC** without increasing the makespan. Comprehensive experiments confirmed that the proposed KTMA significantly outperforms other comparison algorithms in solving EMBFJSP and the solutions obtained by the KTMA have better convergence performance and diversity performance.

For future work, several directions can be further studied: (i) designing more effective energy-saving strategies for EMBFJSP; (ii) considering more reliable rescheduling strategies to handle the situation of machine breakdowns; (iii) combining reinforcement learning technology to solve EMBFJSP; (iv) studying on other job shop scheduling problems under different dynamic events; (v) Investigating scheduling problems with priority constraints.

Acknowledgment

This work was partly supported by the National Natural Science Foundation of China under Grant No. 62076225.

References

Abedi, M., Chiong, R., Noman, N., & Zhang, R. (2020). A multi-population, multi-objective memetic algorithm for energy-efficient job-shop scheduling with deteriorating machines. *Expert Systems with Applications*, *157*, 113348.

Afsar, S., Palacios, J. J., Puente, J., Vela, C. R., & González-Rodríguez, I. (2022). Multi-objective enhanced memetic algorithm for green job shop scheduling with uncertain times. *Swarm and Evolutionary Computation*, *68*, 101016.

An, Y., Chen, X., Gao, K., Li, Y., & Zhang, L. (2022). Multiobjective flexible job-shop rescheduling with new job insertion and machine preventive maintenance. *IEEE Transactions on Cybernetics*, (pp. 1–13).

Bhatt, N., & Chauhan, N. R. (2015). Genetic algorithm applications on job shop scheduling problem: A review. In *2015 International Conference on Soft Computing Techniques and Implementations* (*ICSCTI*) (pp. 7–14).

Caldeira, R. H., Gnanavelbabu, A., & Vaidyanathan, T. (2020). An effective backtracking search algorithm for multi-objective flexible job shop scheduling considering new job arrivals and energy consumption. *Computers & Industrial Engineering*, *149*, 106863.

Deb, K., Pratap, A., Agarwal, S., & Meyarivan, T. (2002). A fast and elitist multiobjective genetic algorithm: Nsga-ii. *IEEE Transactions on Evolutionary Computation*, *6*, 182–197.

Deng, Q., Gong, G., Gong, X., Zhang, L., Liu, W., & Ren, Q. (2017). A bee evolutionary guiding nondominated sorting genetic algorithm ii for multiobjective flexible job-shop scheduling. *Computational Intelligence and Neuroscience*, *2017*.

Duan, J., & Wang, J. (2021). Energy-efficient scheduling for a flexible job shop with machine breakdowns considering machine idle time arrangement and machine speed level selection. *Computers & Industrial Engineering*, *161*, 107677.

Duan, J., & Wang, J. (2022). Robust scheduling for flexible machining job shop subject to machine breakdowns and new job arrivals considering system reusability and task recurrence. *Expert Systems with Applications*, *203*, 117489.

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

Ferreira, C., Figueira, G., & Amorim, P. (2022). Effective and interpretable dispatching rules for dynamic job shops via guided empirical learning. *Omega*, *111*, 102643.

Gao, K., Cao, Z., Zhang, L., Chen, Z., Han, Y., & Pan, Q. (2019a). A review on swarm intelligence and evolutionary algorithms for solving flexible job shop scheduling problems. *IEEE/CAA Journal of Automatica Sinica*, *6*, 904–916.

Gao, K., Yang, F., Zhou, M., Pan, Q., & Suganthan, P. N. (2019b). Flexible job-shop rescheduling for new job insertion by using discrete jaya algorithm. *IEEE Transactions on Cybernetics*, *49*, 1944–1955.

Gong, G., Chiong, R., Deng, Q., Gong, X., Lin, W., Han, W., & Zhang, L. (2022). A two-stage memetic algorithm for energy-efficient flexible job shop scheduling by means of decreasing the total number of machine restarts. *Swarm and Evolutionary Computation*, *75*, 101131.

Gong, G., Deng, Q., Gong, X., & Huang, D. (2021). A non-dominated ensemble fitness ranking algorithm for multi-objective flexible job-shop scheduling problem considering worker flexibility and green factors. *Knowledge-Based Systems*, *231*, 107430.

Hatami, S., Ruiz, R., & Andrés-Romano, C. (2015). Heuristics and metaheuristics for the distributed assembly permutation flowshop scheduling problem with sequence dependent setup times. *International Journal of Production Economics*, *169*, 76–88.

Hidri, L., Al-Samhan, A. M., & Mabkhot, M. M. (2019). Bounding strategies for the parallel processors scheduling problem with no-idle time constraint, release date, and delivery time. *IEEE Access*, *7*, 170392–170405.

Lei, K., Guo, P., Wang, Y., Xiong, J., & Zhao, W. (2022). An end-to-end hierarchical reinforcement learning framework for large-scale dynamic flexible job-shop scheduling problem. In *2022 International Joint Conference on Neural Networks (IJCNN)* (pp. 1–8).

Li, K., Deng, Q., Zhang, L., Fan, Q., Gong, G., & Ding, S. (2021). An effective mcts-based algorithm for minimizing makespan in dynamic flexible job shop scheduling problem. *Computers & Industrial Engineering*, *155*, 107211.

Li, M., & Yao, X. (2020). What weights work for you? adapting weights for any pareto front shape in decomposition-based evolutionary multiobjective optimisation. *Evolutionary Computation*, *28*, 227–253.

Li, R., Gong, W., & Lu, C. (2022a). A reinforcement learning based rmoea/d for bi-objective fuzzy flexible job shop scheduling. *Expert Systems with Applications*, *203*, 117380.

Li, R., Gong, W., Lu, C., & Wang, L. (2022b). A learning-based memetic algorithm for energy-efficient flexible job shop scheduling with type-2 fuzzy processing time. *IEEE Transactions on Evolutionary Computation*, (pp. 1–1).

Li, Y., Huang, W., Wu, R., & Guo, K. (2020). An improved artificial bee colony algorithm for solving multi-objective low-carbon flexible job shop scheduling problem. *Applied Soft Computing*, *95*, 106544.

Liang, S., Yang, J., & Ding, T. (2022). Performance evaluation of ai driven low carbon manufacturing industry in china: An interactive network dea approach. *Computers & Industrial Engineering*, *170*, 108248.

Liang, X., Huang, Y., & Huang, M. (2020). Prediction of optimal rescheduling mode of flexible job shop under the arrival of a new job. In *2020 IEEE 8th International Conference on Computer Science and Network Technology (ICCSNT)* (pp. 55–58).

Lou, H., Wang, X., Dong, Z., & Yang, Y. (2022). Memetic algorithm based on learning and decomposition for multiobjective flexible job shop scheduling considering human factors. *Swarm and Evolutionary Computation*, *75*, 101204.

Lu, C., Gao, L., Gong, W., Hu, C., Yan, X., & Li, X. (2021a). Sustainable scheduling of distributed permutation flow-shop with non-identical factory using a knowledge-based multi-objective memetic optimization algorithm. *Swarm and Evolutionary Computation*, *60*, 100803.

Lu, C., Gao, L., Yi, J., & Li, X. (2021b). Energy-efficient scheduling of distributed flow shop with heterogeneous factories: A real-world case from automobile industry in china. *IEEE Transactions on Industrial Informatics*, *17*, 6687–6696.

Lu, C., Zhang, B., Gao, L., Yi, J., & Mou, J. (2022). A knowledge-based multiobjective memetic algorithm for green job shop scheduling with variable machining speeds. *IEEE Systems Journal*, *16*, 844–855.

Luo, S., Zhang, L., & Fan, Y. (2022). Real-time scheduling for dynamic partial-no-wait multiobjective flexible job shop by deep reinforcement learning. *IEEE Transactions on Automation Science and Engineering*, *19*, 3020–3038.

Ming, F., Gong, W., & Wang, L. (2022). A two-stage evolutionary algorithm with balanced convergence and diversity for many-objective optimization. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, *52*, 6222–6234.

Mohan, J., Lanka, K., & Rao, A. N. (2019). A review of dynamic job shop scheduling techniques. *Procedia Manufacturing*, *30*, 34–39.

Pan, Z., Lei, D., & Wang, L. (2022). A bi-population evolutionary algorithm with feedback for energy-efficient fuzzy flexible job shop scheduling. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, *52*, 5295–5307.

Roy, & K, R. (2001). *Design of experiments using the Taguchi approach: 16 steps to product and process improvement*. John Wiley & Sons.

Serrano-Ruiz, J. C., Mula, J., & Poler, R. (2022). Development of a multidimensional conceptual model for job shop smart manufacturing scheduling from the industry 4.0 perspective. *Journal of Manufacturing Systems*, *63*, 185–202.

Soofi, P., Yazdani, M., Amiri, M., & Adibi, M. A. (2021). Robust fuzzy-stochastic programming model and meta-heuristic algorithms for dual-resource constrained flexible job-shop scheduling problem under machine breakdown. *IEEE Access*, *9*, 155740–155762.

Wang, H., Cheng, J., Liu, C., Zhang, Y., Hu, S., & Chen, L. (2022). Multi-objective reinforcement learning framework for dynamic flexible job shop scheduling problem with uncertain events. *Applied Soft Computing*, *131*, 109717.

Wang, J.-j., & Wang, L. (2021). A bi-population cooperative memetic algorithm for distributed hybrid flow-shop scheduling. *IEEE Transactions on Emerging Topics in Computational Intelligence*, *5*, 947–961.

Wang, X., Gao, L., Zhang, C., & Shao, X. (2010). A multi-objective genetic algorithm based on immune and entropy principle for flexible job-shop scheduling problem. *The International Journal of Advanced Manufacturing Technology*, *51*, 757–767.

Wei, Z., Liao, W., & Zhang, L. (2022). Hybrid energy-efficient scheduling measures for flexible job-shop problem with variable machining speeds. *Expert Systems with Applications*, *197*, 116785.

While, L., Hingston, P., Barone, L., & Huband, S. (2006). A faster algorithm for calculating hypervolume. *IEEE Transactions on Evolutionary Computation*, *10*, 29–38.

Wu, X., & Sun, Y. (2018). A green scheduling algorithm for flexible job shop with energy-saving measures. *Journal of Cleaner Production*, *172*, 3249–3264.

Wu, Z., Sun, S., & Xiao, S. (2018). Risk measure of job shop scheduling with random machine breakdowns. *Computers & Operations Research*, *99*, 1–12.

Zaharie, B., Işlak, G., Dehelean, C., & Miclea, L. (2017). A hierarchical approach of order acceptance and delivery date setting problems in the apparel industry. In *2017 18th International Carpathian Control Conference (ICCC)* (pp. 267–272).

Zhang, C., Li, P., Guan, Z., & Rao, Y. (2007). A tabu search algorithm with a new neighborhood structure for the job shop scheduling problem. *Computers & Operations Research*, *34*, 3229–3242.

Zhang, F., Mei, Y., Nguyen, S., & Zhang, M. (2021). Evolving scheduling heuristics via genetic programming with feature selection in dynamic flexible job-shop scheduling. *IEEE Transactions on Cybernetics*, *51*, 1797–1811.

Zhang, G., Lu, X., Liu, X., Zhang, L., Wei, S., & Zhang, W. (2022). An effective two-stage algorithm based on convolutional neural network for the bi-objective flexible job shop scheduling problem with machine breakdown. *Expert Systems with Applications*, *203*, 117460.

Zhang, Q., & Li, H. (2007). Moea/d: A multiobjective evolutionary algorithm based on decomposition. *IEEE Transactions on Evolutionary Computation*, *11*, 712–731.

Zhao, F., Ma, R., & Wang, L. (2022). A self-learning discrete jaya algorithm for multiobjective energy-efficient distributed no-idle flow-shop scheduling problem in heterogeneous factory system. *IEEE Transactions on Cybernetics*, *52*, 12675–12686.

Zheng, X.-L., & Wang, L. (2018). A collaborative multiobjective fruit fly optimization algorithm for the resource constrained unrelated parallel machine green scheduling problem. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, *48*, 790–800.

Zitzler, E., Deb, K., & Thiele, L. (2000). Comparison of multiobjective evolutionary algorithms: Empirical results. *Evolutionary Computation*, *8*, 173–195.

Zitzler, E., Laumanns, M., & Thiele, L. (2001). Spea2: Improving the strength pareto evolutionary algorithm. *TIK-report*, *103*.

Supplementary Files

Table S-I: Example of benchmark Ins01.

		M1	M2	M3	M4	M5	M6
J_1	$O_{1,1}$	5	–	4	–	–	–
	$O_{1,2}$	–	1	5	–	3	–
	$O_{1,3}$	–	–	4	–	–	2
	$O_{1,4}$	1	6	–	–	–	5
	$O_{1,5}$	–	–	1	–	–	–
	$O_{1,6}$	–	–	6	3	–	6
J_2	$O_{2,1}$	–	6	–	–	–	–
	$O_{2,2}$	–	–	1	–	–	–
	$O_{2,3}$	2	–	–	–	–	–
	$O_{2,4}$	–	6	–	6	–	–
	$O_{2,5}$	1	6	–	–	–	5
J_3	$O_{3,1}$	–	6	–	–	–	–
	$O_{3,2}$	–	–	4	–	–	2
	$O_{3,3}$	1	6	–	–	–	5
	$O_{3,4}$	–	6	4	–	–	6
	$O_{3,5}$	1	–	–	–	5	–
J_4	$O_{4,1}$	1	6	–	–	–	5
	$O_{4,2}$	–	6	–	–	–	–
	$O_{4,3}$	–	–	1	–	–	–
	$O_{4,4}$	–	1	5	–	3	–
	$O_{4,5}$	–	–	4	–	–	2
J_5	$O_{5,1}$	–	1	5	–	3	–
	$O_{5,2}$	1	6	–	–	–	5
	$O_{5,3}$	–	6	–	–	–	–
	$O_{5,4}$	5	–	4	–	–	–
	$O_{5,5}$	–	6	–	6	–	–
	$O_{5,6}$	–	6	4	–	–	6
J_6	$O_{6,1}$	–	–	4	–	–	2
	$O_{6,2}$	2	–	–	–	–	–
	$O_{6,3}$	–	6	4	–	–	6
	$O_{6,4}$	–	6	–	–	–	–
	$O_{6,5}$	1	6	–	–	–	5
	$O_{6,6}$	3	–	–	2	–	–
J_7	$O_{7,1}$	–	–	–	–	–	1
	$O_{7,2}$	3	–	–	2	–	–
	$O_{7,3}$	–	6	4	–	–	6
	$O_{7,4}$	6	6	–	–	1	–
	$O_{7,5}$	–	–	1	–	–	–
J_8	$O_{8,1}$	–	–	4	–	–	2
	$O_{8,2}$	–	6	4	–	–	6
	$O_{8,3}$	1	6	–	–	5	–
	$O_{8,4}$	–	6	–	–	–	–
	$O_{8,5}$	–	6	–	6	–	–
J_9	$O_{9,1}$	–	–	–	–	–	1
	$O_{9,2}$	1	–	–	–	5	–
	$O_{9,3}$	–	–	6	3	–	6
	$O_{9,4}$	2	–	–	–	–	–
	$O_{9,5}$	–	6	4	–	–	6
	$O_{9,6}$	–	6	–	6	–	–
J_{10}	$O_{10,1}$	–	–	4	–	–	2
	$O_{10,2}$	–	6	4	–	–	6
	$O_{10,3}$	–	1	5	–	3	–
	$O_{10,4}$	–	–	–	–	–	1
	$O_{10,5}$	–	6	–	6	–	–
	$O_{10,6}$	3	–	–	2	–	–

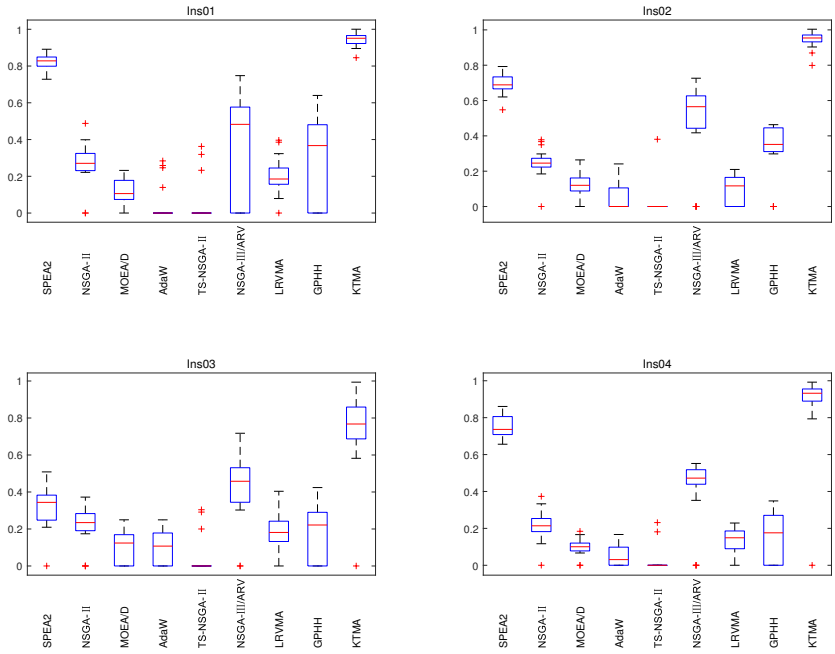


Figure S-1: Boxplot comparison of HV values of all comparison algorithms in the instances of Ins01 - Ins04.

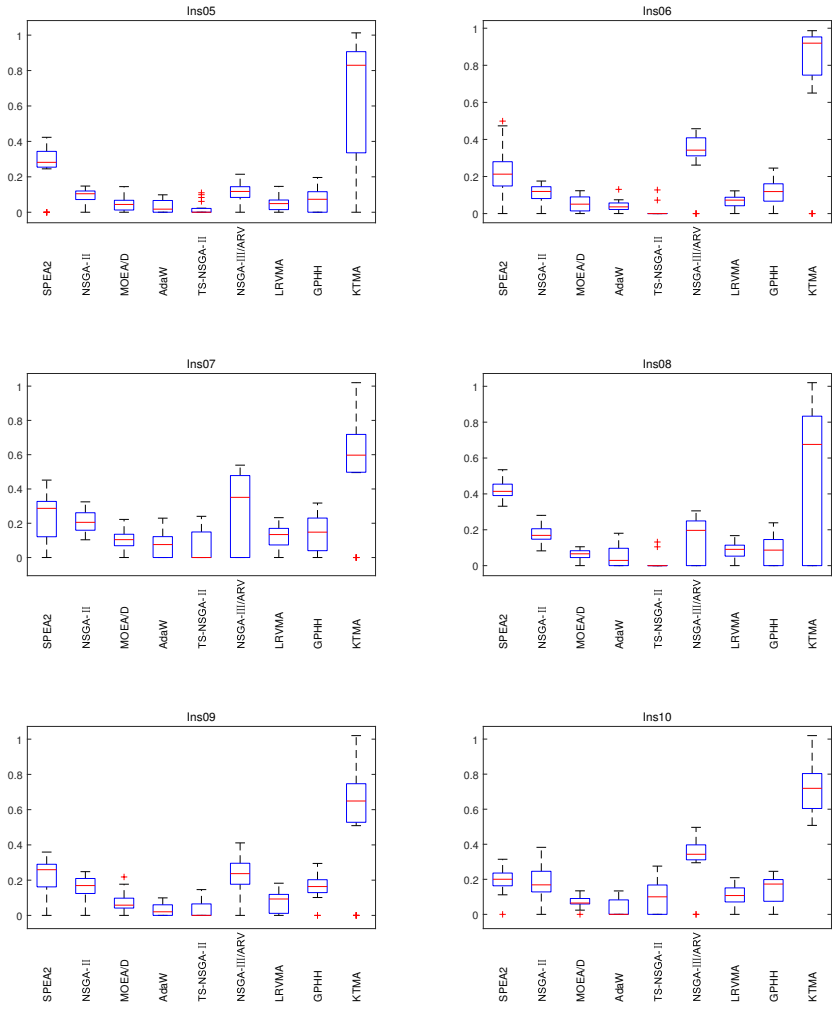


Figure S-2: Boxplot comparison of HV values of all comparison algorithms in the instances of Ins05 - Ins10.

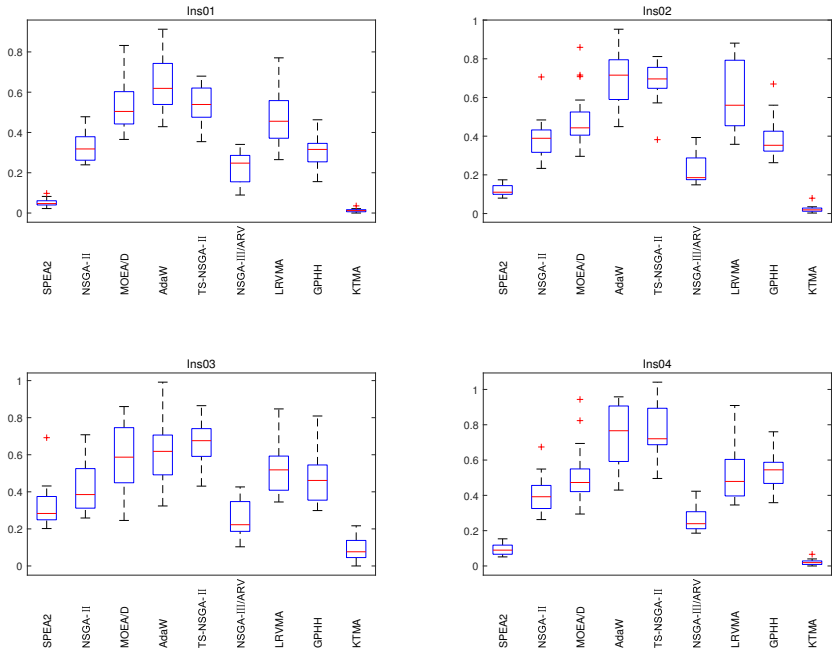


Figure S-3: Boxplot comparison of GD values of all comparison algorithms in the instances of Ins01 - Ins04.

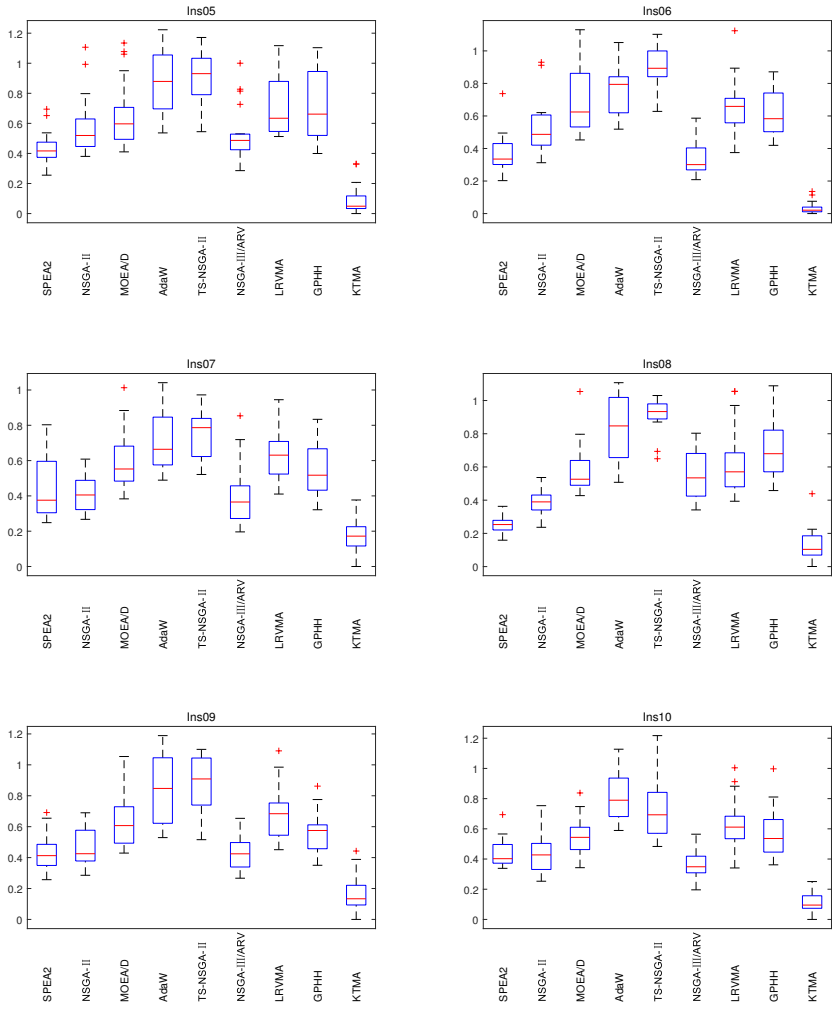


Figure S-4: Boxplot comparison of GD values of all comparison algorithms in the instances of Ins05 - Ins10.

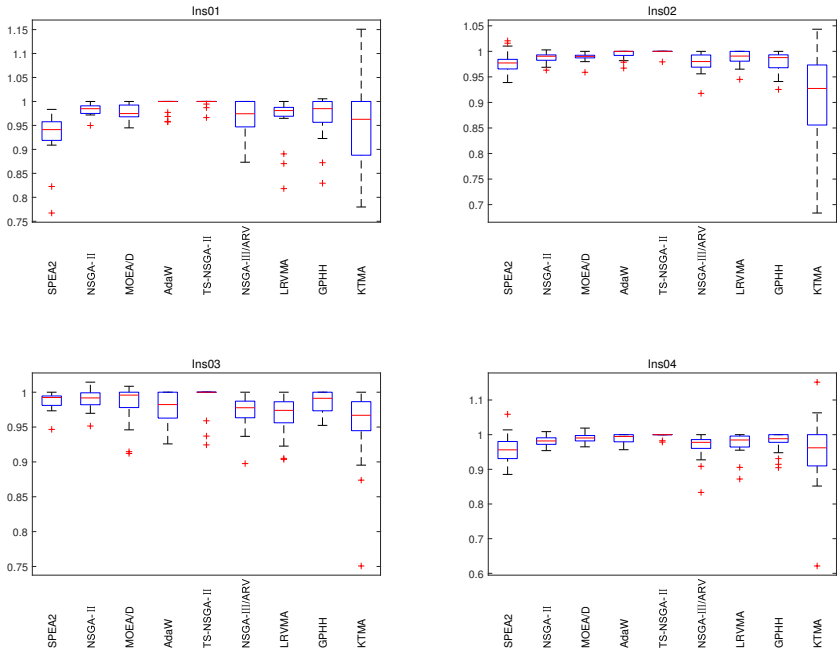


Figure S-5: Boxplot comparison of Spread values of all comparison algorithms in the instances of Ins01 - Ins04.

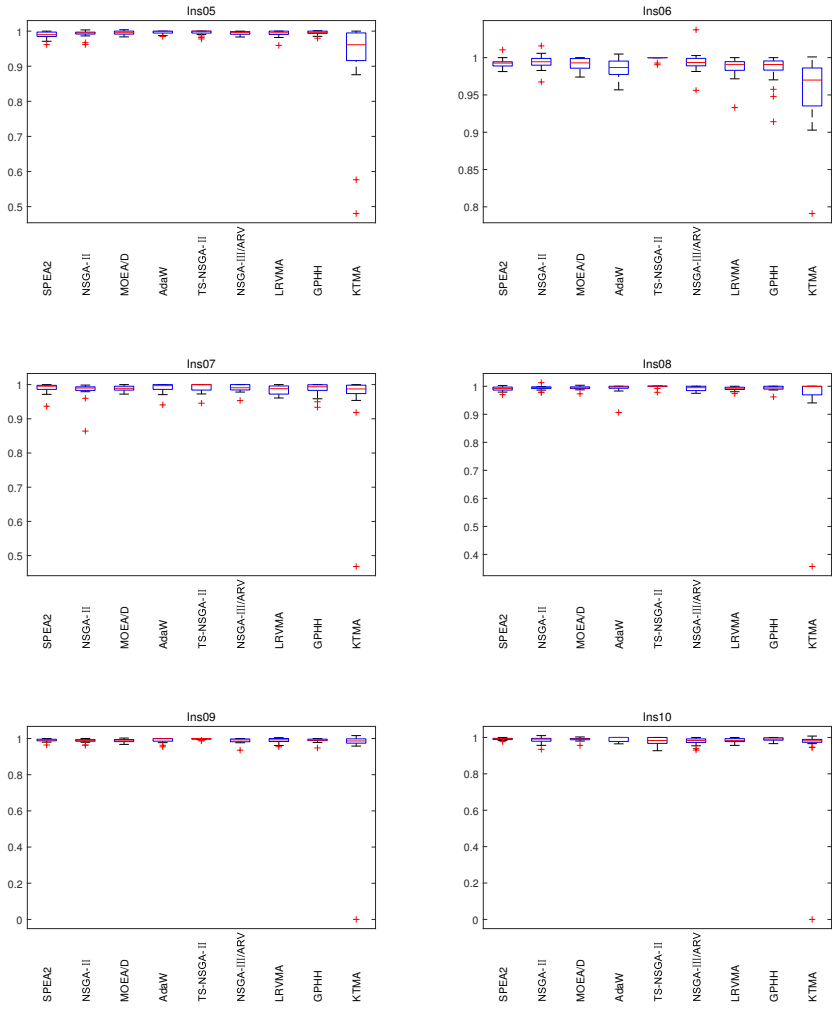


Figure S-6: Boxplot comparison of Spread values of all comparison algorithms in the instances of Ins05 - Ins10.